Table of Contents

[Section 13: Advance Concepts 2](#_Toc145625592)

[13.1.1 Authorization 2](#_Toc145625593)

[13.1.2 Session in .NET Core 3](#_Toc145625594)

[13.1.2.1 Configure Session in .NET Core 3](#_Toc145625595)

[13.1.2.2 Set and Access Session 4](#_Toc145625596)

[13.1.3 View Component 9](#_Toc145625597)

[13.2 Facebook Authentication 11](#_Toc145625598)

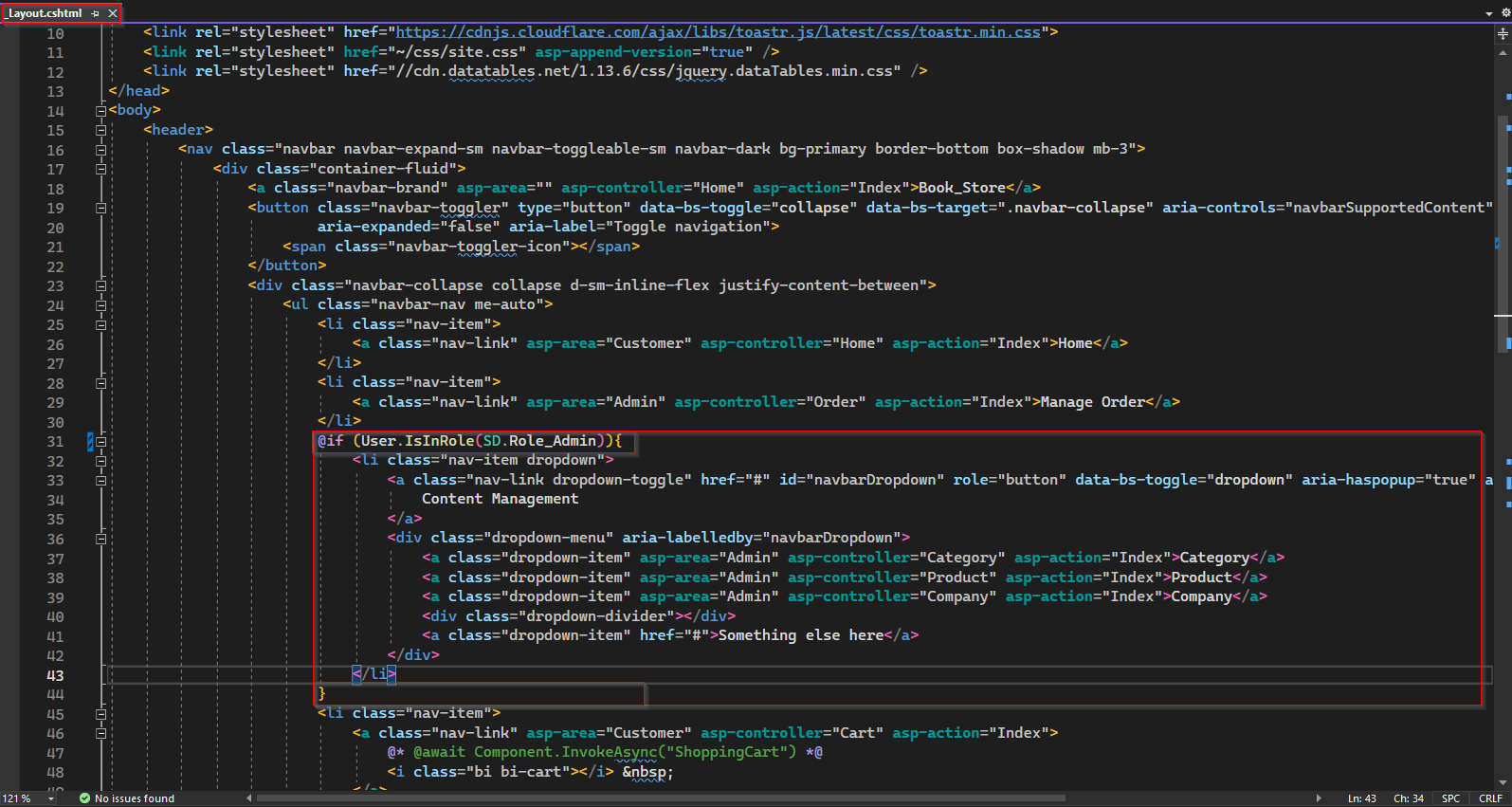
[13.2.2 Configure Facebook Authentication Service 21](#_Toc145625599)

[13.2.3 Add More Fields to External Login 24](#_Toc145625600)

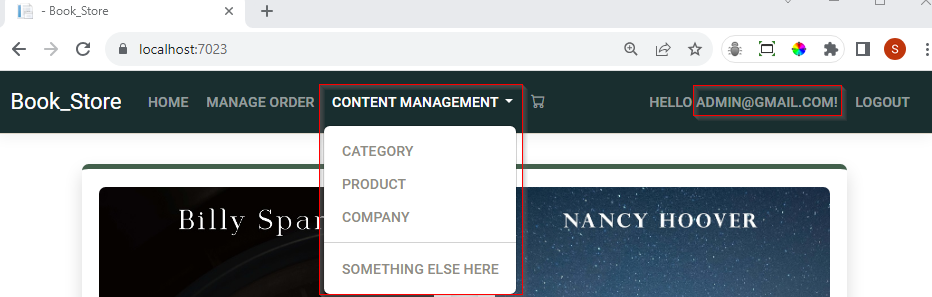
[3.3 Creating Admin and Employee Accounts 28](#_Toc145625601)

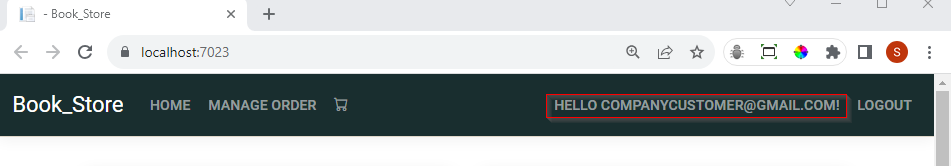
# Section 13: Advance Concepts

## 13.1.1 Authorization



To provide CRUD operation on Category, Product, Company, we can place "Content Management" drop down list of navigation bar inside if condition with statement verifying if user has role "Admin". Hence the "Content Management" will only be visible in user interface if user is under the role of "Admin".



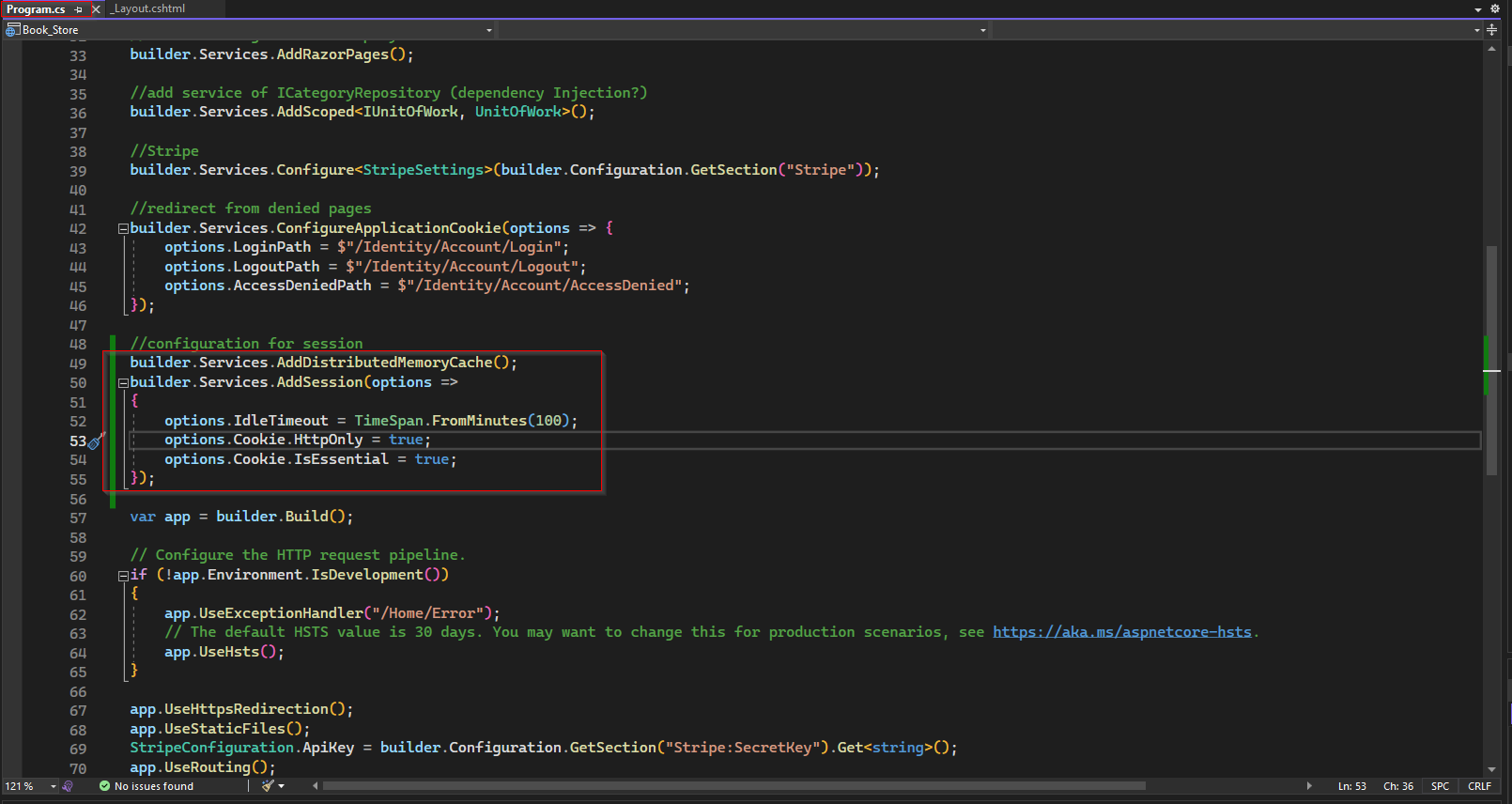


## 13.1.2 Session in .NET Core

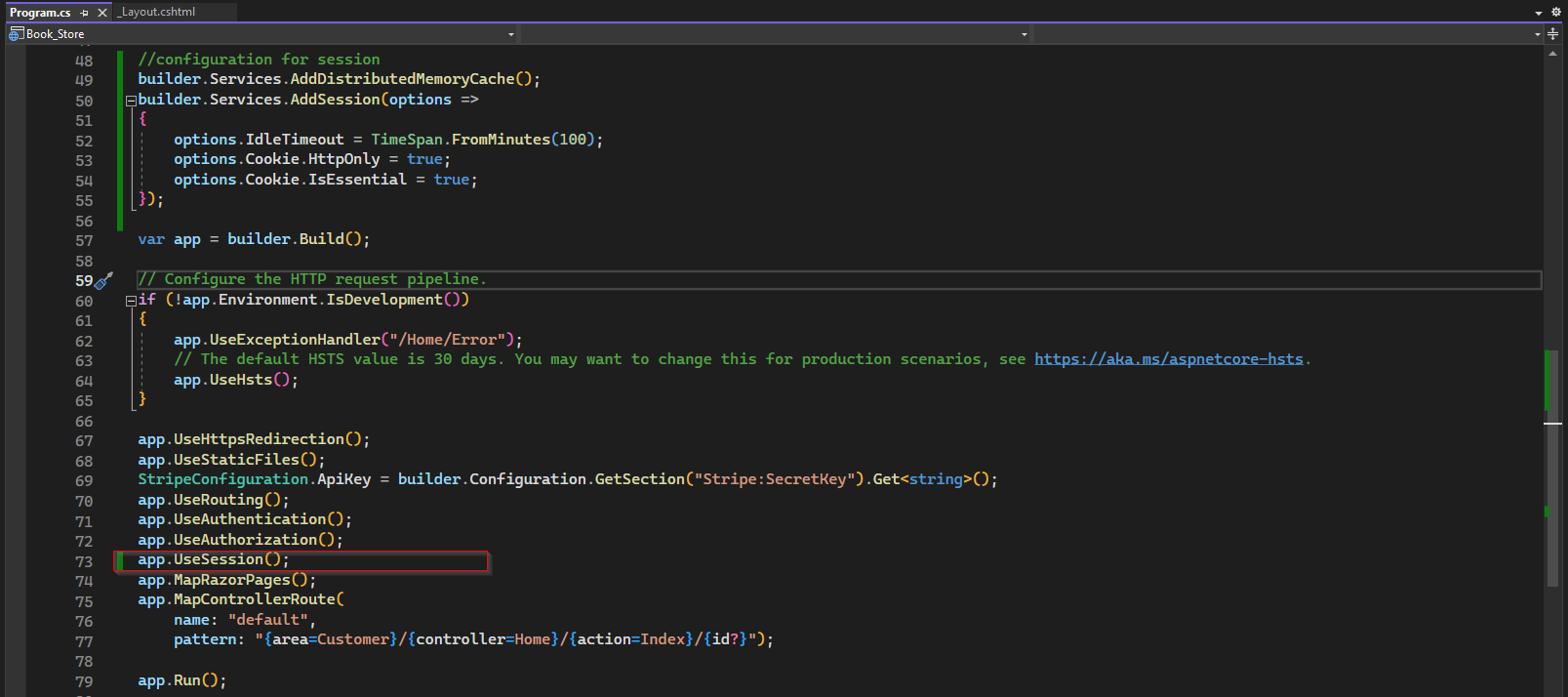
In real life e-commerce applications when a user adds or update items to the shopping cart, we can see count of distinct item along with shopping cart icon which is use as navigation button to redirect to shopping cart page. In this application, to achieve similar functionality we can use we can use session variable. But in a default .NET Core project, it is not added by default. Hence to use session in, we need to do some configuration.

### 13.1.2.1 Configure Session in .NET Core

To configure session in our application we need to add following two service in program.cs file.



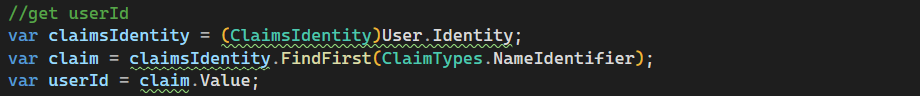
Then, after UseAuthorization , we need to add UseSession middleware as shown below.

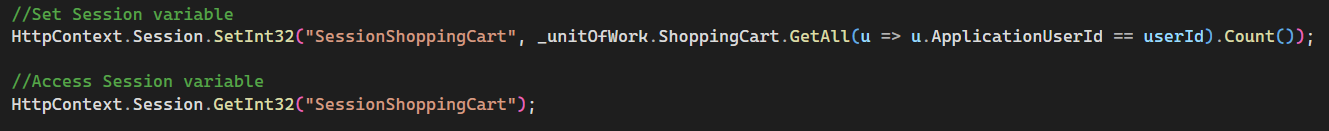


Now this application is configured to use session.

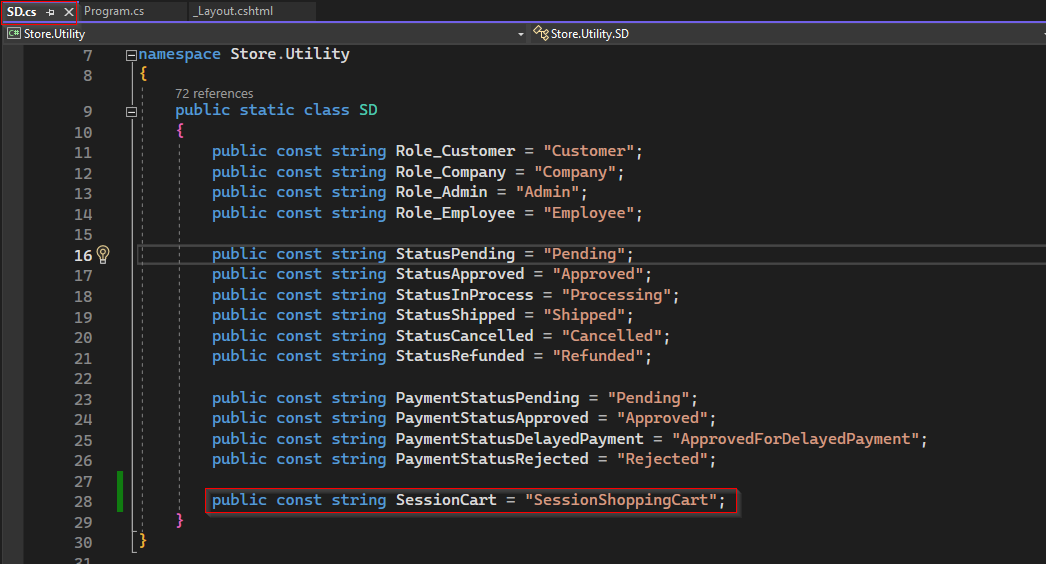
### 13.1.2.2 Set and Access Session

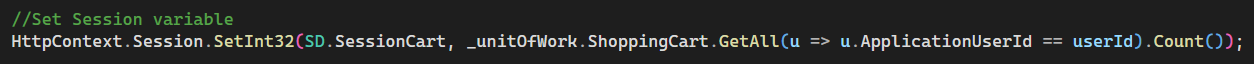
To set session variable, we use HttpContext and add a key value pair. Example session variable for number of distinct items in cart of current user.





We need to set, update, and access the session variable in several places, instead of using magic string "SessionShoppingCart", to avoid spelling mistake, we can add a constant and use it.

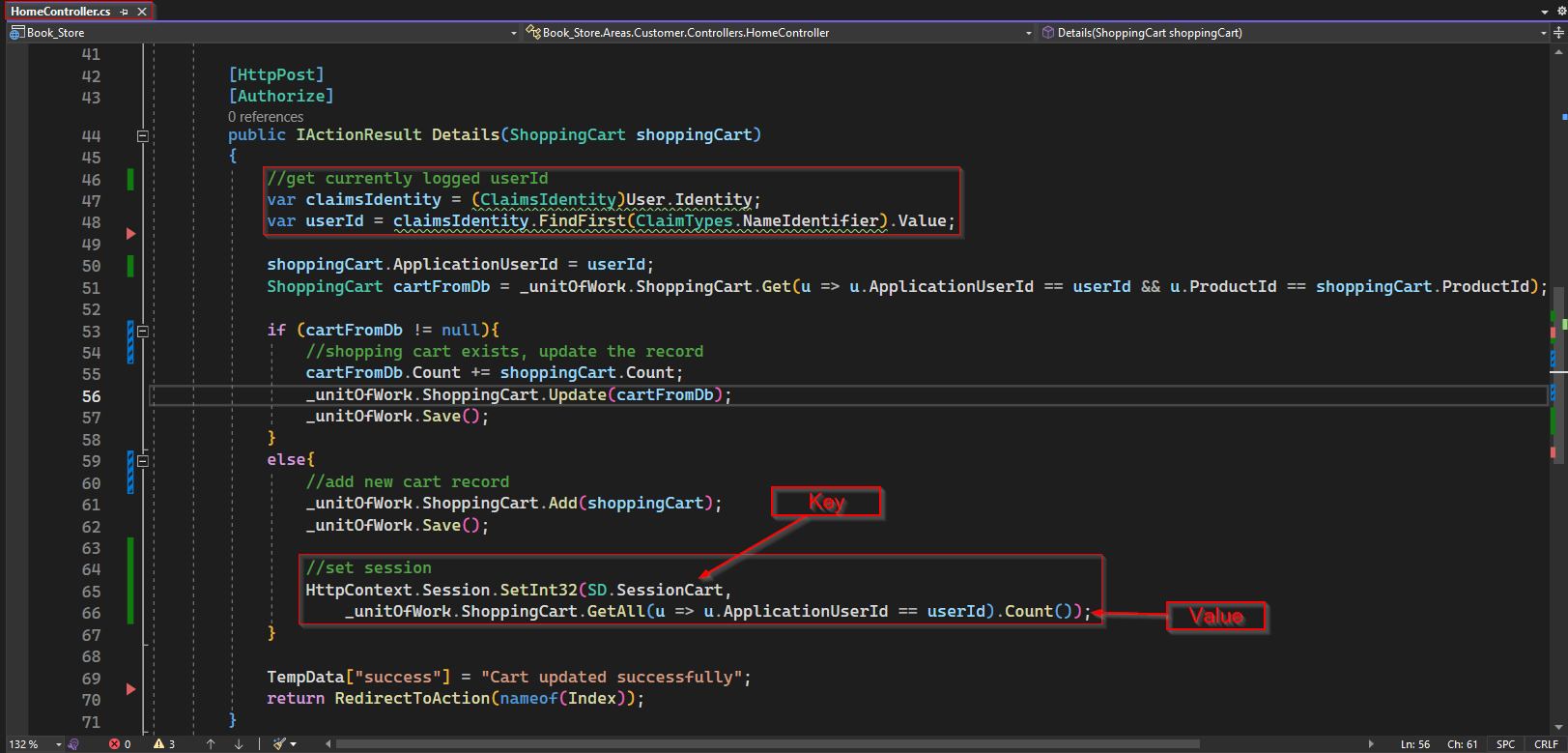




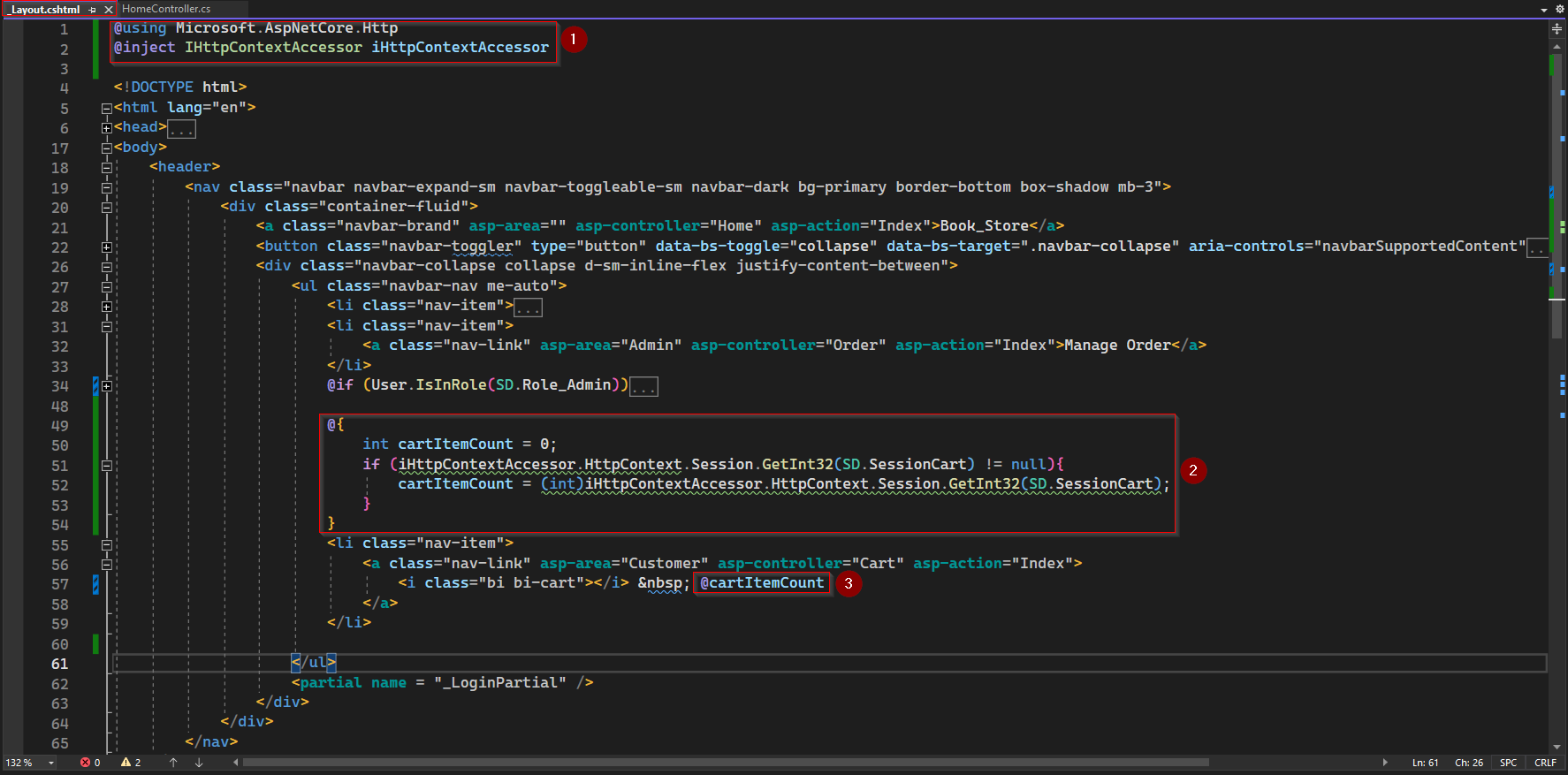


To display distinct cart items for current user. Basically, we first need to get number of distinct items for current user from database and set session variable each time when user logged in and update its value when user add item on cart or modify count or remove items from cart, and clear session variable when the user logged out.

#### Set Session while adding item on cart:

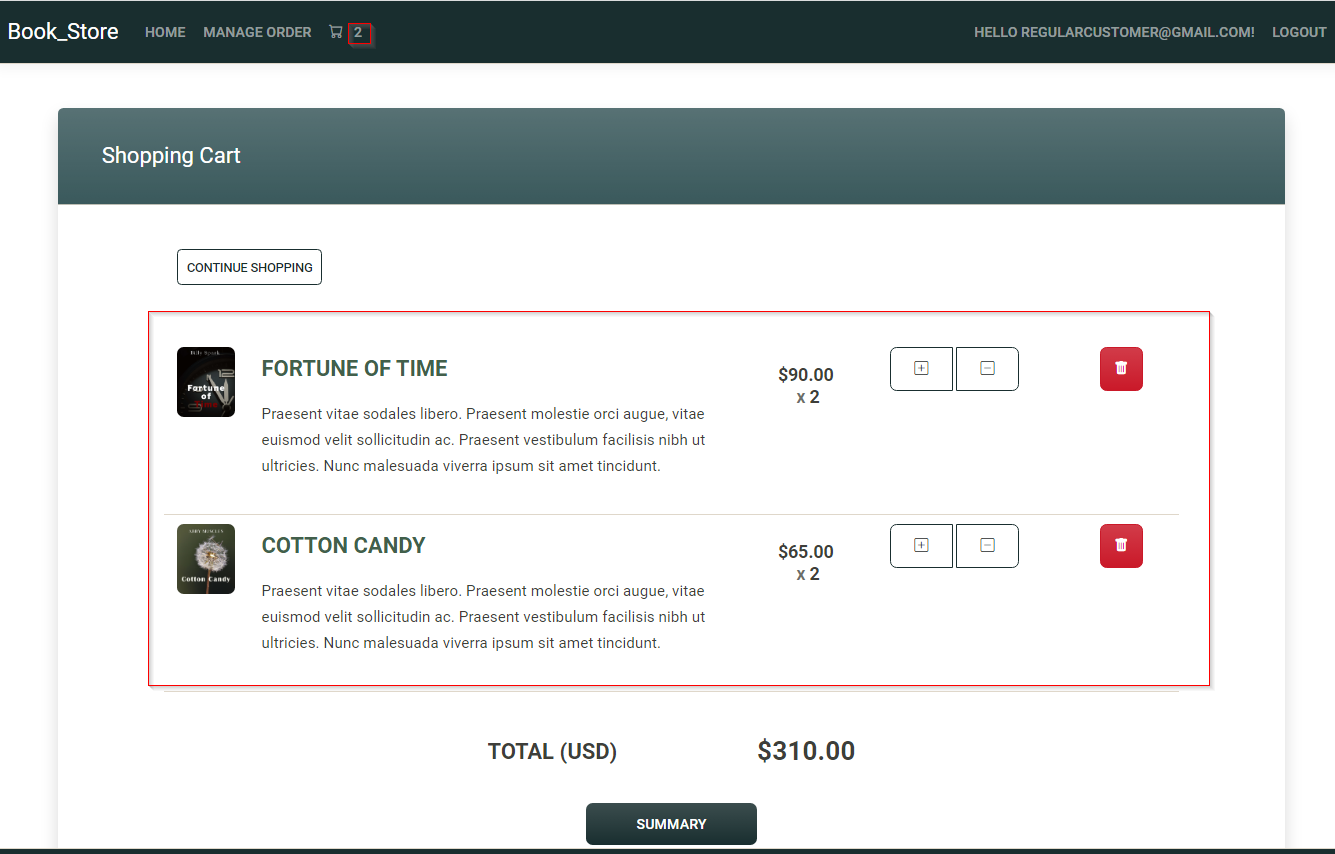


#### Access session variable on \_Layout page:

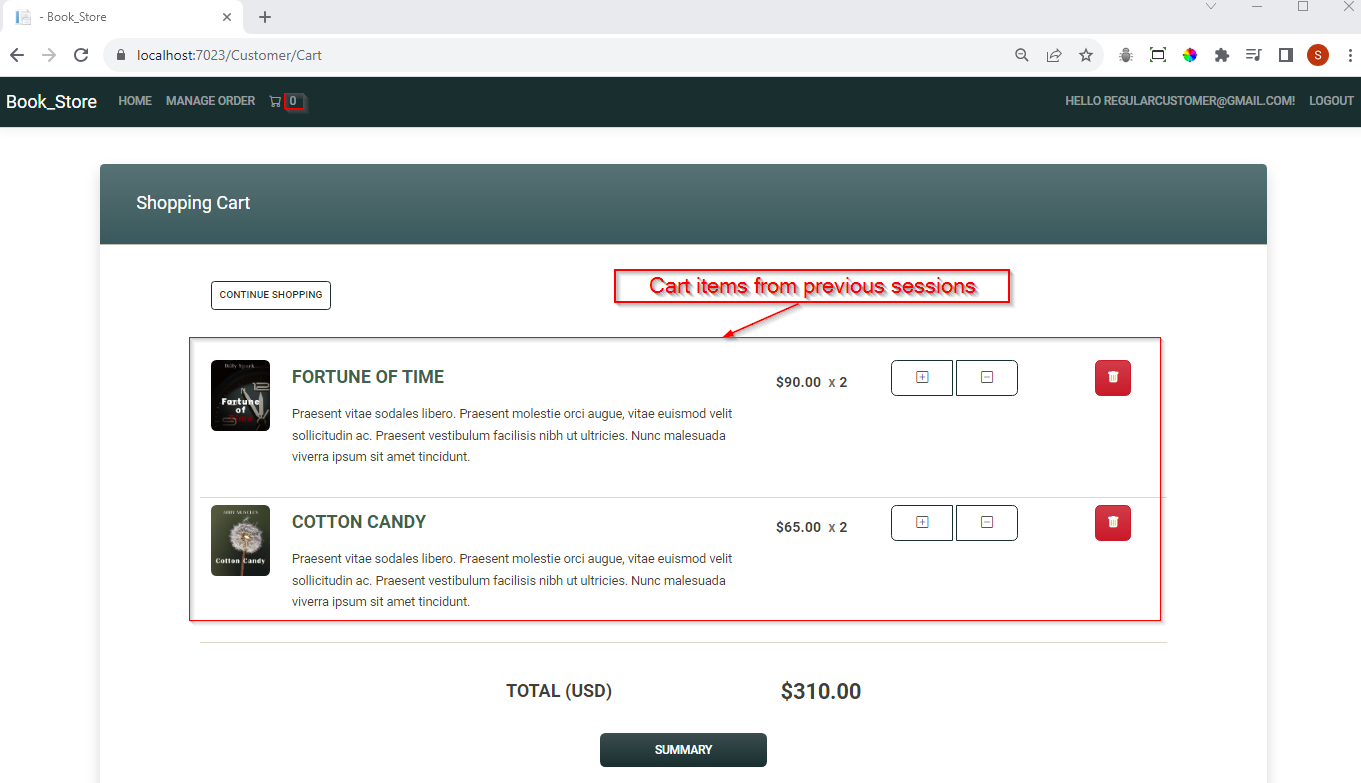


To access HttpContext in a view we need to explicitly inject that in the page. But before injecting iHttpContextAccessor, we first have to add a using statement for Microsoft.ASPNetCore.Http.

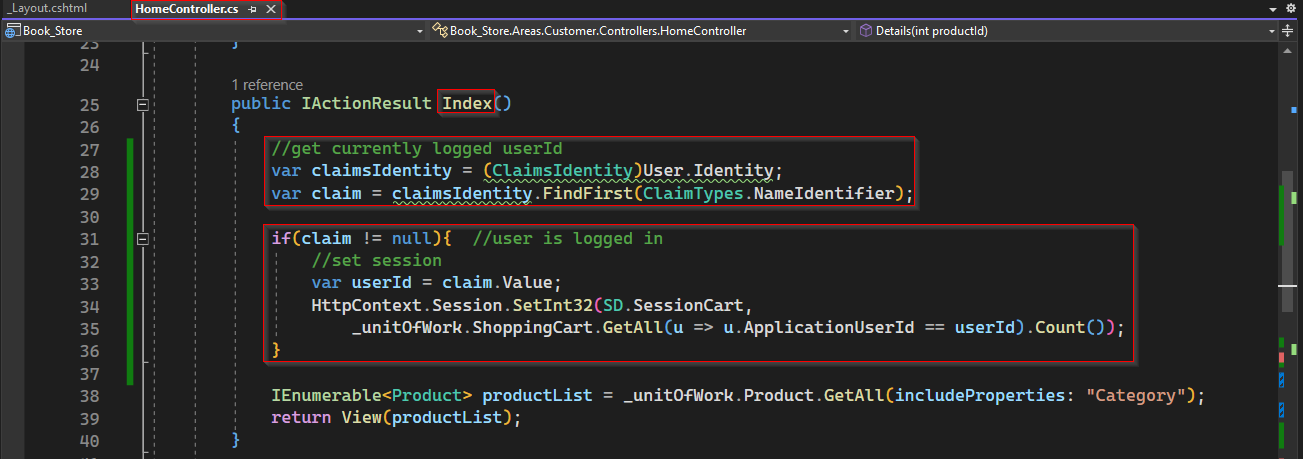
Using that context accessor now we can access the shopping cart session. In above screenshot, we have declared a variable cartItemCount of type integer and assigned default value zero to it. Then we have if statement to check is session have value for key SD.SessionCart and if yes, it will update the cartItemCount. In any case cartItemCount will be displayed along with cart icon on navigation bar.



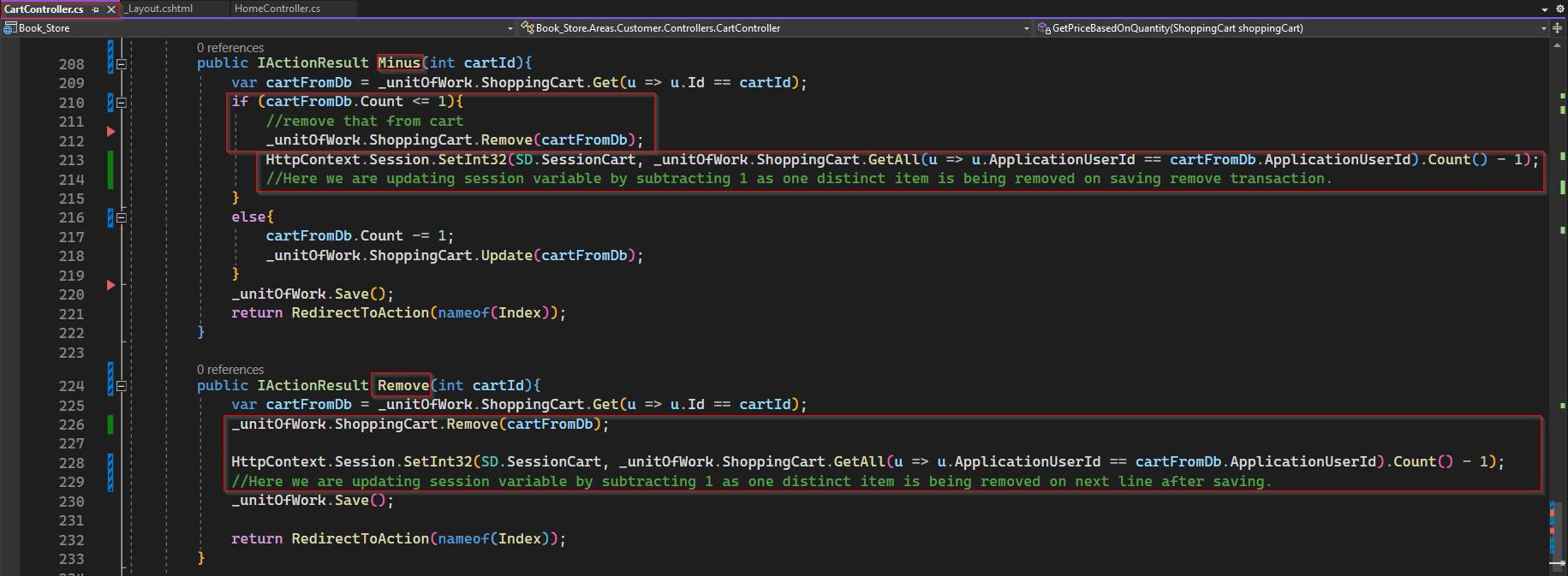
With above code application will display cart item when any user add item to cart, but it will not display item which was added on previous session or before login.



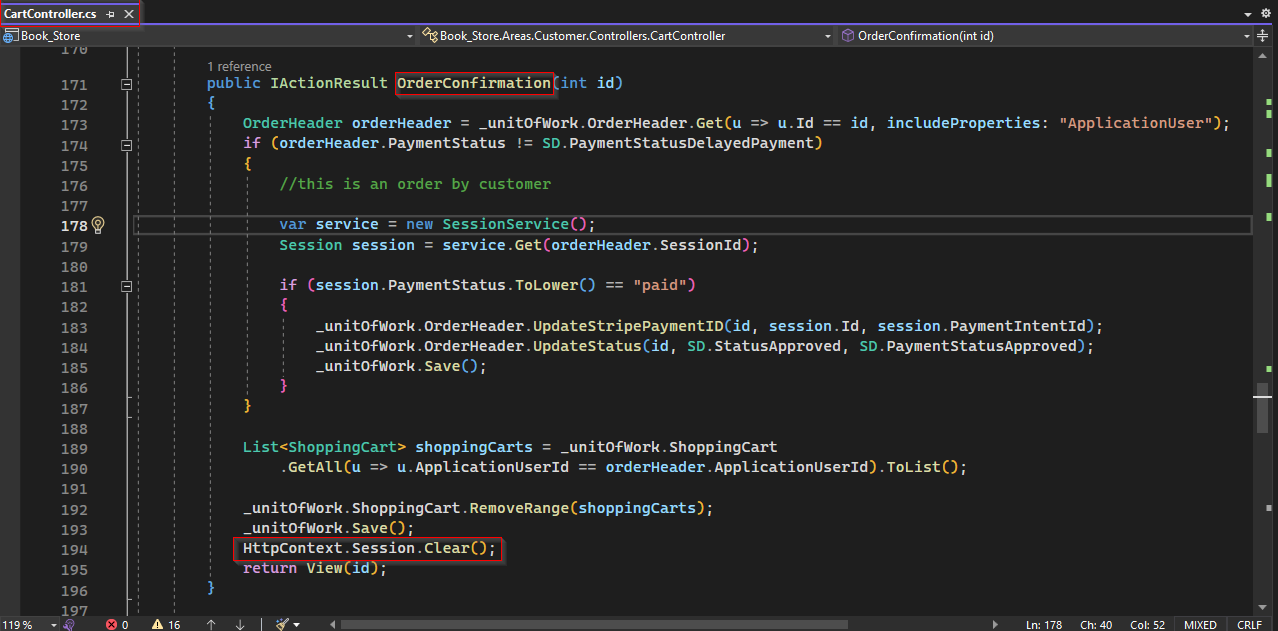
To correct above bug, we can set session while loading home page that mean on Action method Home/Index.



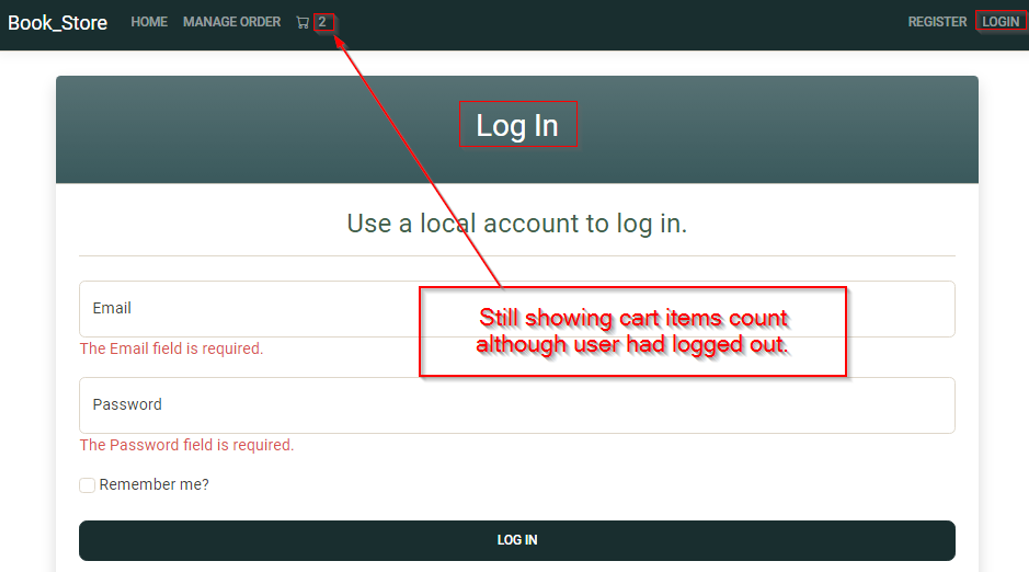
Next issue this functionality is whenever user delete items from the cart it will not update the value displayed. To solve the issue, we need to update session variable whenever any cart item is removed.



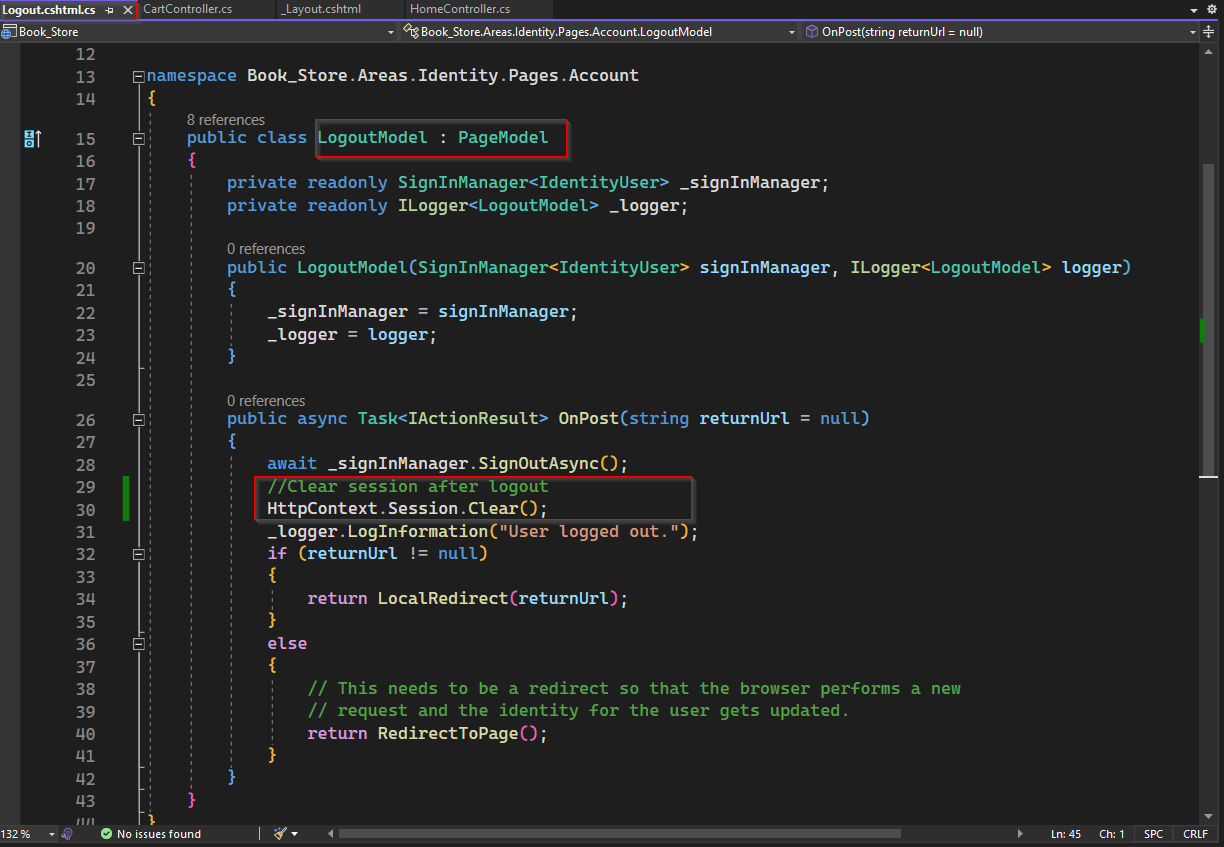
Next bug is whenever any customer submit order, all the item in the cart will be deleted but cart item count is not updated on navbar. This can be simply resolve by clearing the session.



Next issue for item count is whenever a user logged out with some item on cart, the item count will not erased.



To fix the issue, whenever user is logout, we can set the session variable to zero or instead, simply clear the session, which will remove all session variable. To do so we need to modify Logout.cshtml.cs which is the page model of Razor page Identity/Pages/Account.Logout.cshtml as shown below.



Now finally, count of distinct item on navigation bar seem working as expected.

Instead of going through all these hacky ways, it would be more straight forward if whenever \_Layout.cshtml loads it would automatically check the cart items count for current user and display it. But \_Layout.cshtml is a simple view file which do not have page model or code behind where we can write backend code as in Razor page, it is not possible with it. But to achieve same functionality we can using view component.

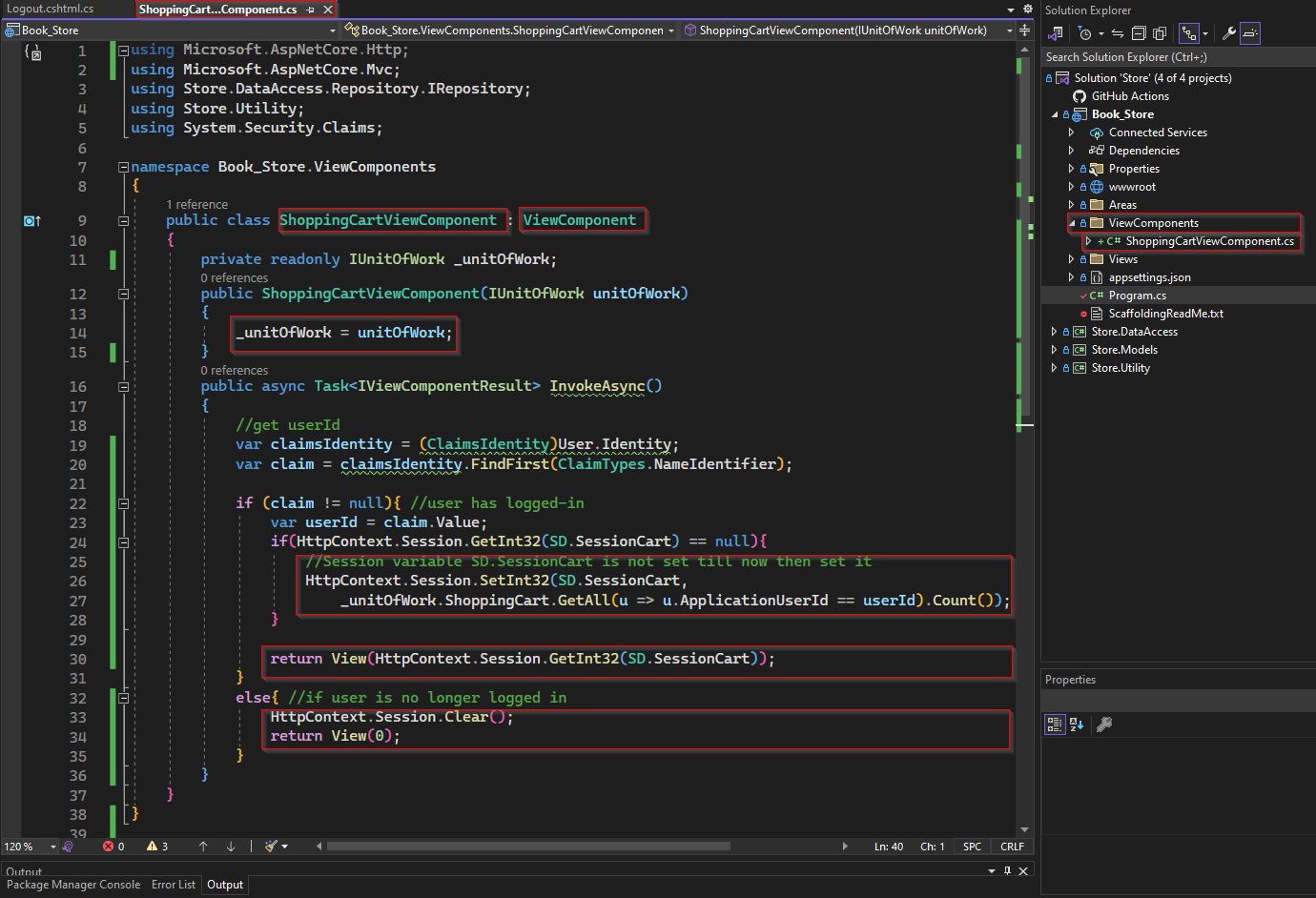
## 13.1.3 View Component

Now, view component is like a partial view. But the difference between them is, if you are working with partial view, you do not have a page model. While with view component will have a page model or a code behind file where you can write backend code for your UI.

It is one of the advanced concepts in MVC, to use view component we have do configuration after that is done, everything will come together.

To get started with view components, inside main web project, we need to create a folder and rename it as ViewComponents. Inside that folder, we will be adding our page model of our view components, which will basically be some Class file having name followed by special keyword "ViewComponent.cs" which must inherit the base class "ViewComponent" of Microsoft.AspNetCore.Mvc.

Now, this will be the backend file for our view component, so we need to think about what we want to do here. Basically, as we did earlier, we have to go to the shopping cart database and get the shopping cart for a logged-in user as shown below.



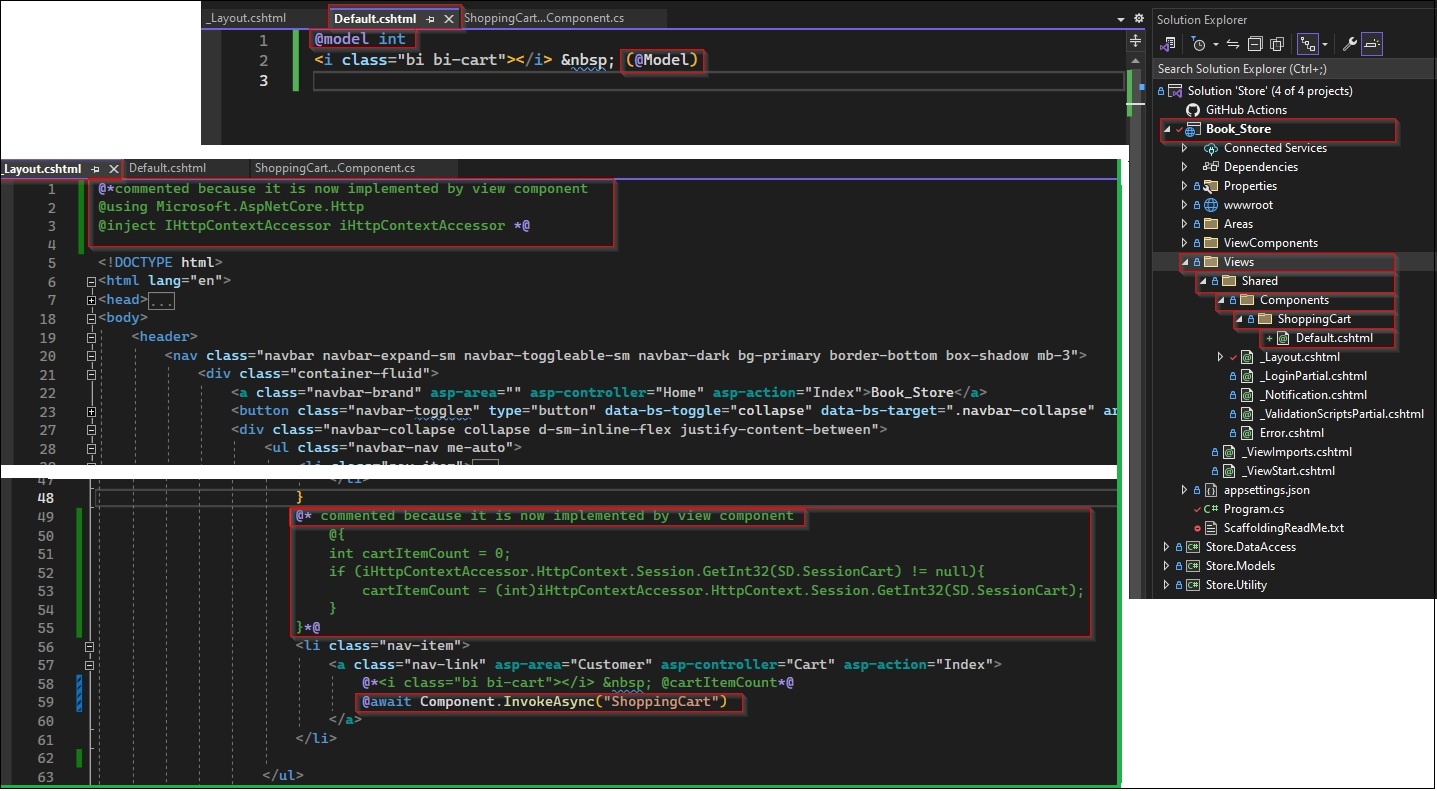
Inside above snippet of code, as we need to get data from database unitOfWork has been injected using dependency injection.

Then there is an async method handle the backend functionality of view component to be created. This method will return async Task, it can call as IActionResult, but as we know it will be returning view component, so return type has been explicitly declare as ViewComponentResult.

Now we need to create view for above page model. For that we need to create a folder inside "Views/Shared" under main web project and rename it as "Component". Now inside "Component", create a folder with same name of above page model i.e., "ShoppingCart.cshtml" and inside this folder we must create our view with name "Default.cshtml".

In Default.cshtml, we just need to add a cart icon and the populate value pass by invokeAsync method of view component.

And on \_Layout now de don’t need anymore IHttpContextAccessor and the if statement to access session variable. We just need to use view component replacing the cart icon.



**Note:**

* **Here naming file and folder is crucial and should be strictly followed.**
* **After implementing view component, if application couldn’t run successfully and some error encountered then clean solution and build it.**

Once view component is implementing as above,

* we can delete or comment all code we added to Logout.cshtml.cs, \_Layout.cshtml and action method Home/Index to display distinct cart item count on navigation bar.
* But we still need the code we added to action methods to add item to cart and update or remove items from cart.

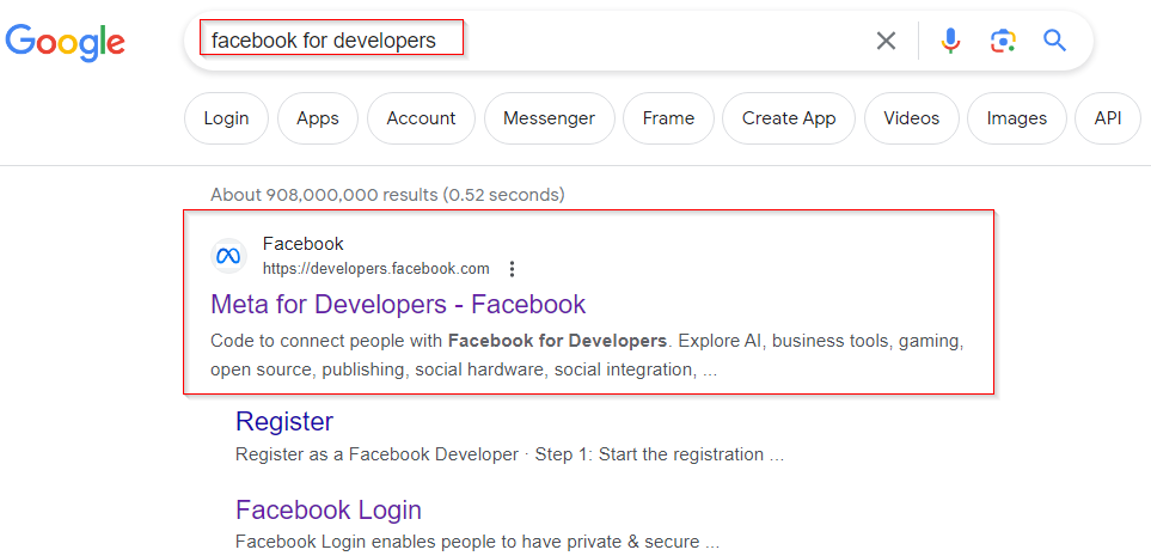
## 13.2 Facebook Authentication

Now a days it is common in today's website, when you try to register, we can use another service to register like Facebook, Google, LinkedIn, and other services that enable that feature. We can easily add that in .NET Core application.

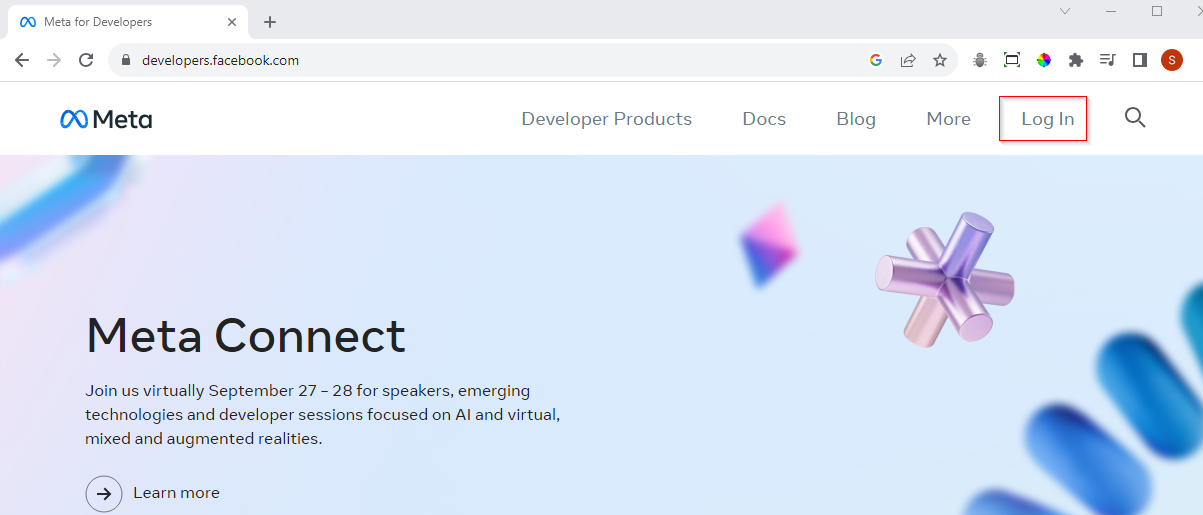
13.2.1 Facebook Login - App Setup

Here we will try to add Facebook login. For that we to <https://developers.facebook.com>.

It can find we easily from search engine typing keyword like "facebook for develop" or something similar.

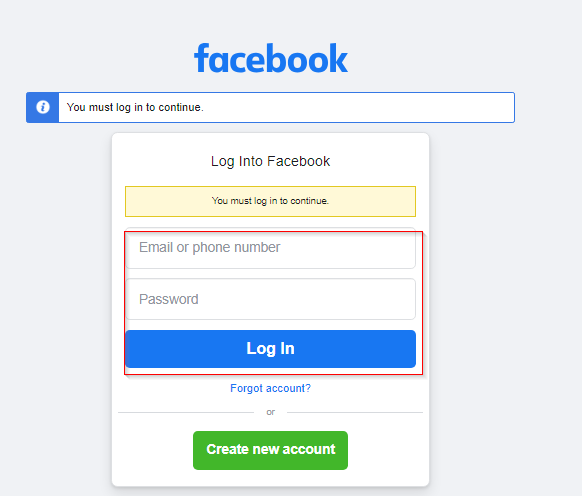


Click on search reasult as shown above.

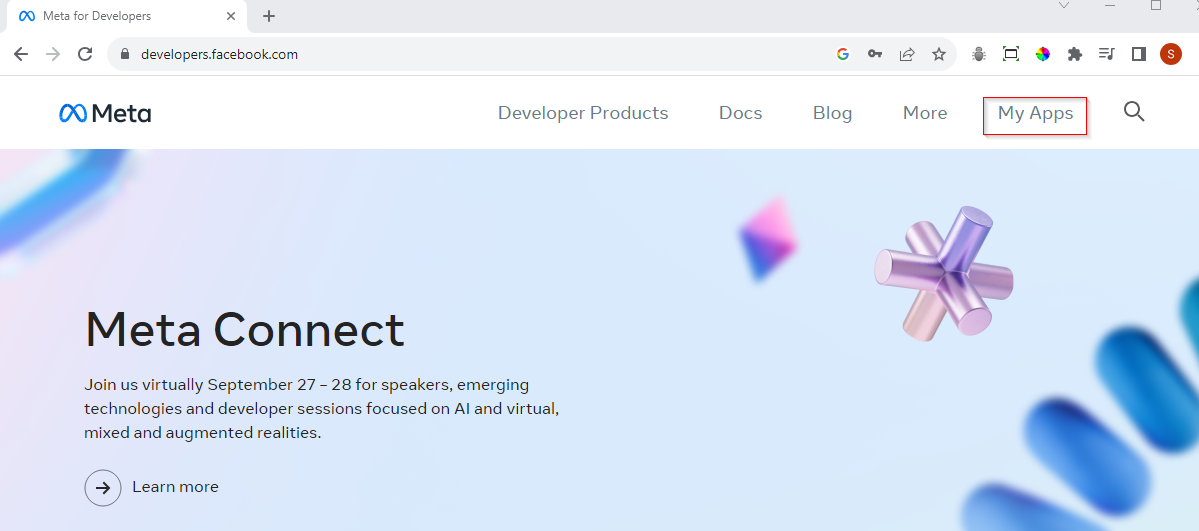


Now click on Log In button.

Here you can login if you have face book account or you can create new account

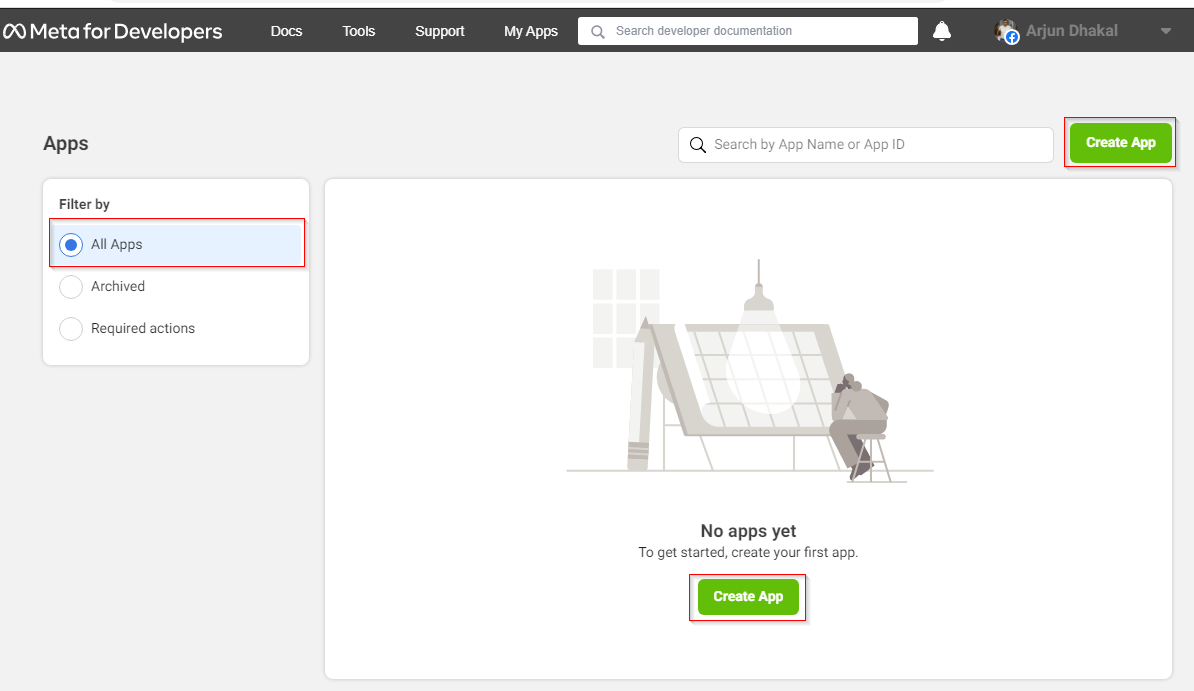


Once you login you will get following home page. Now click on "My Apps" button.

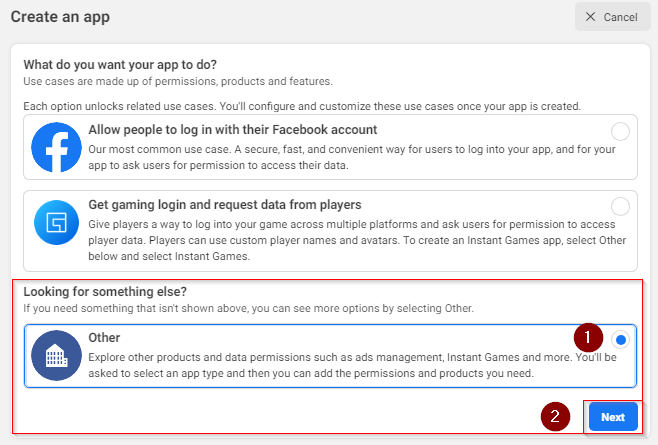


To add Facebook authentication service in our web application, first we need to create an App here and set it up to get an "App ID" and a "Secret Key" that will be used to configure service inside "Program.cs" file.

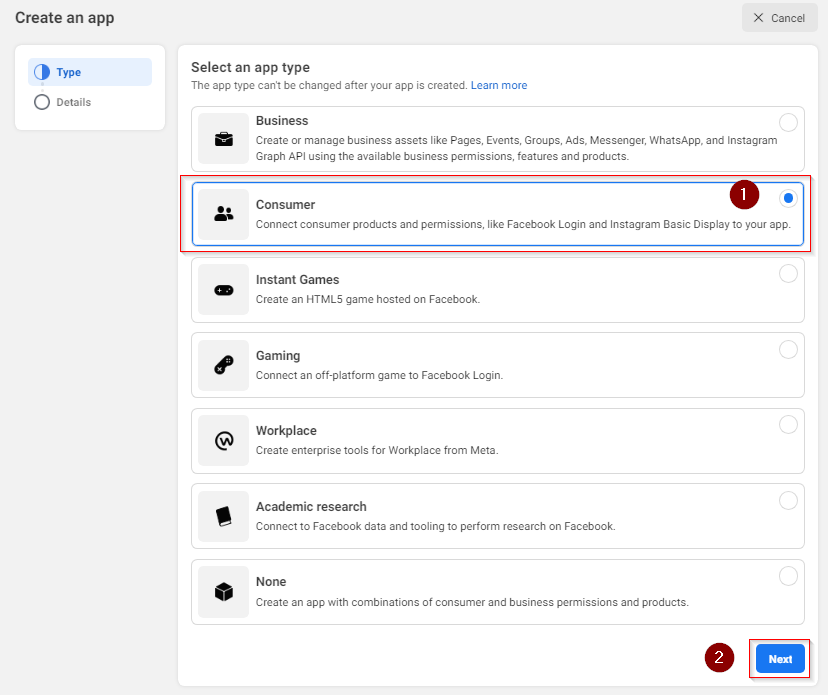
Now to create a new application, click any "Create App" button.



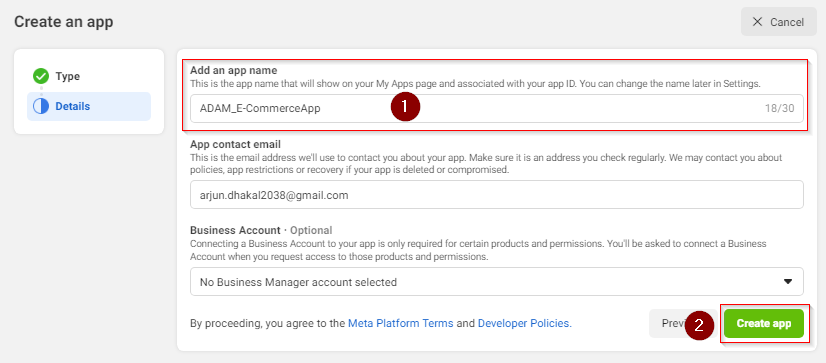
Now select on "Other" option and click next.



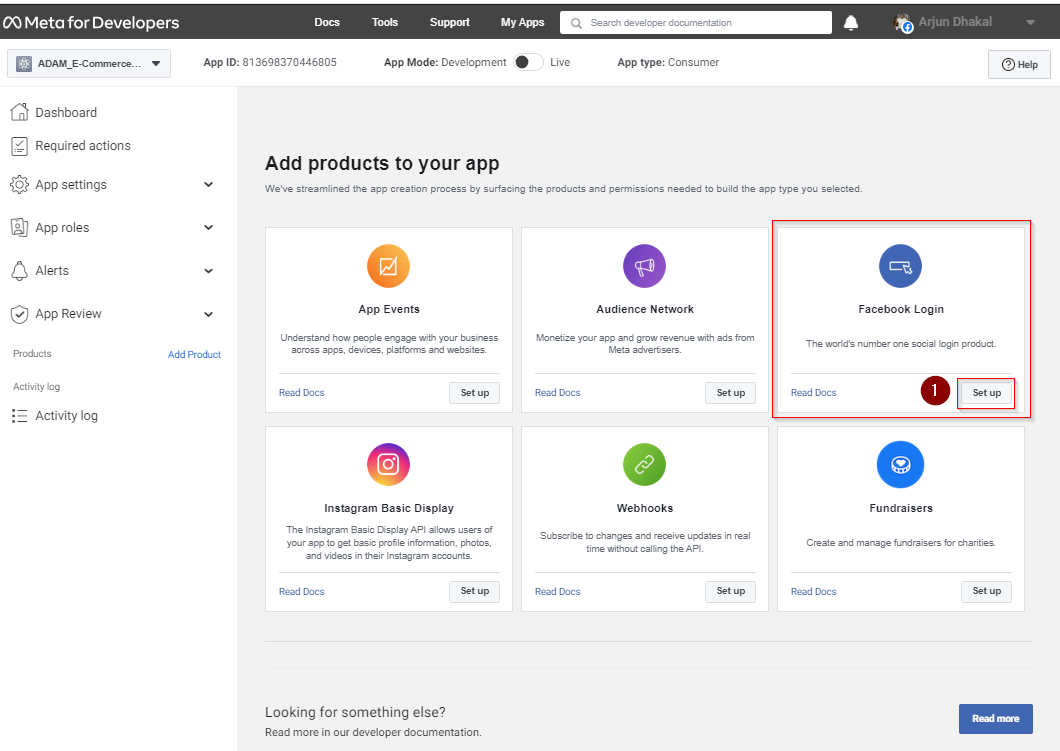
Here Select "Consumer" and then "Next".



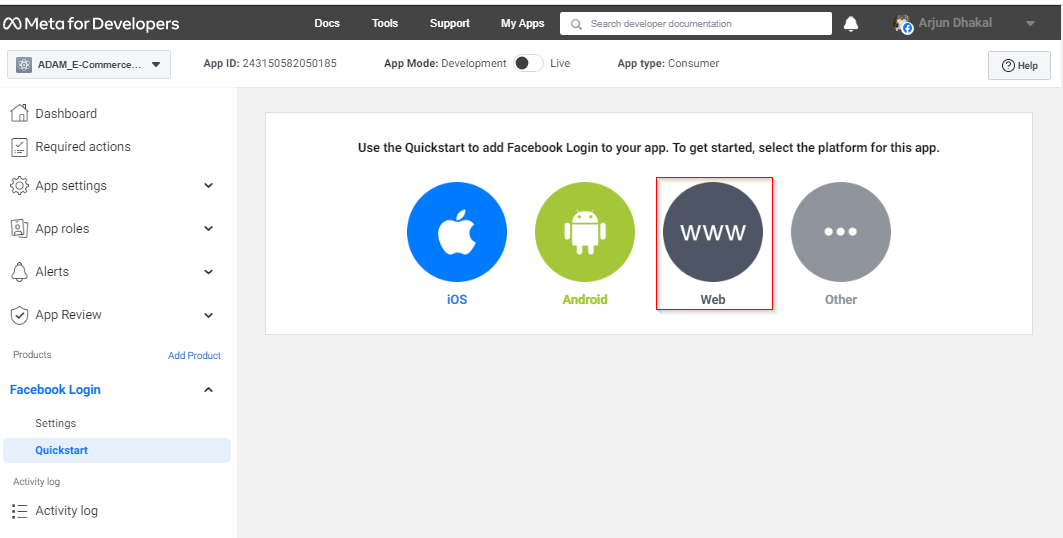
Now add any name to this new create App configuration and hit "Create App" button.



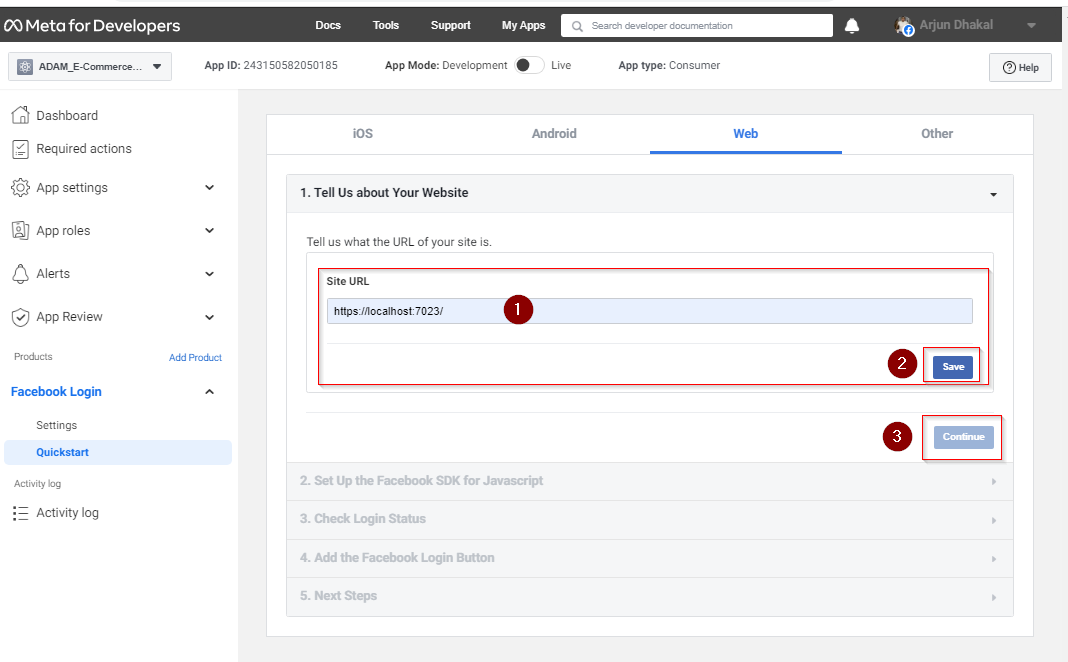
Once App is created and redirected inside it, we can see its name and Id below navigation. Now next step is to setup the newly created app and get "secret key". Now on click on "Set up" inside Facebook Login.

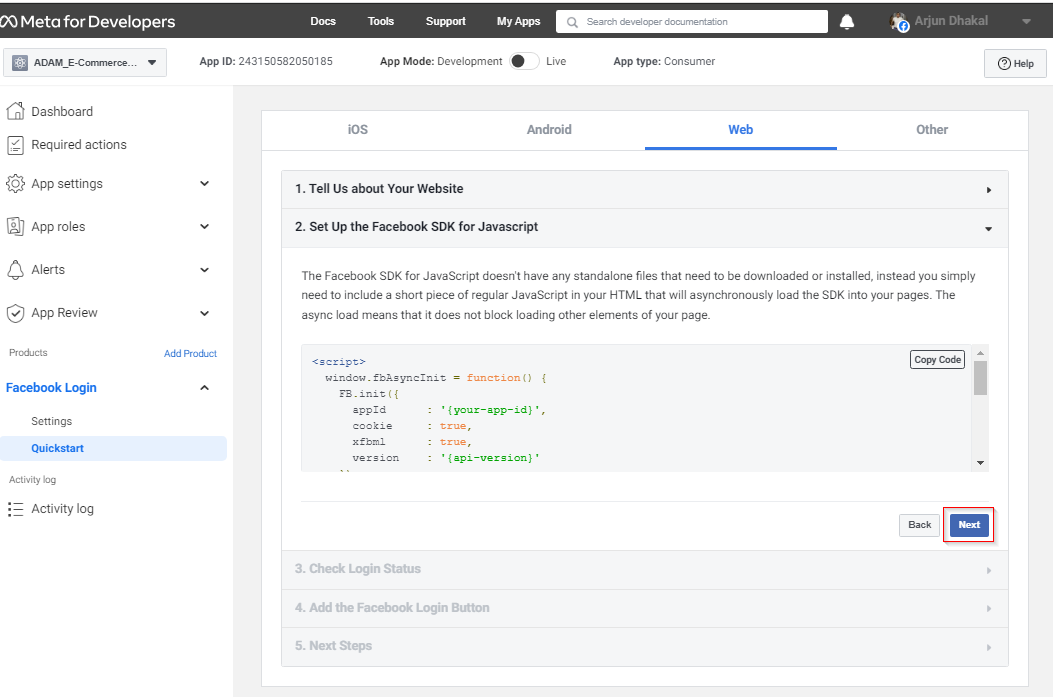


Now click on "WWW web" for web application.

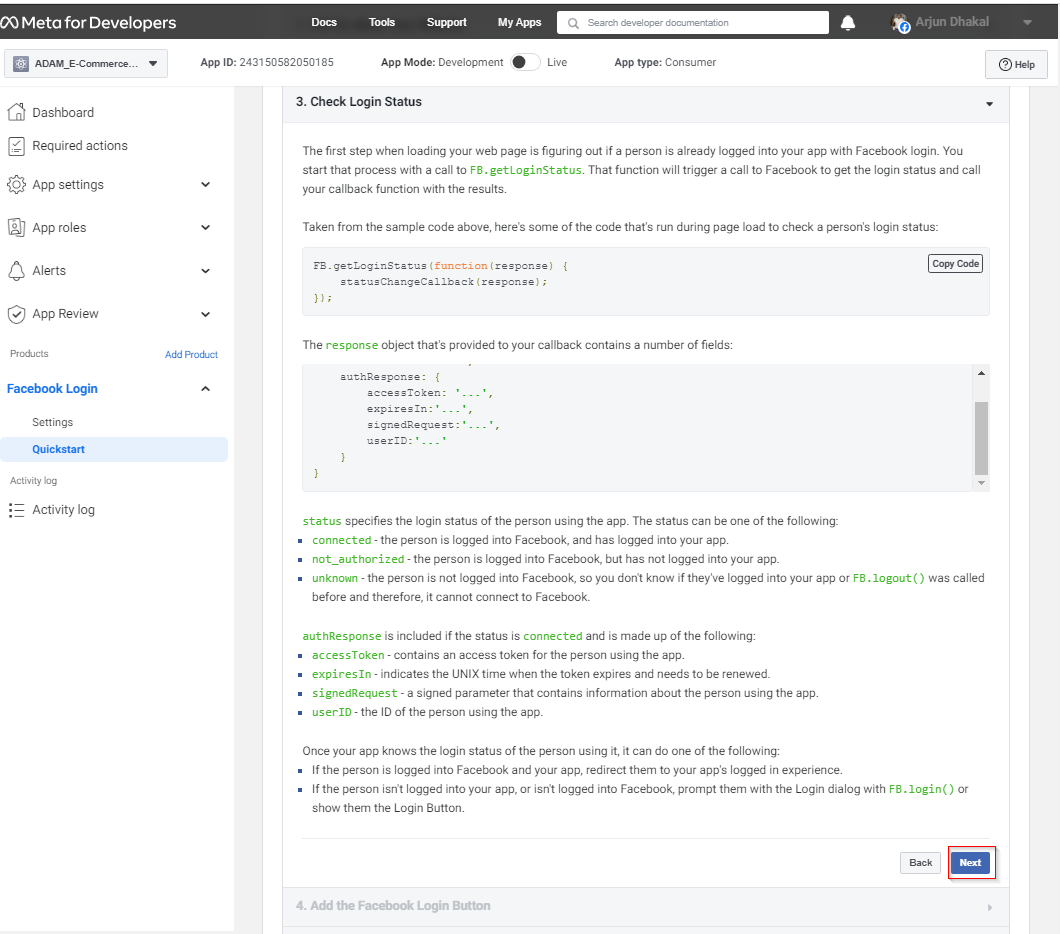


On "Tell Us about Your Website", we need to provide URL of your web app, then click "Save" and then "Continue".

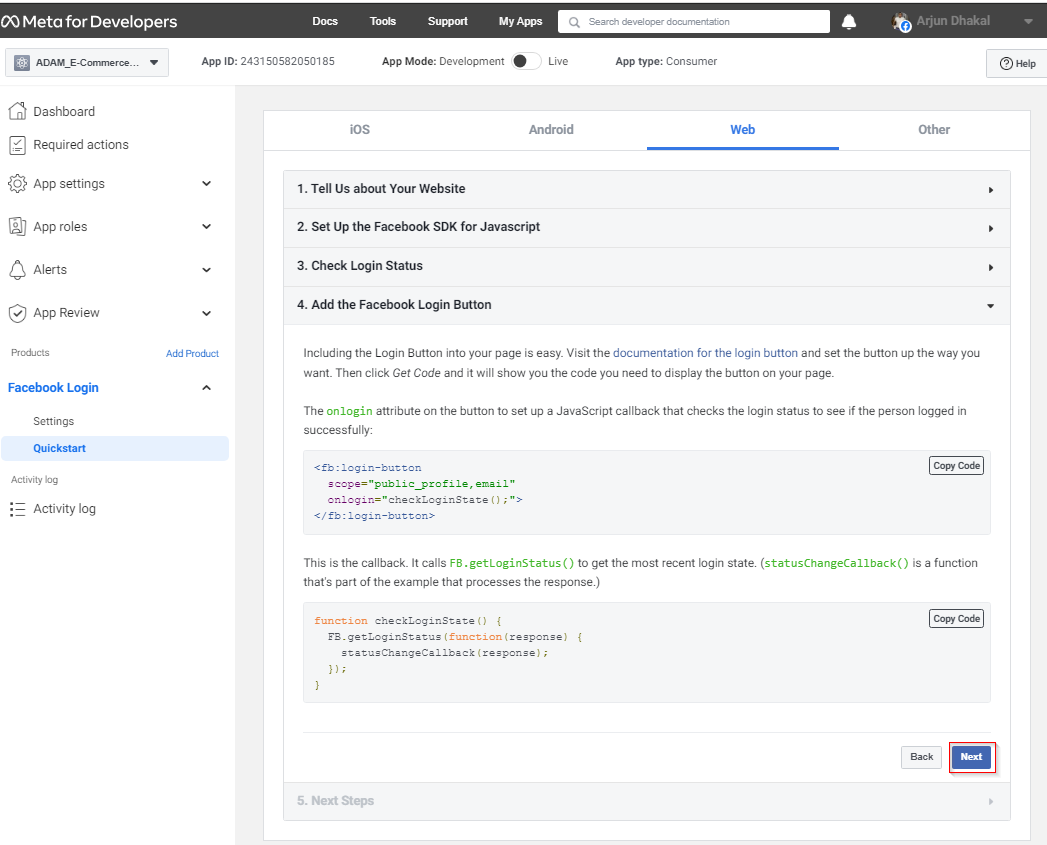


For "Set Up the Facebook SDK for Javascript", we don’t need to change anything for "" just click "Next".

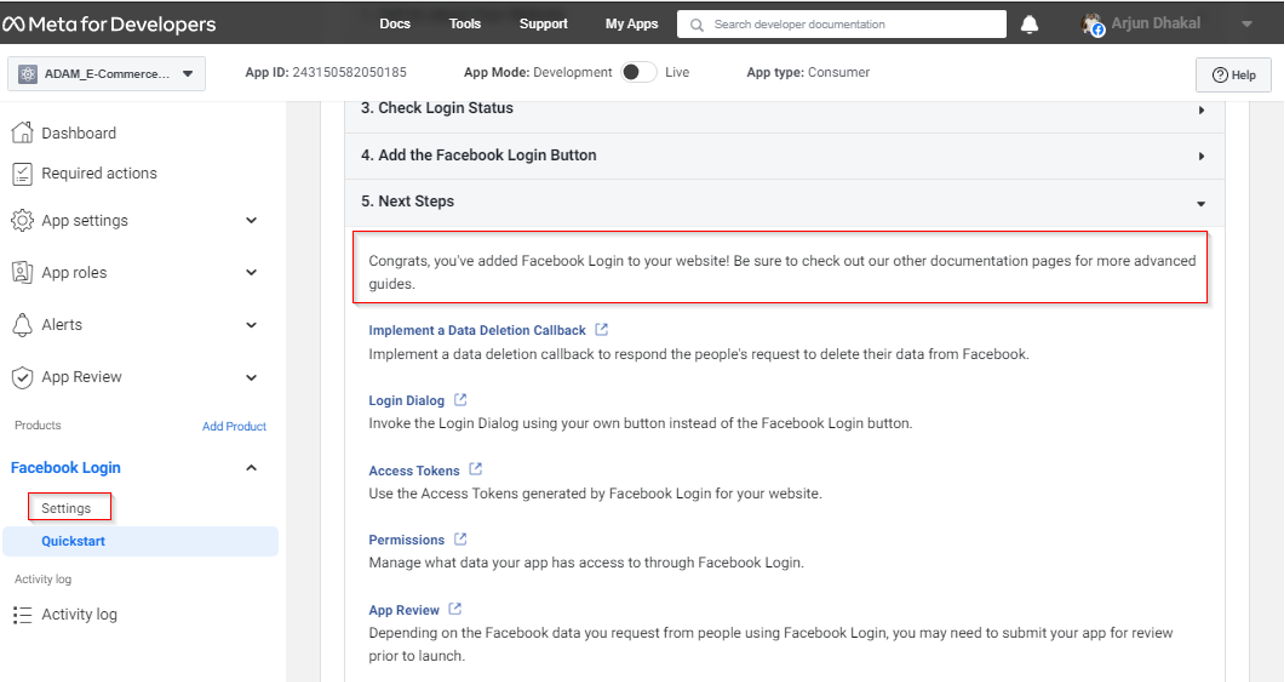
On "Check Login Status", here too we leave the default setting and click "Next".



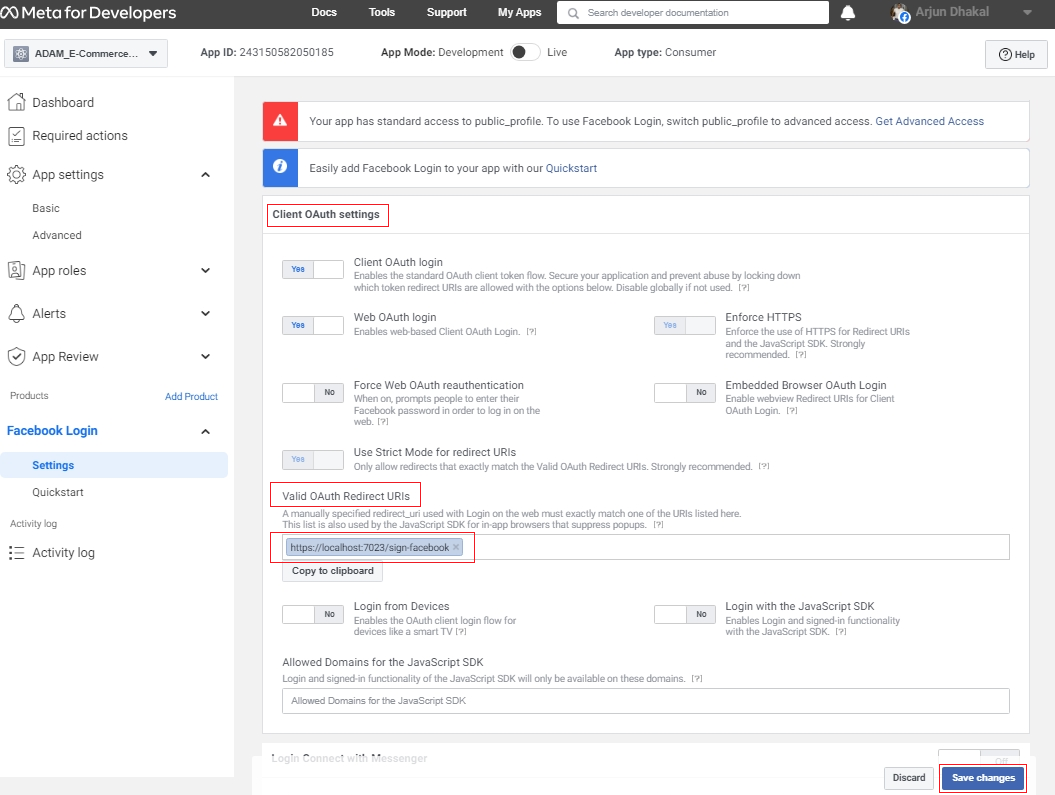
On "Add the Facebook Login Button" too just leave default setting and click "Next".

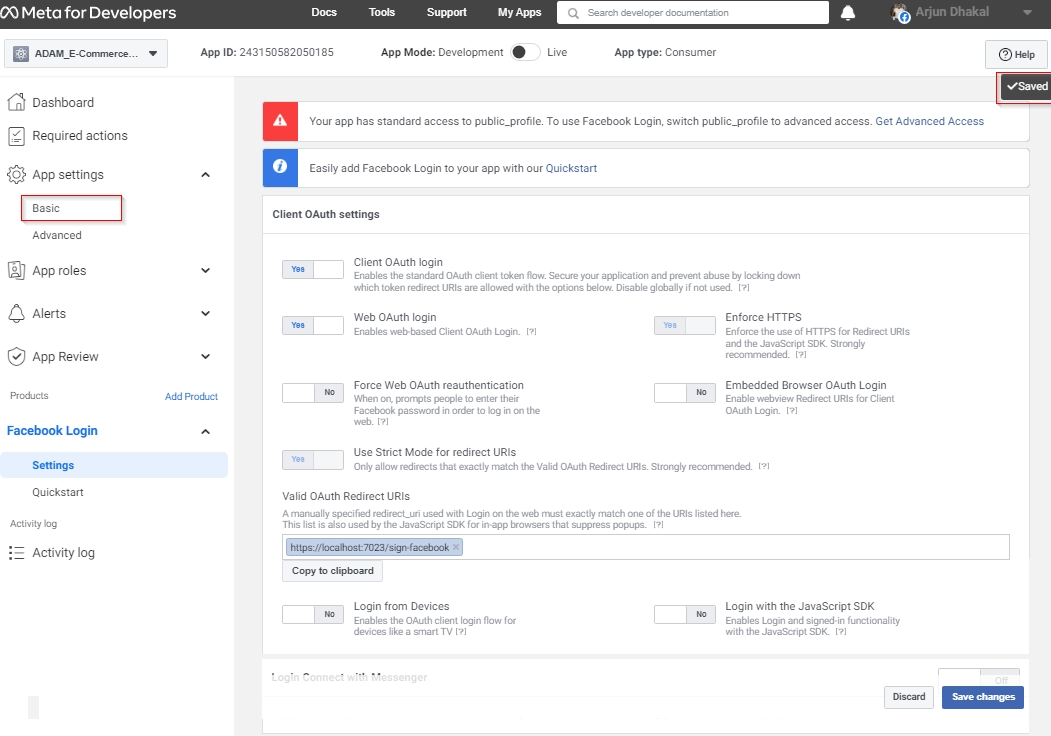


Facebook Login is successfully added, Now select "Settings" under Facebook Login on left navigation bar this will redirect you to "Client OAuth Settings".

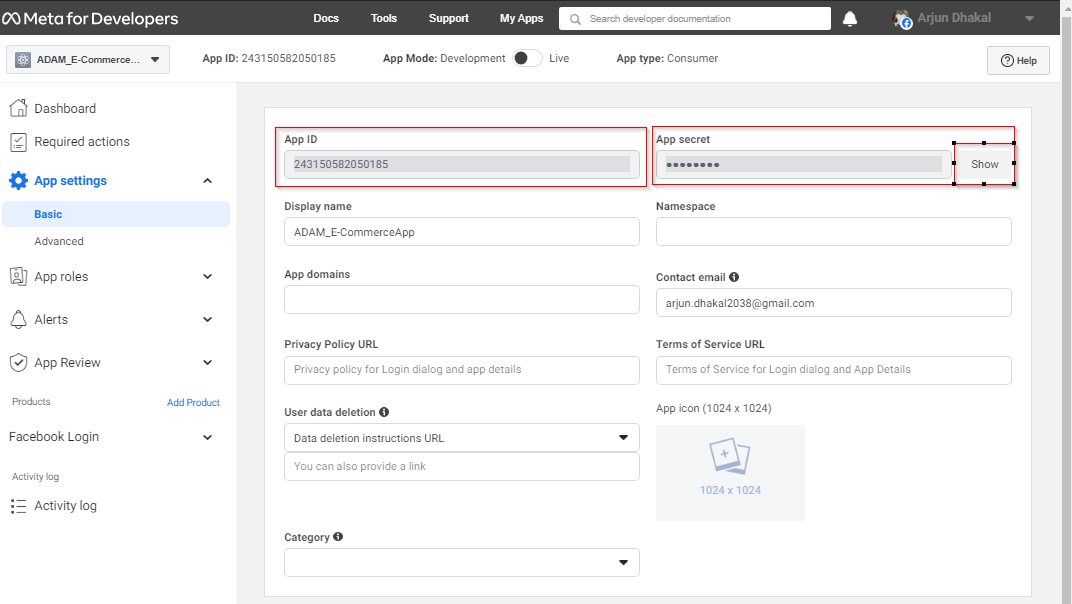


Here we need to provide our valid OAuth Redirect URIs which is basically our URL of our web application followed by "/sign-facebook" . Then click "Save Change".



Once Client OAuth Setting is saved and "Secret Key" is created. To view it go to "App Settings" and then "Basic". 

Here to view, App Secret, click on "Show" button. It may ask for your username and password for Facebook.

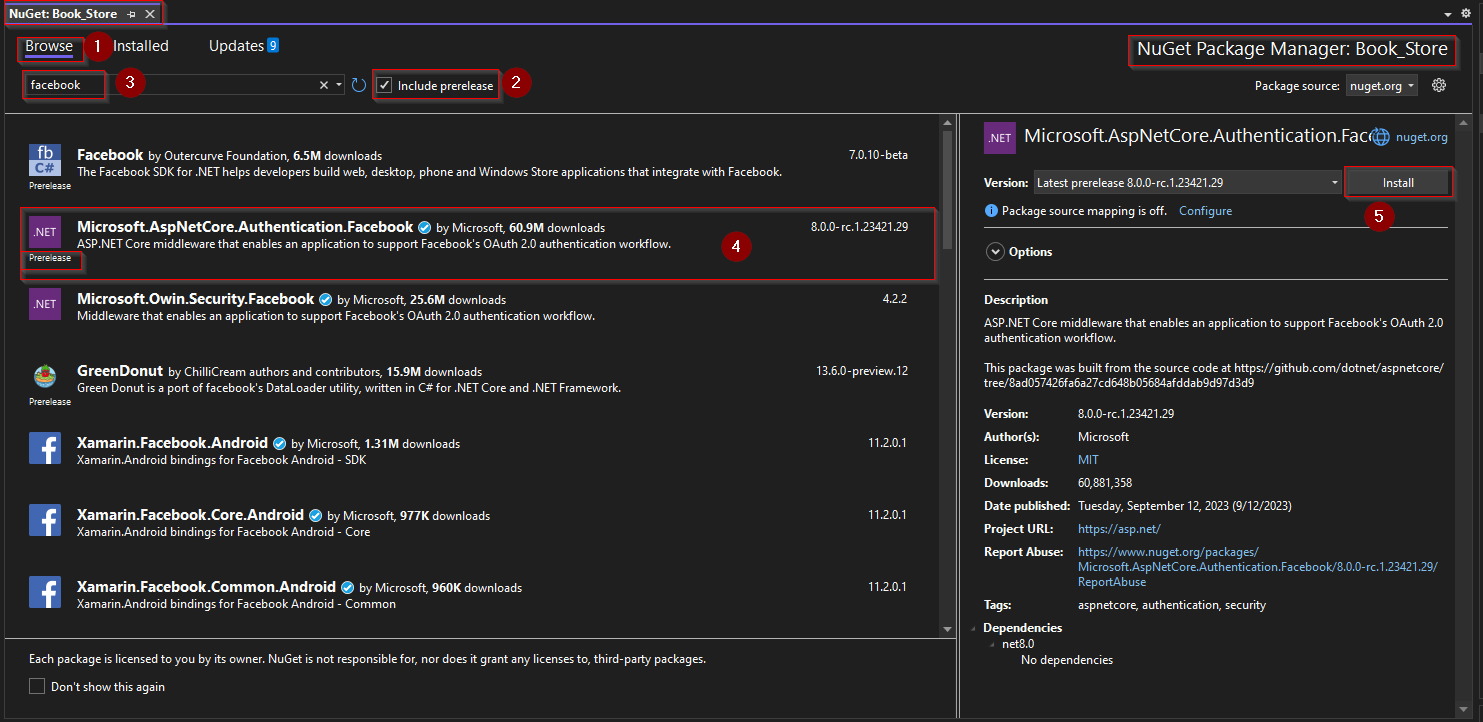


Hence, we have created app, perform necessary setup for Facebook Login and achieve App Id and App Secret Key. Next step to configure Facebook authentication service in our application.

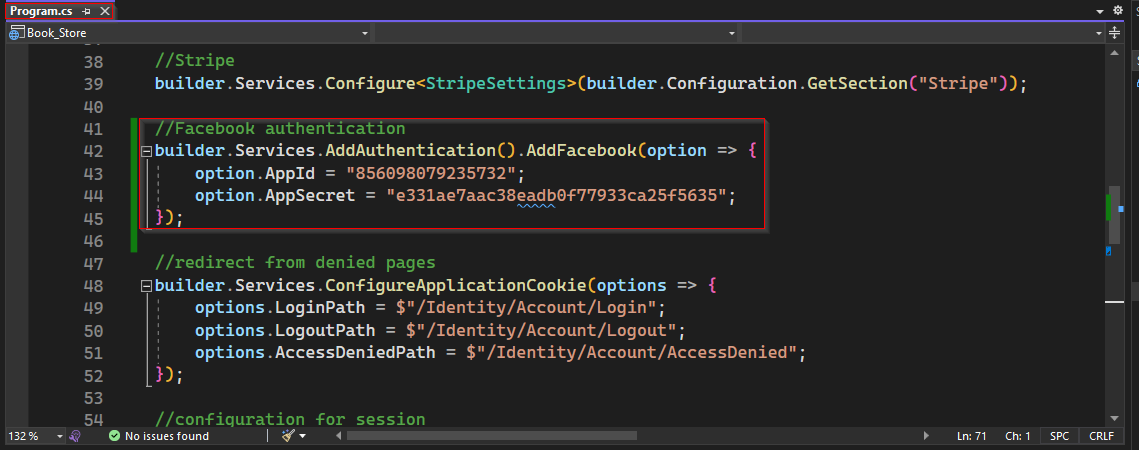
## 13.2.2 Configure Facebook Authentication Service

In old days of .NET Core, Facebook, Google, and all of them were included in the .NET package but now they have separated that and hence we need to add package before configuring it. Now right click on web project and select Manage NuGet Packages then.

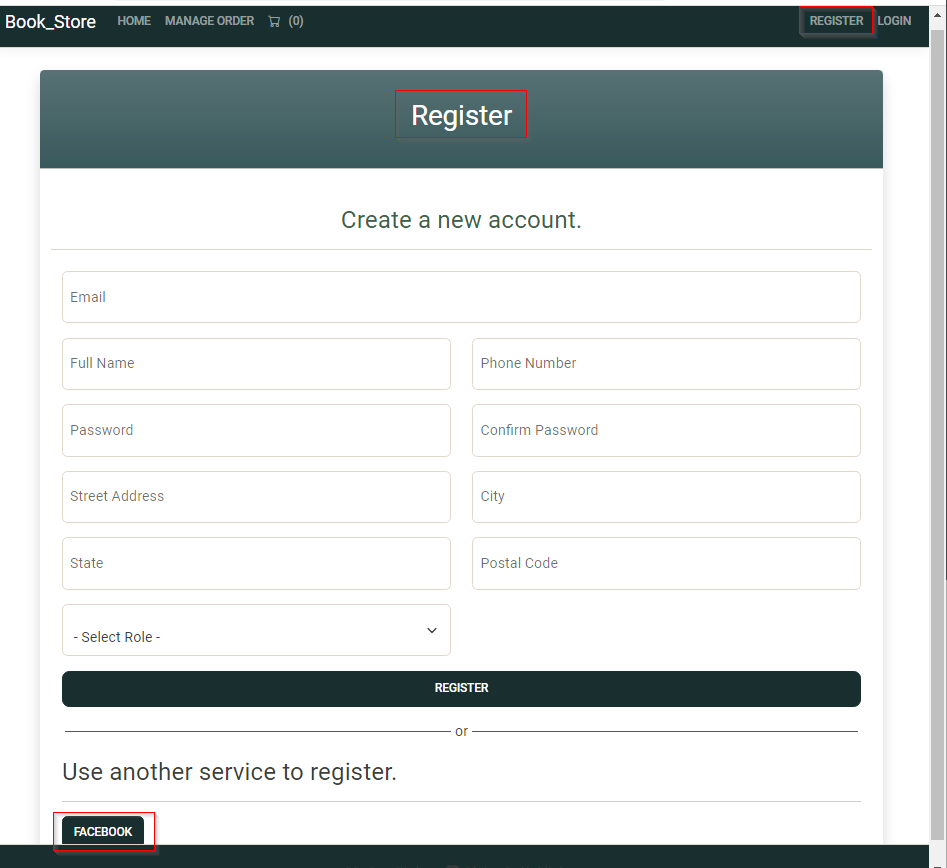
* Select Browse tab.
* Check Include prerelease.
* In search box type some key word like "Facebook" or something similar.
* Select prerelease version of Microsoft.AspNetCore.Authentication.Facebook
* Then click install than click accept and apply for next window that popup during installation.



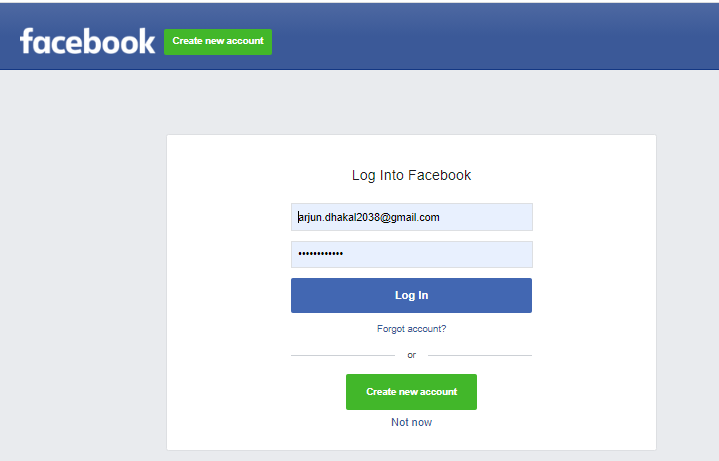
Now we have added the support for Facebook authentication and last thing that we must do is add and configure that in our project using Program.cs as shown below.



After this configuration if we run our application and go to register page, we can see the button for Facebook authentication to register under "Use another service to register" as shown below.



Once you click on "FACEBOOK", if you are not logged-in to your Facebook account on computer you are using it will first redirect to Facebook sign in page as shown below then to ExternalLogin page.

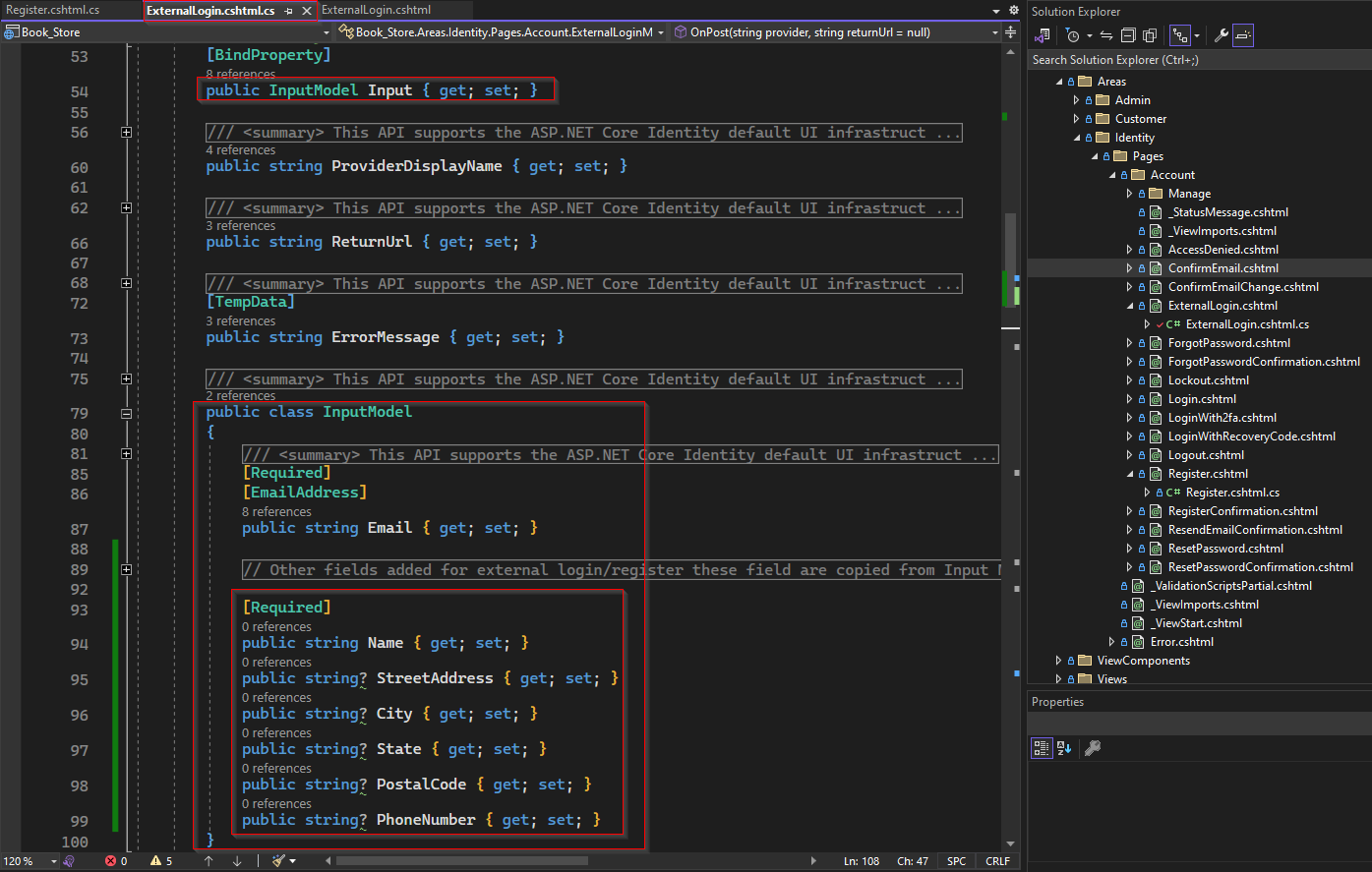


Incase you need to sign-in to your Facebook account, on successful login you will redirect to ExternalLogin page with email field populated as shown below.

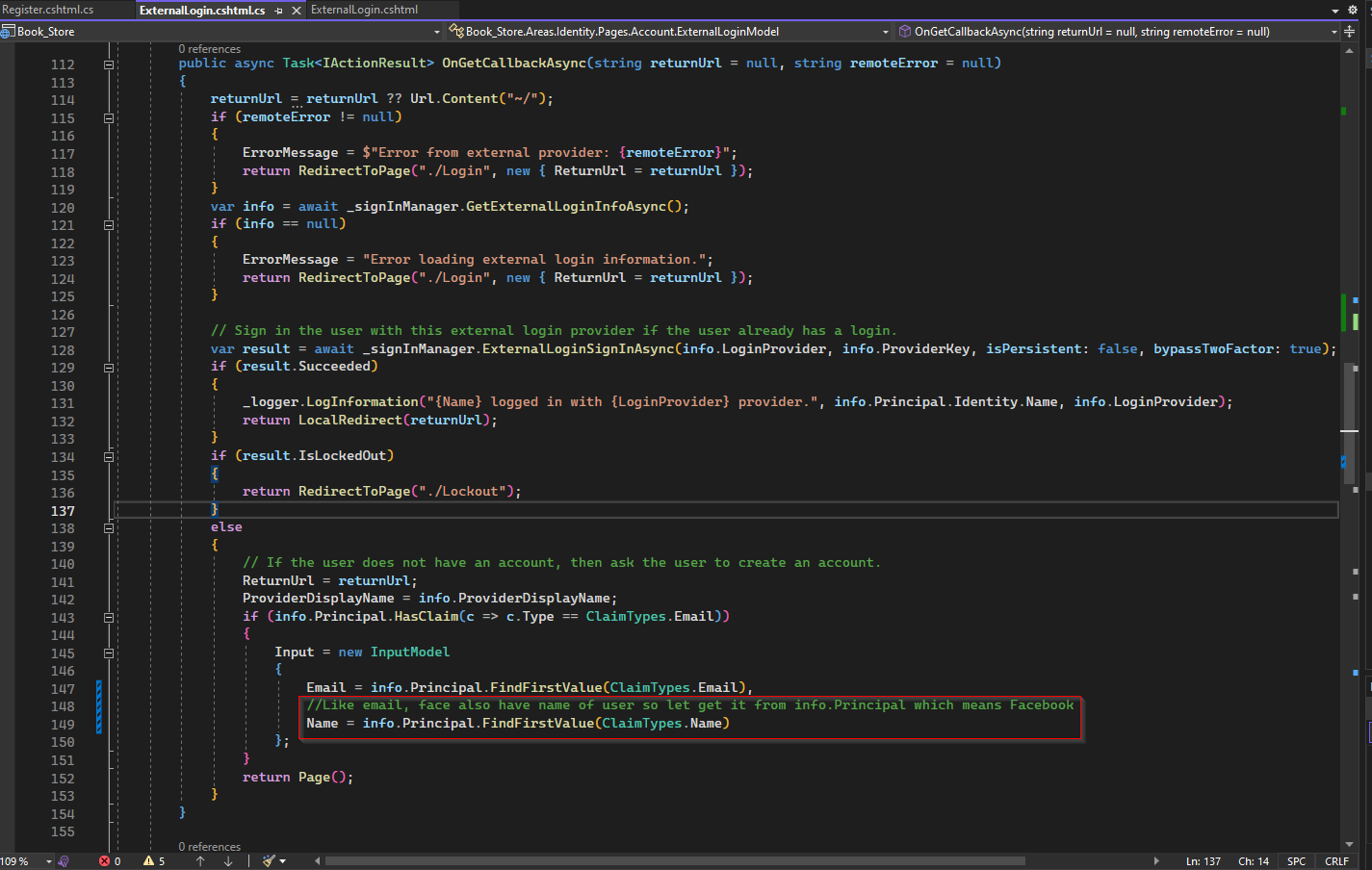


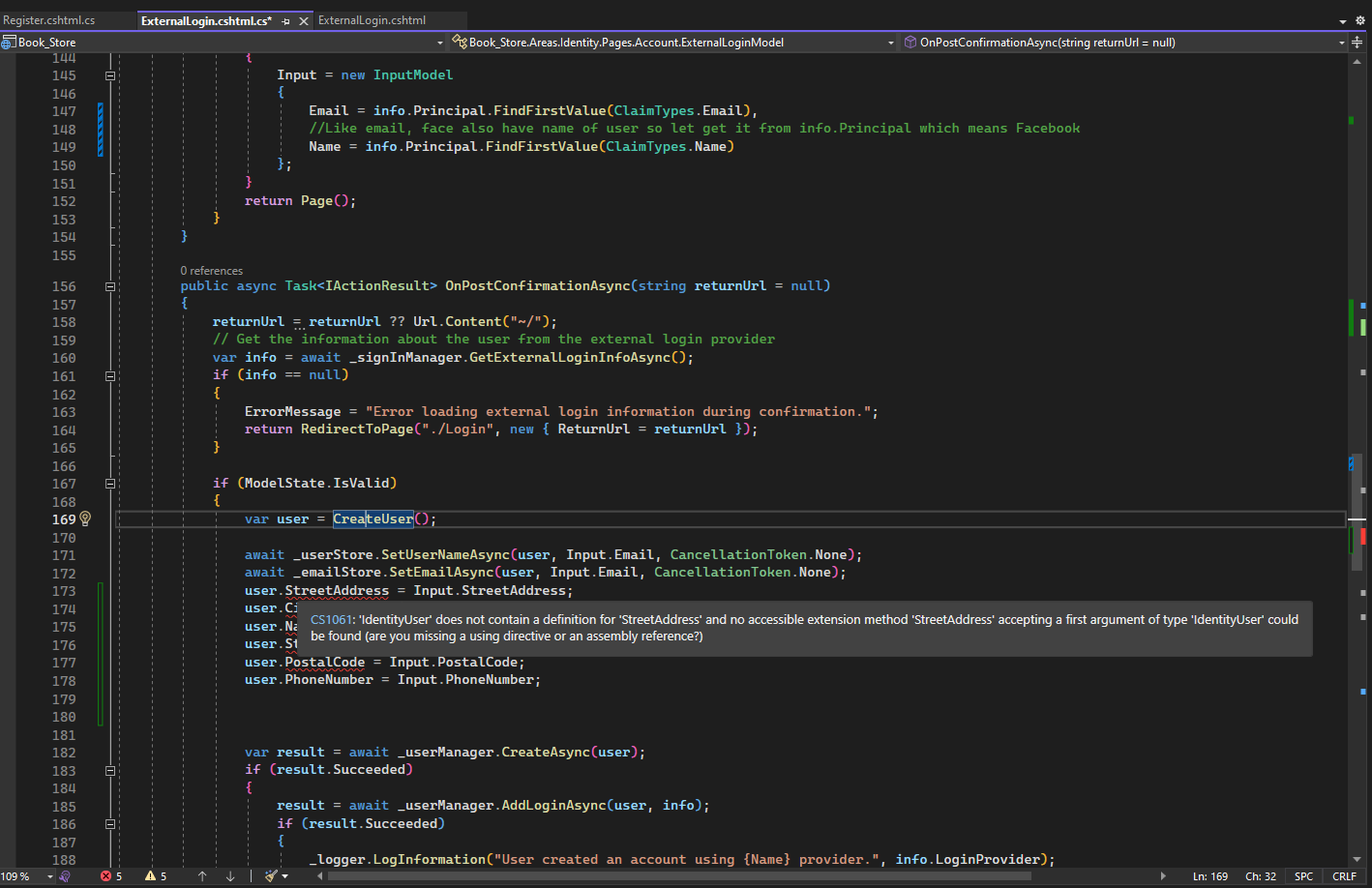
Here Register Form has only email field, but application register page has more field like name, phone number and address details. Let add those missing field in next step.

### 13.2.3 Add More Fields to External Login

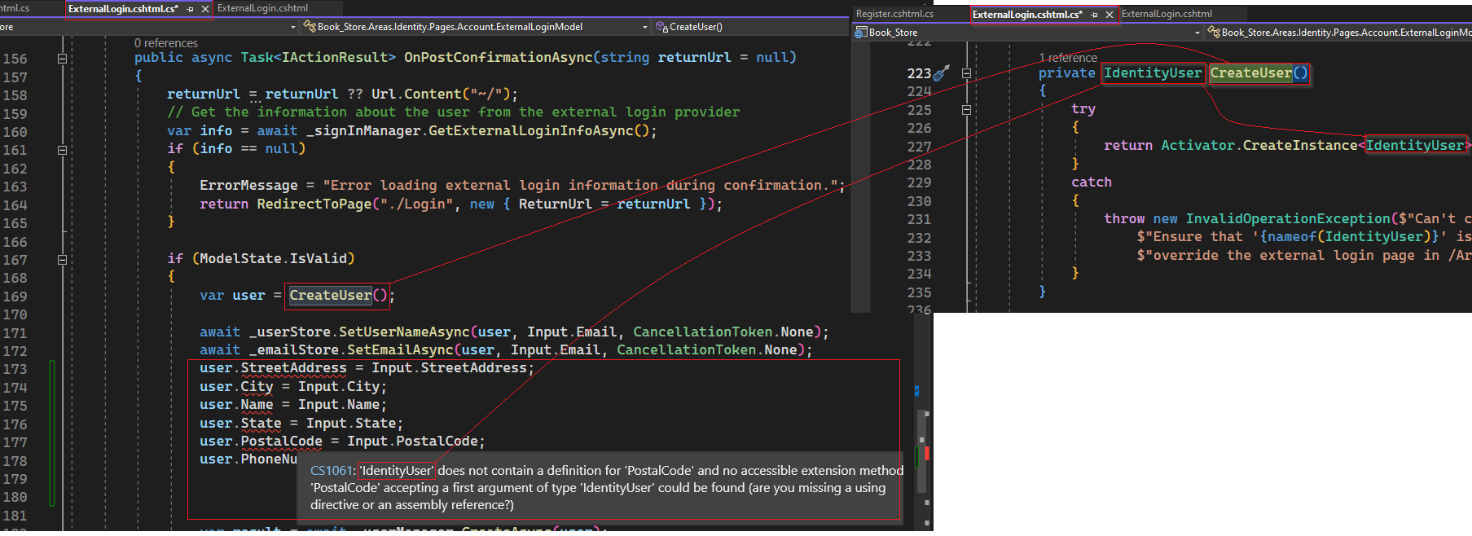


Here we have get callback async and it is populating input model and it is getting email from info.principle, which will be from Facebook. Similar to that, Facebook will also have name of the user, so we can get the claim types of name and will populate name of the user when we were on the registration page.

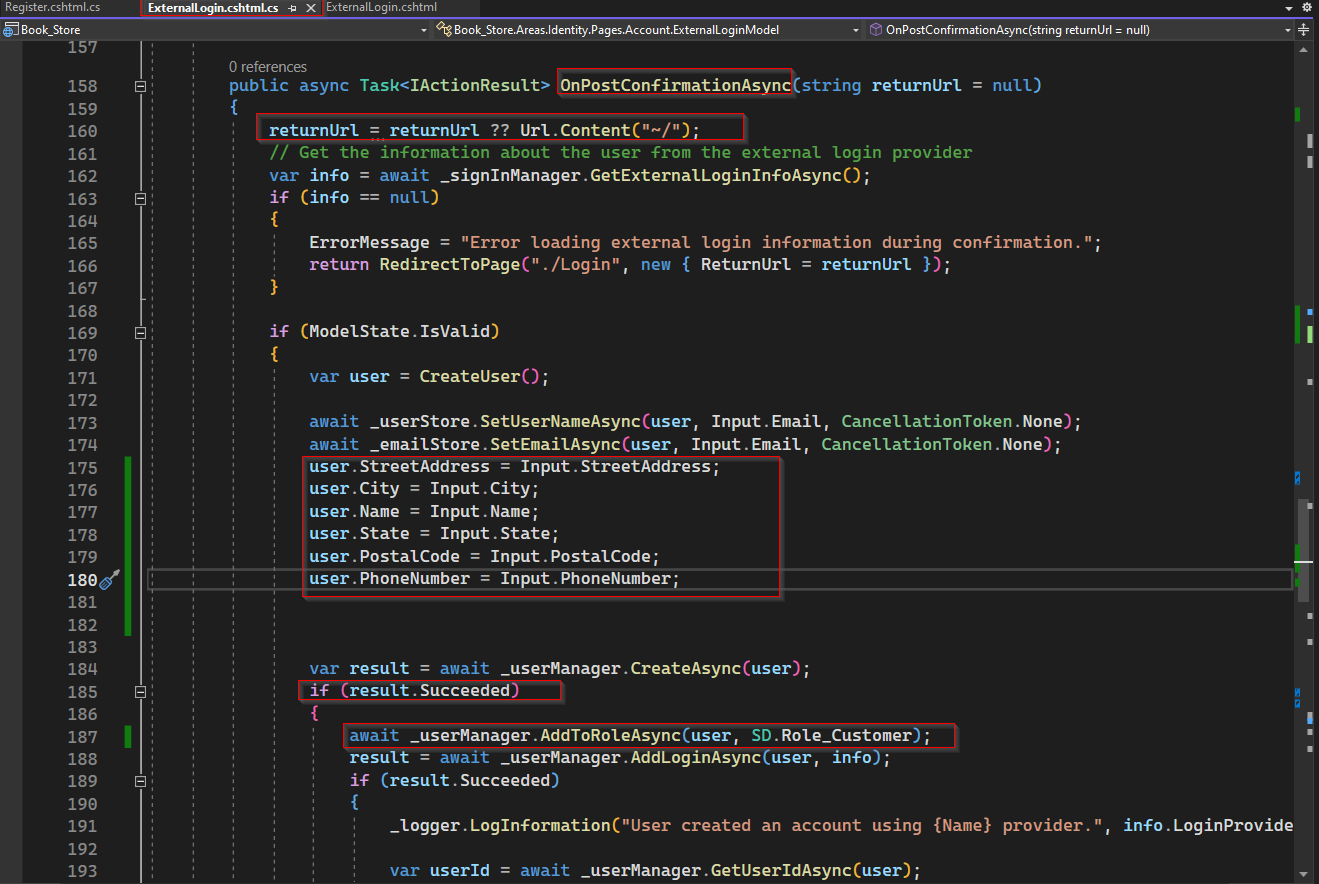




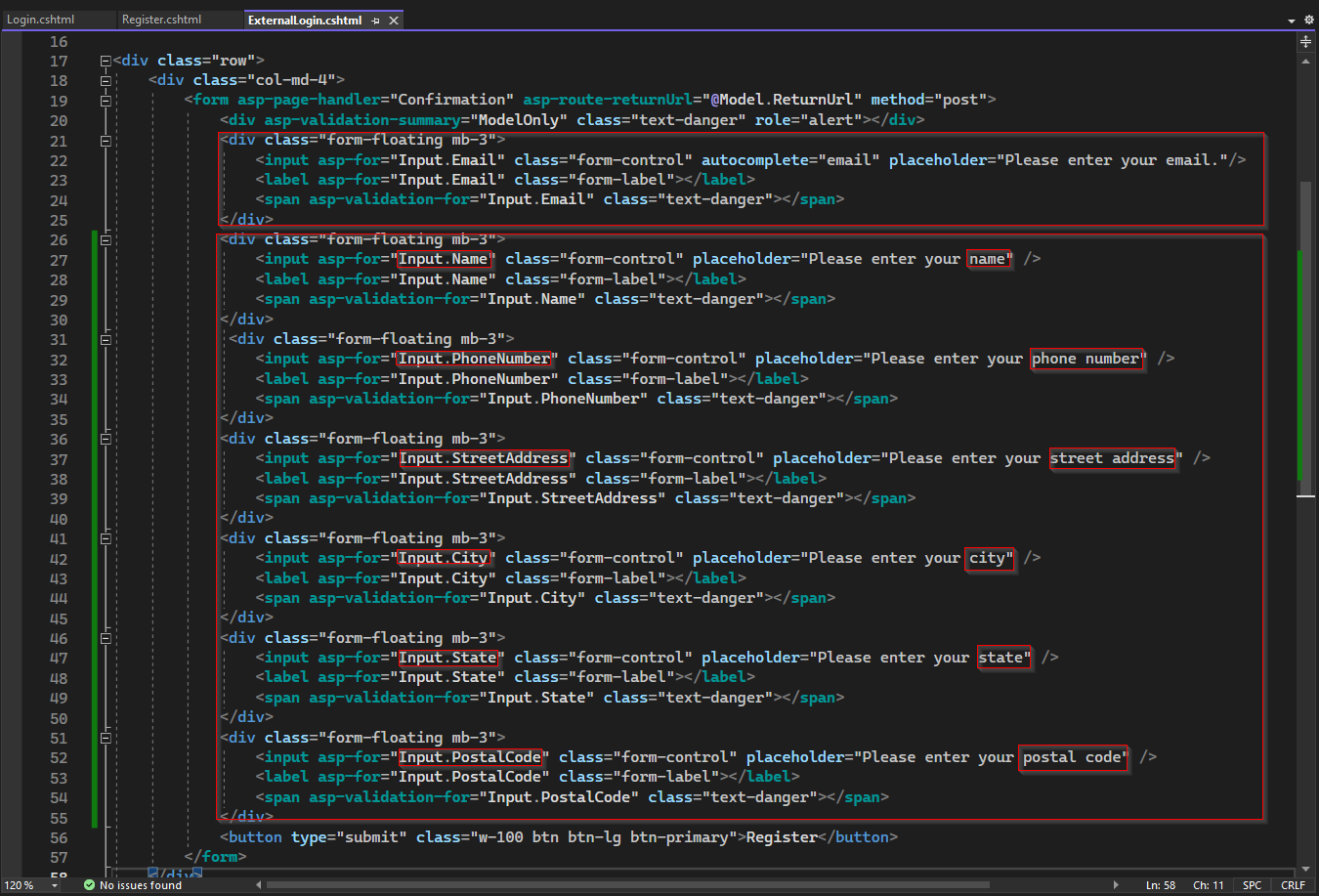
After that on action method OnPostConfirmationAsync, on line 171, we are setting are setting the input.email and input.Name. If we add other properties, but if we try that, they do not exist on identity user. To reason of problem is that return type of CreateUser is IdentityUser so we need to modify it. For put cursor on Crate user and press F12, it will bring you to CreateUser method. On replacing IndentityUser with ApplicationUser the error will go away.



Next change on OnPostConfirmationAsync is after result.success, we can assing role to new user as customer as shown in line 187 below.



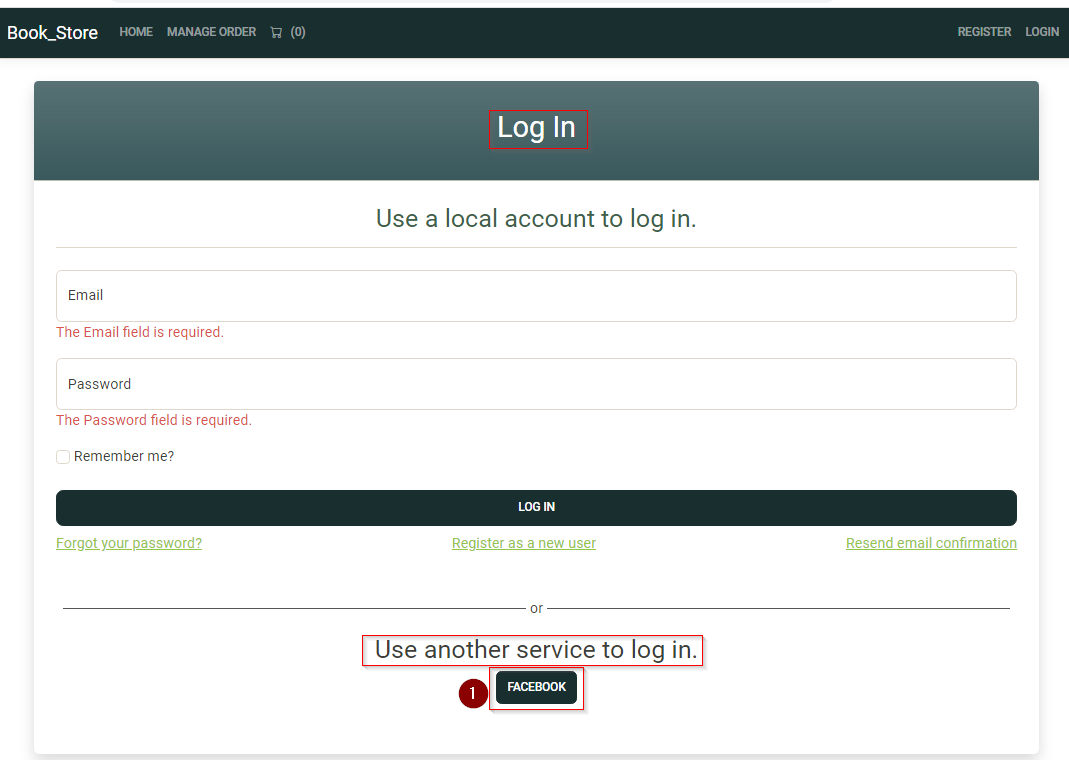
Next, we must add more field in ExternalLogin.cshtml. For this, copy div for email, paste 6 times and modify for other fields as shown below.



Now the Register page will have more fields with email and name pre-populated.



Once you provide value to empty field and hit Register, it will register you as a new user. Next time you can also sign in using Facebook button as shown below and you don’t need to provide email and password.

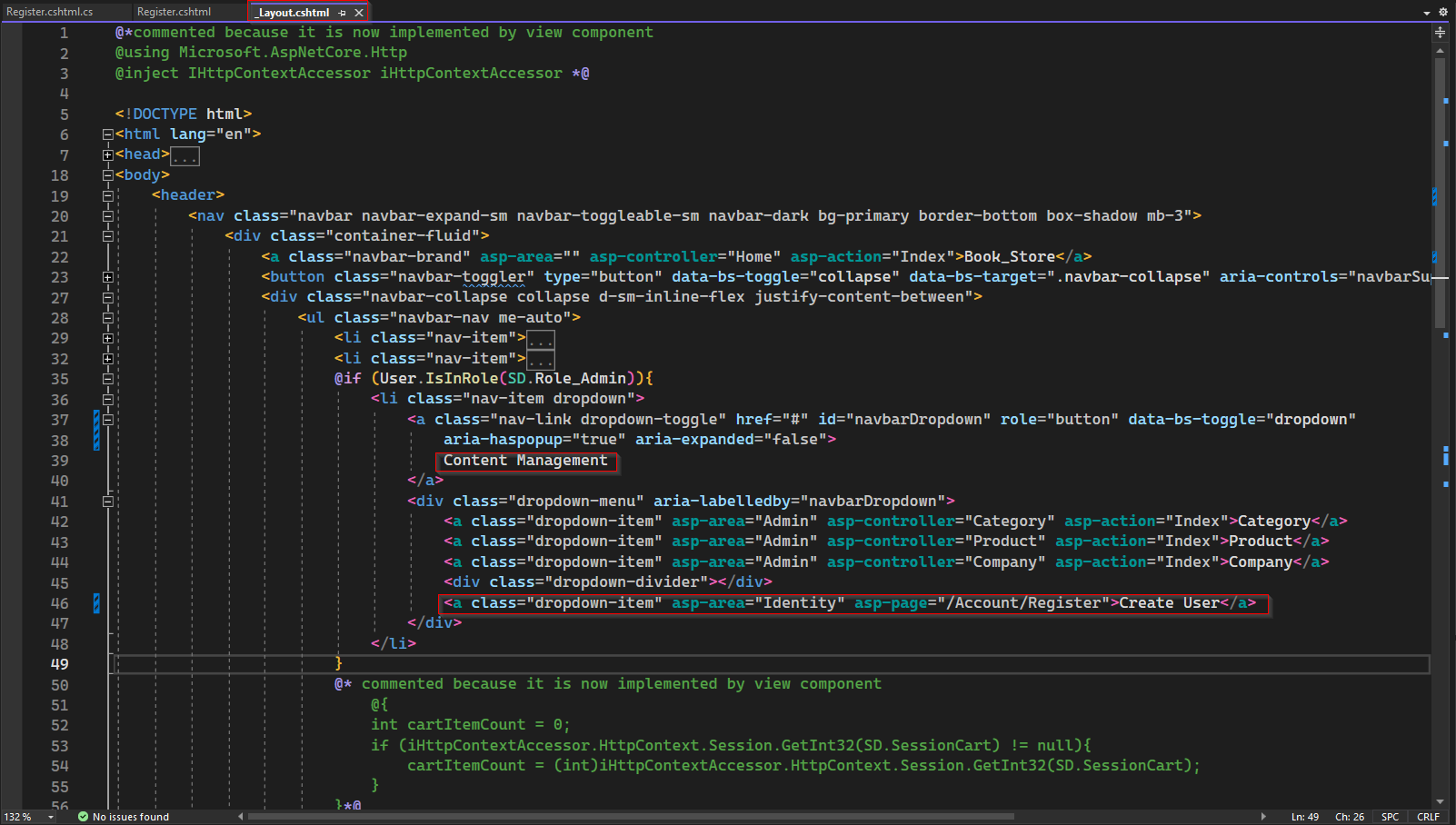


## 3.3 Creating Admin and Employee Accounts

Now, let's say our application is done and we are publishing it online. There, on signup or the register page, we do not want anyone to select a role because admin, employee cannot be created by anyone that browses the website. Also, based on the requirement, a company account can only be created or registered by admin only.

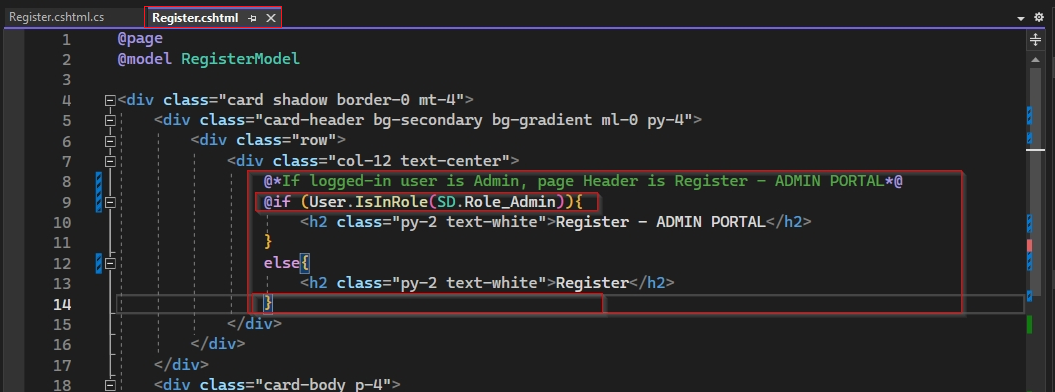
So, to have that functionality

* we need to add navigation of register page inside content management which will be access by admin only.
* We need to modify register page and its page model adding few if statements to check if currently logged-in user is admin or not.

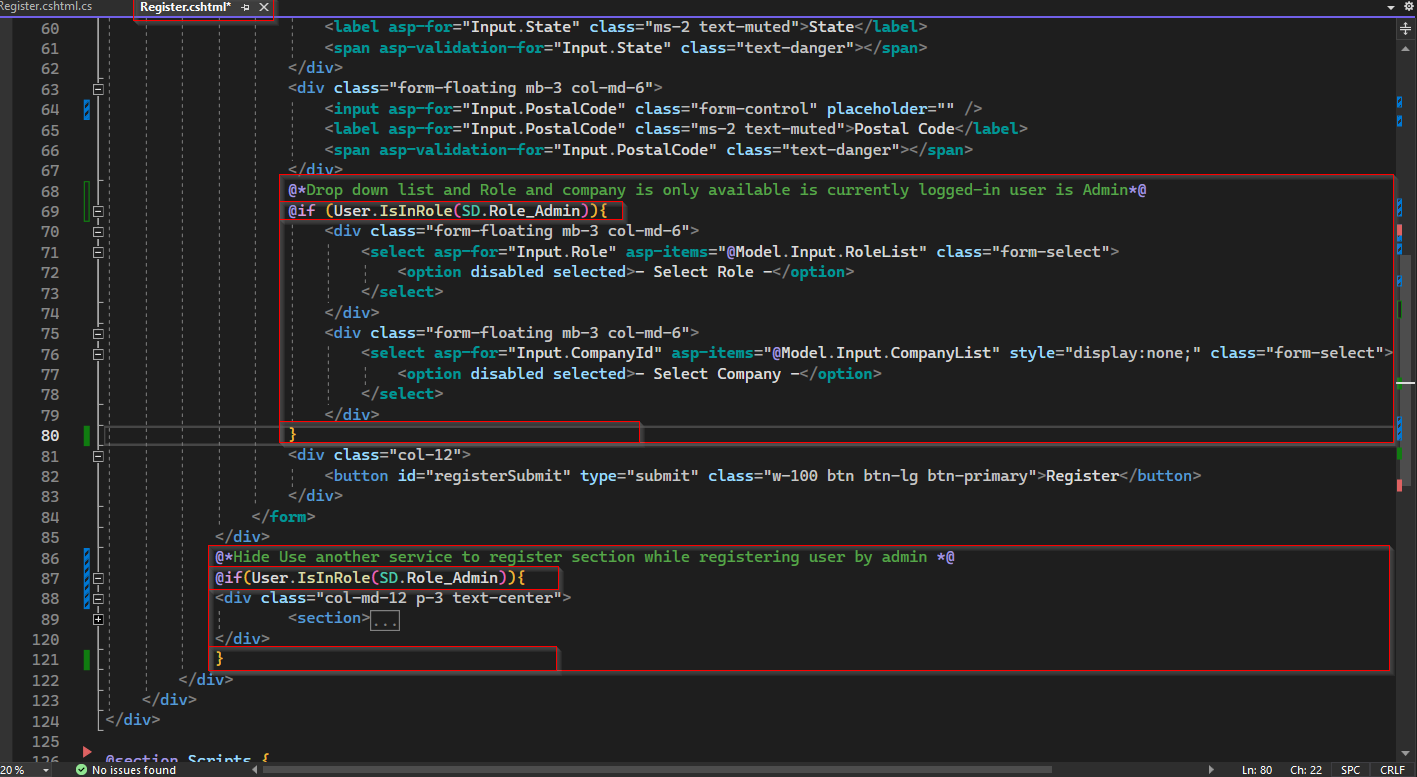


Here we have added a link to register page under Content Management which is only access by Admin. Note since Register.cshtml is a Razor page we don’t have controller and action method.

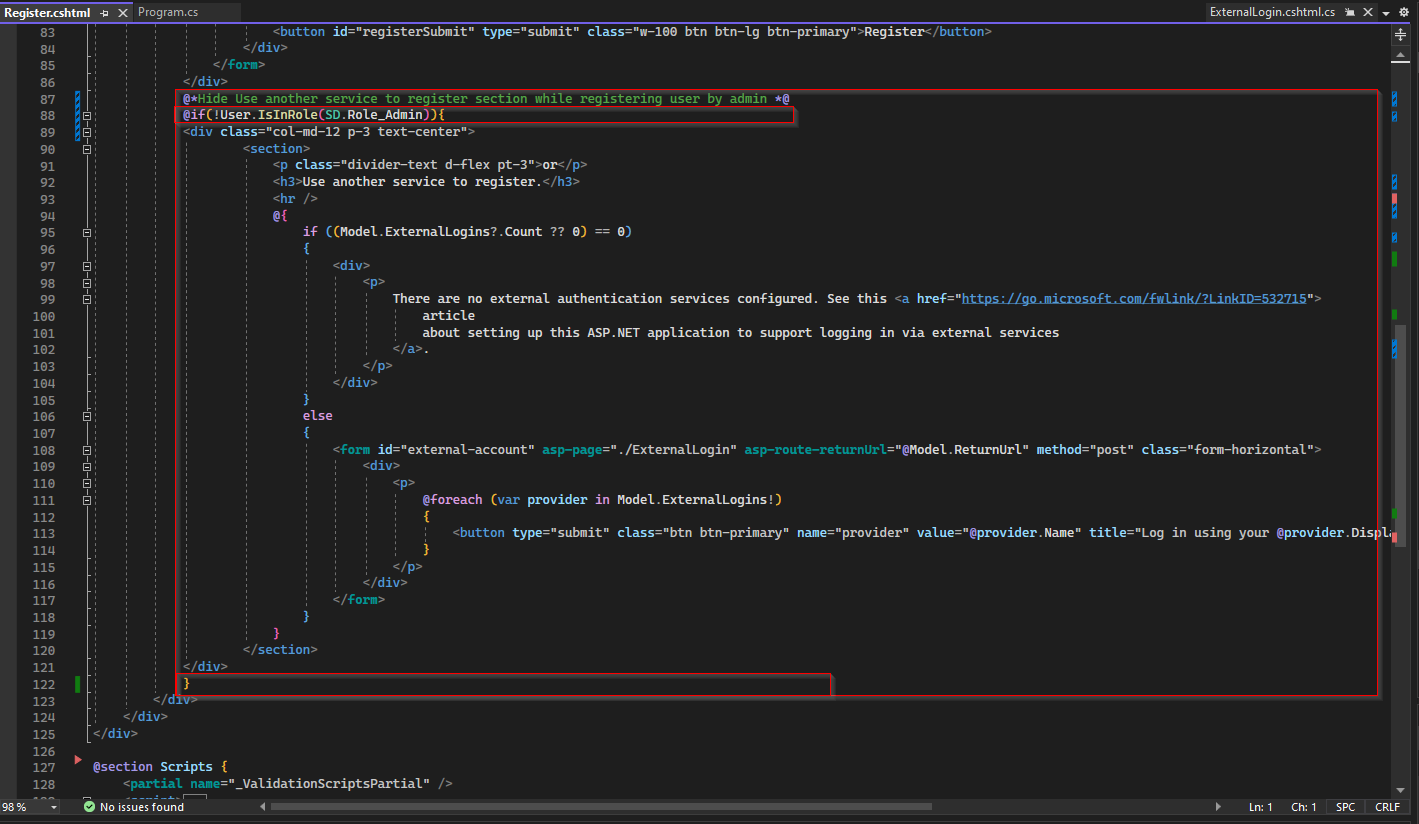
First modification to Register page, if admin is using the page to create new user, let change the header of page to be "Register – ADMIN PORTAL" else "Register".



2nd modification: if logged in user has Admin role, then dropdown list to select role and company will be available else not.



3rd modification: if admin using register page, then it is not necessary to view the section to register using a external authentication like Facebook.



4th modification: This modification is on Register.cshtml.cs page model, when admin register a new user it should display success message instead automatically login and redirect to home for new user.

