Using Identity In ASP.NET Core MVC Authentication

Ramasagar Pulidindi Updated date Aug 12, 2020

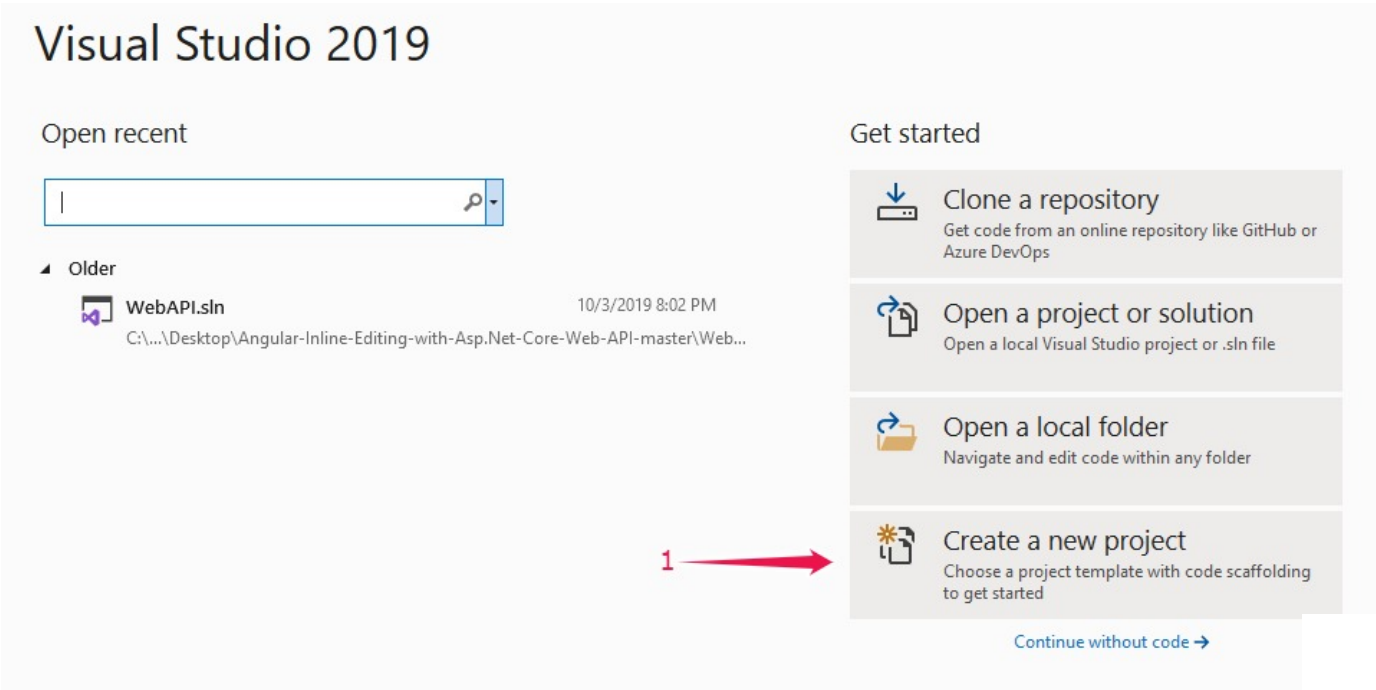
Introduction

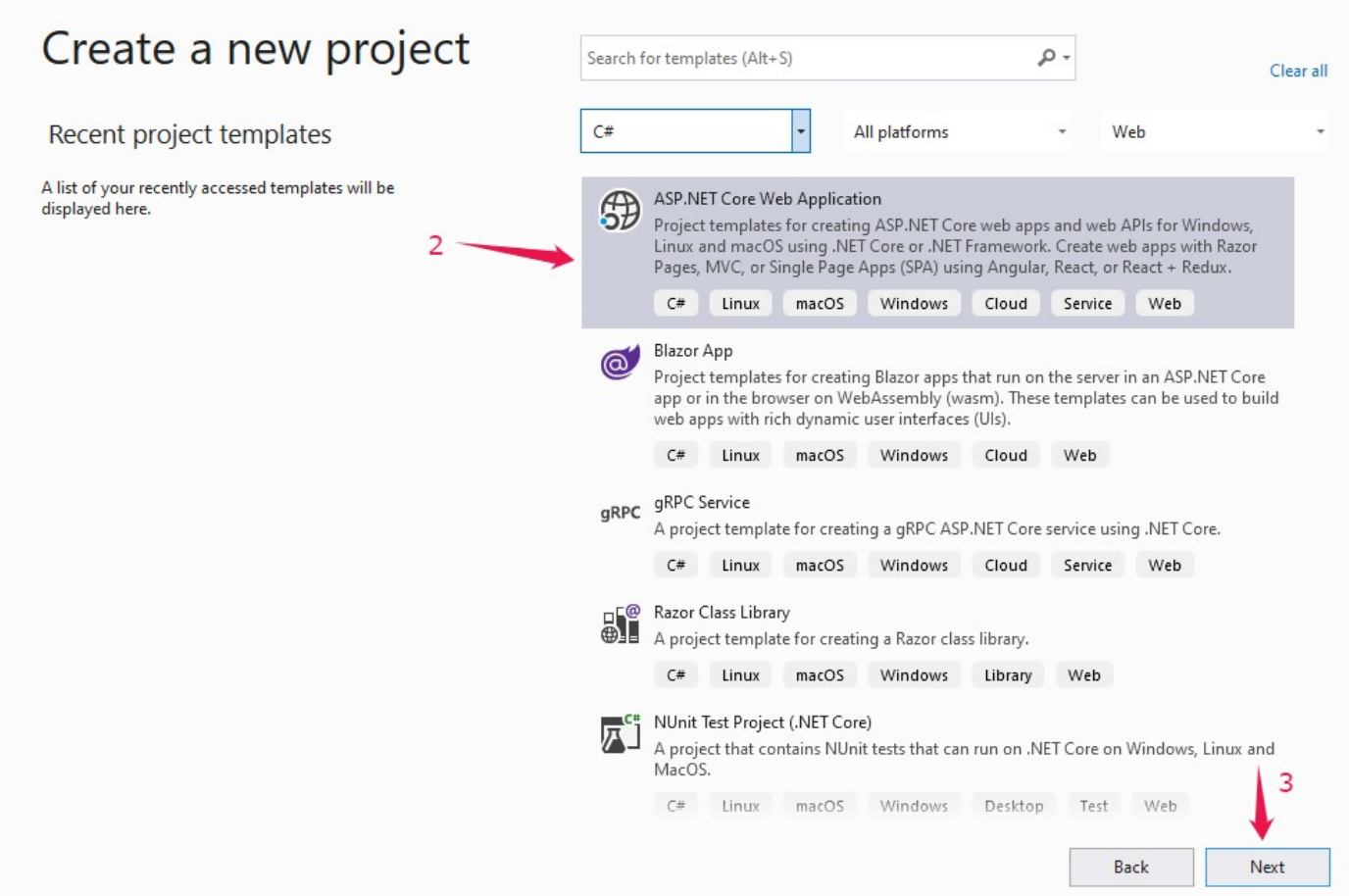
This article explains how to get started with Identity.UI in ASP.Net Core MVC user authentication and registration.

Link to download the project source code here.

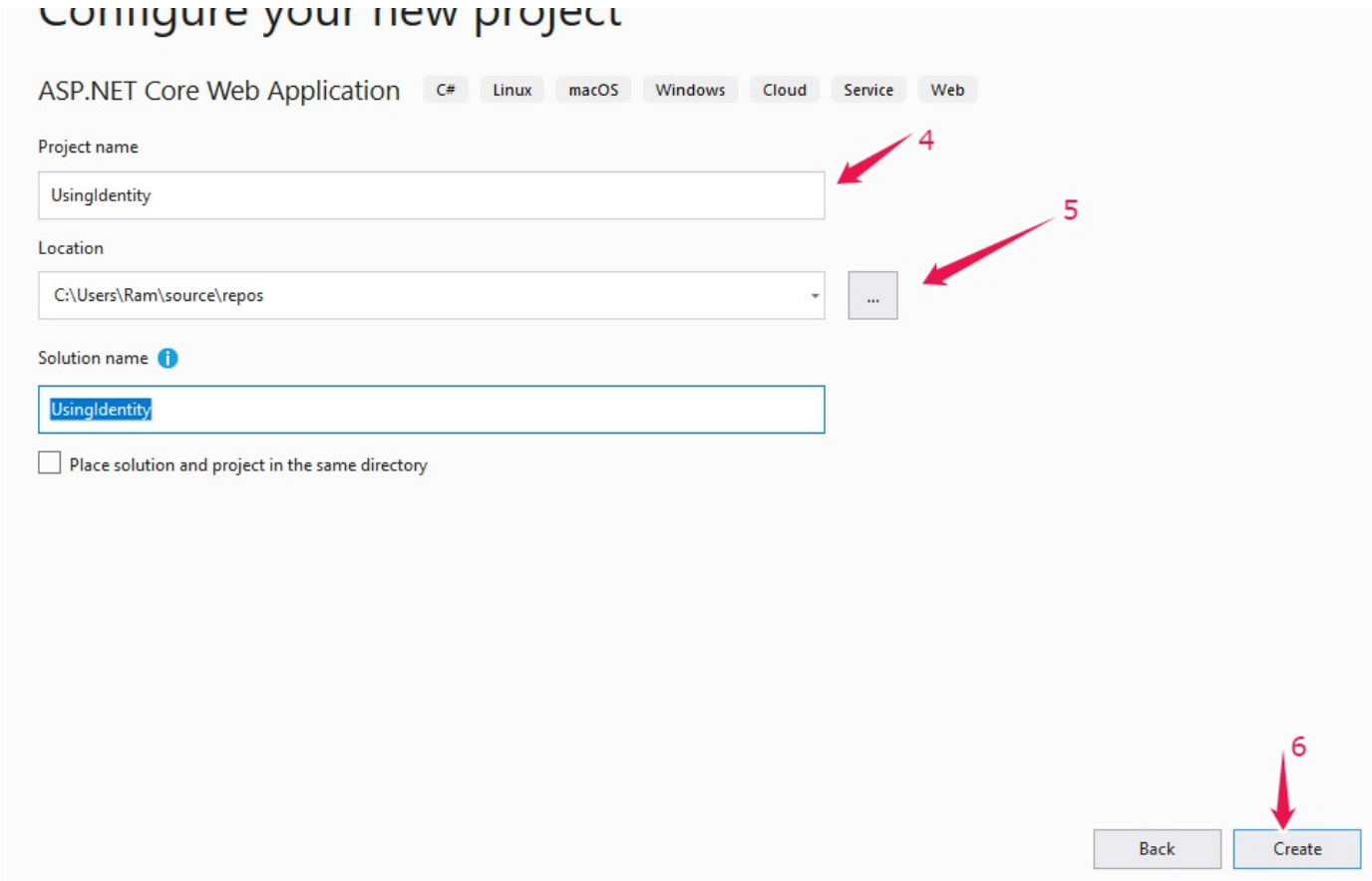
Step 1

Let's create an ASP.NET Core web application. Open Visual Studio 2019 and click on Create a new project.

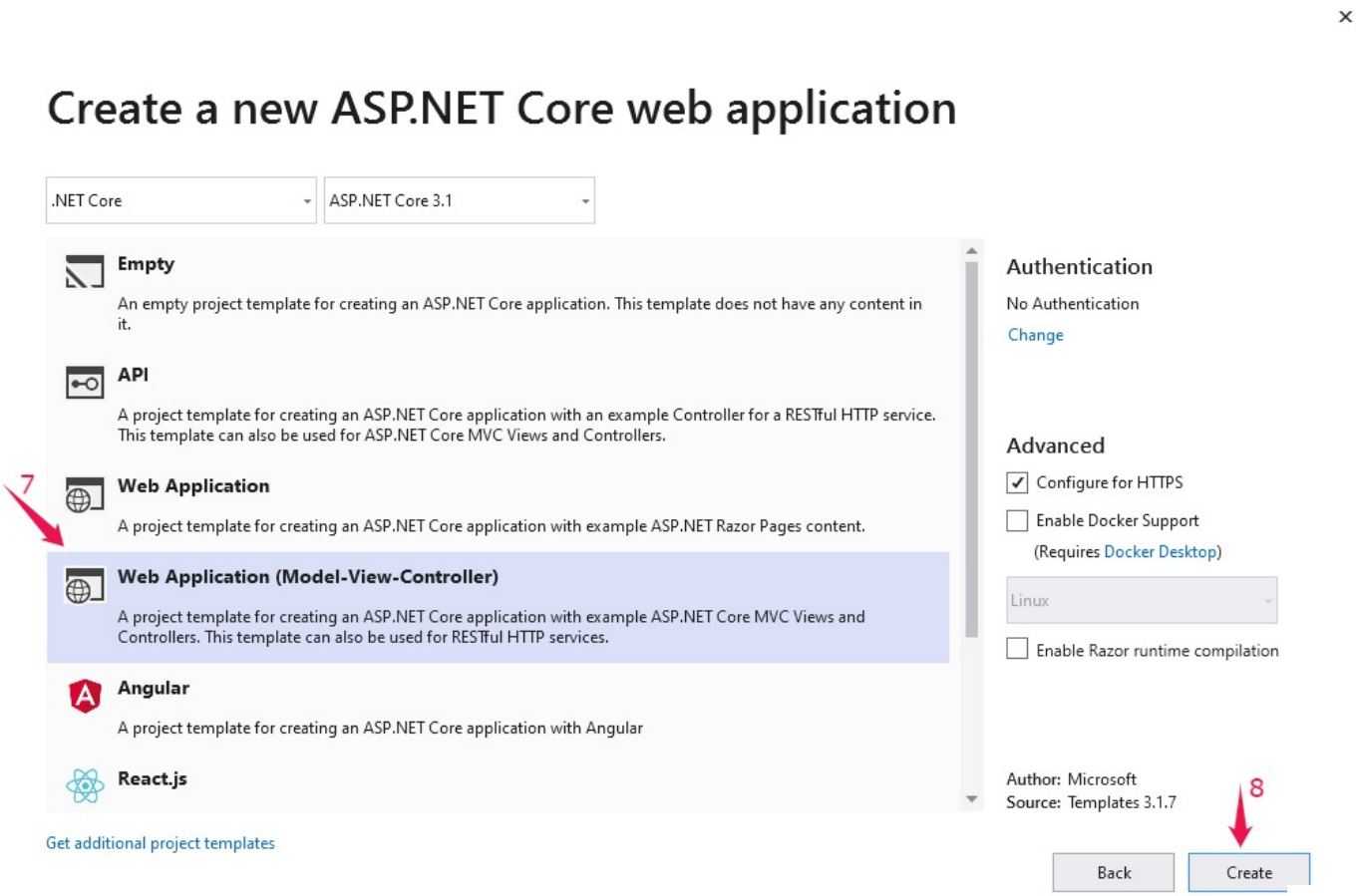




Provide the project name of your choice, select the preferred location & click on Create

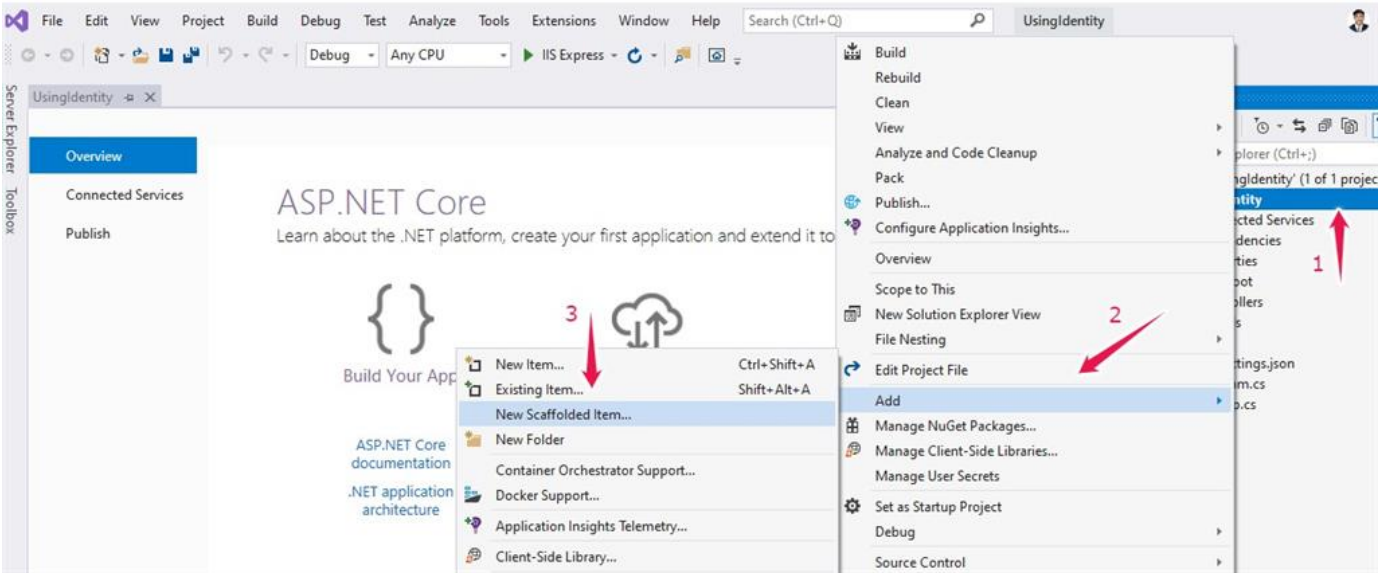


Select MVC Template and click on create, as shown below:



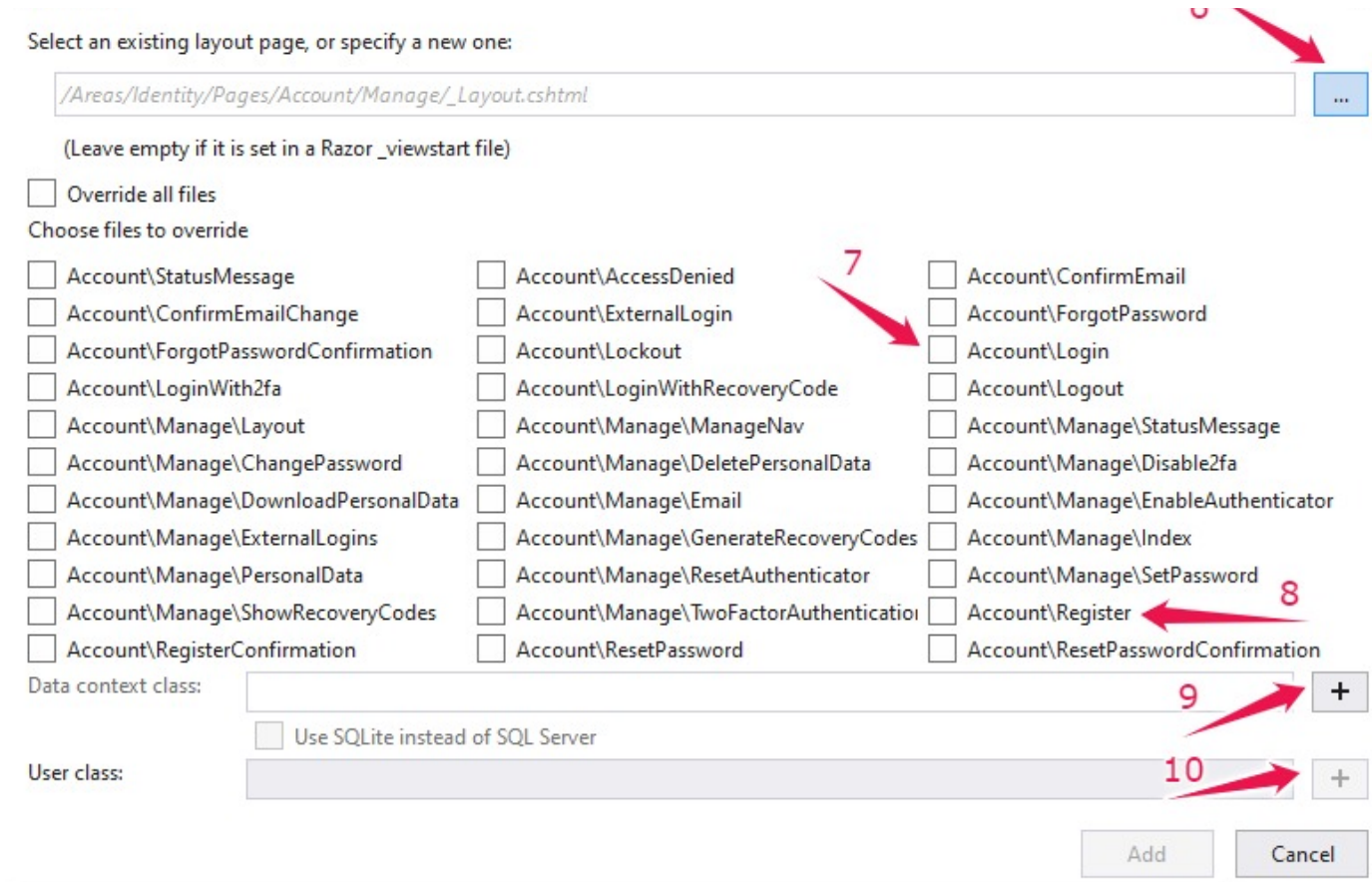
Step 2

Scaffold item

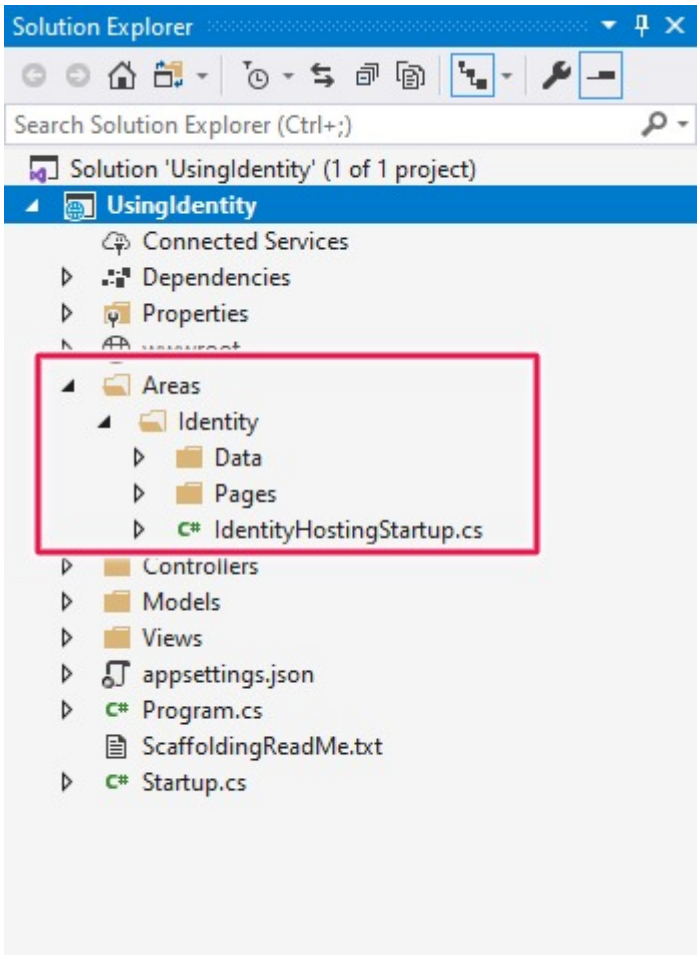


Select Identity and click on add

Now select the layout page, as we want authentication. Let's select login and Register as shown below and provide DbContext class and user class and click on Add



We can find the Areas in the application with Data & Razor pages, as shown below:



-

Now let's add user authentication to the application. Open the Startup class and Modify as shown below.

In order to add support to the razor pages, we have to call the function services.AddRazorPages() and endpoints.MapRazorPages()

```
01. public class Startup
02.    {
03.        public Startup(IConfiguration configuration)
04.        {
05.            Configuration = configuration;
06.        }
07.
08.        public IConfiguration Configuration { get; }
09.
10.        // This method gets called by the runtime. Use this method to ad
11.        public void ConfigureServices(IServiceCollection services)
12.        {
13.            services.AddControllersWithViews();
14.            services.AddRazorPages();
15.        }
16.
17.        // This method gets called by the runtime. Use this method to co
18.        public void Configure(IApplicationBuilder app, IWebHostEnvironme
19.        {
20.            if (env.IsDevelopment())
21.            {
22.                app.UseDeveloperExceptionPage();
23.            }
24.            else
25.            {
26.                app.UseExceptionHandler("/Home/Error");
27.                // The default HSTS value is 30 days. You may want to ch
    hsts.
28.                app.UseHsts();
29.            }
30.            app.UseHttpsRedirection();
31.            app.UseStaticFiles();
32.
33.            app.UseRouting();
34.            app.UseAuthentication();
35.
36.            app.UseAuthorization();
37.
38.            app.UseEndpoints(endpoints =>
39.            {
40.                endpoints.MapControllerRoute(
41.                    name: "default",
42.                    pattern: "
    {controller=Home}/{action=Index}/{id?}");
43.                endpoints.MapRazorPages();
44.            });
45.        }
```

Step 4

Now let's start with creating the database for the application.

Open UsingIdentityUser class and add the properties and decorate with the attribute PersonalData.

```
01. public class UsingIdentityUser : IdentityUser
02.    {
03.        [PersonalData]
04.        [Column(TypeName ="nvarchar(100)")]
05.        public string Firstname { get; set; }
06.        [PersonalData]
07.        [Column(TypeName = "nvarchar(100)")]
08.        public string LastName { get; set; }
09.    }
```

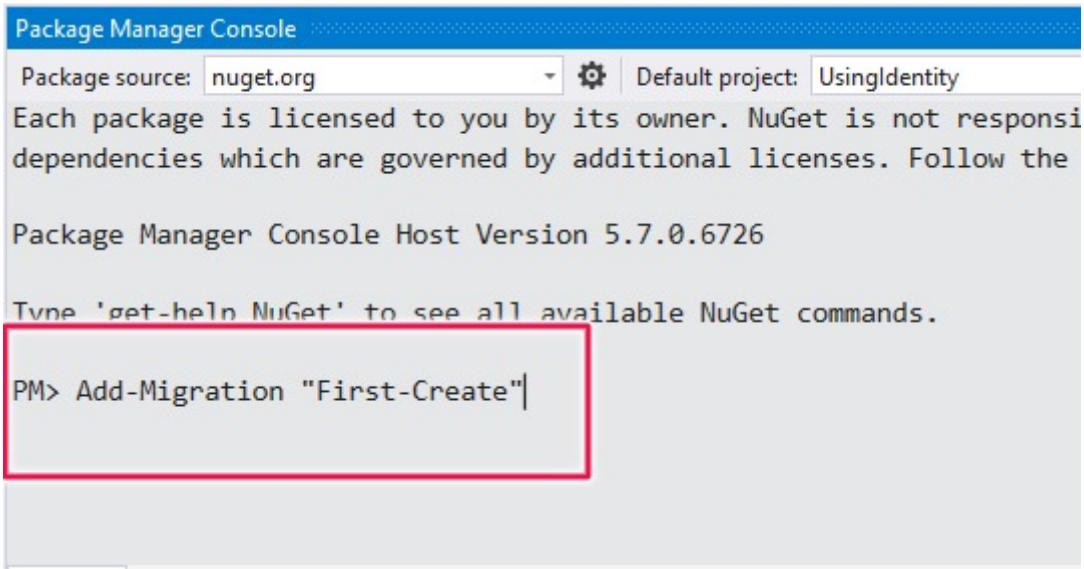
We have Dbcontext, which is also inherited from the parent class IdentityDB context. This identity db context is injected inside this identityhostingstartup class.

```
01. public class IdentityHostingStartup : IHostingStartup
02.    {
03.        public void Configure(IWebHostBuilder builder)
04.        {
05.            builder.ConfigureServices((context, services) => {
06.                services.AddDbContext<UsingIdentityContext>
    (options =>
07.                    options.UseSqlServer(
08.                        context.Configuration.GetConnectionString("Using
09.
10.                services.AddDefaultIdentity<UsingIdentityUser>
    (options => options.SignIn.RequireConfirmedAccount = true)
11.                    .AddEntityFrameworkStores<UsingIdentityContext>
    ();
12.            });
13.        }
14.    }
```

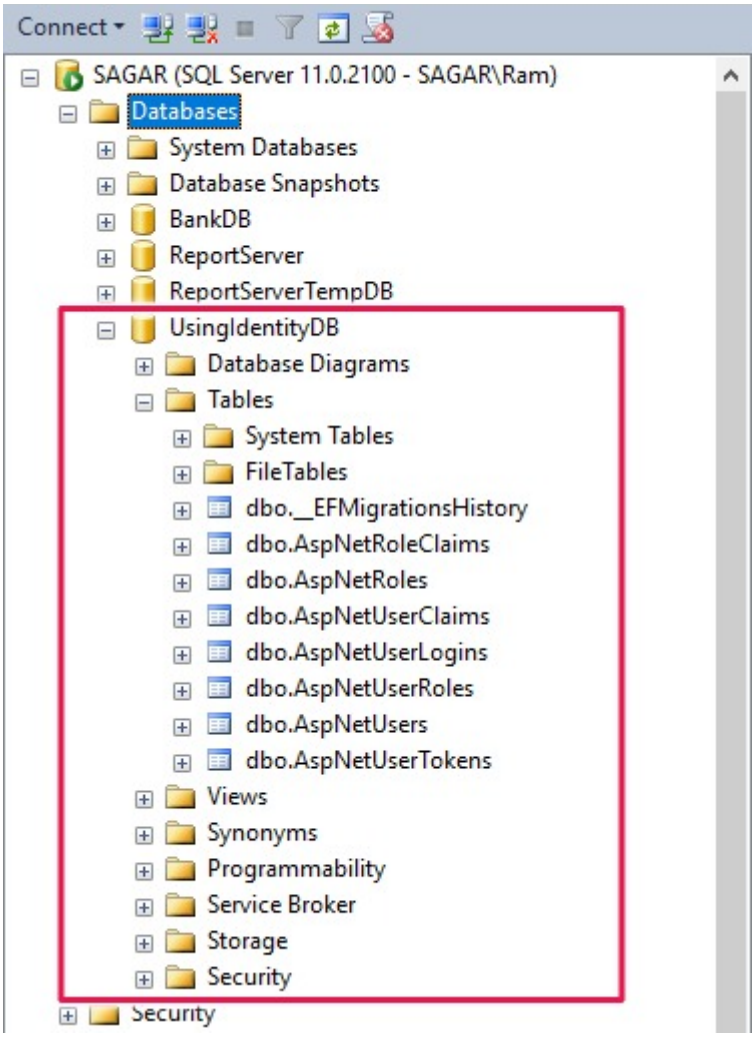
Now open appsttings.json. We can find the connection string with the name UsingIdentityContextConnection. By default, it will connect to the local DB.

```
01. {
02.   "Logging": {
03.     "LogLevel": {
04.       "Default": "Information",
05.       "Microsoft": "Warning",
06.       "Microsoft.Hosting.Lifetime": "Information"
07.     }
08.   },
09.   "AllowedHosts": "*",
10.   "ConnectionStrings": {
```

Now open the Package manager console and Execute the command Add-Migration "First-Create" to generate the actual physical DB

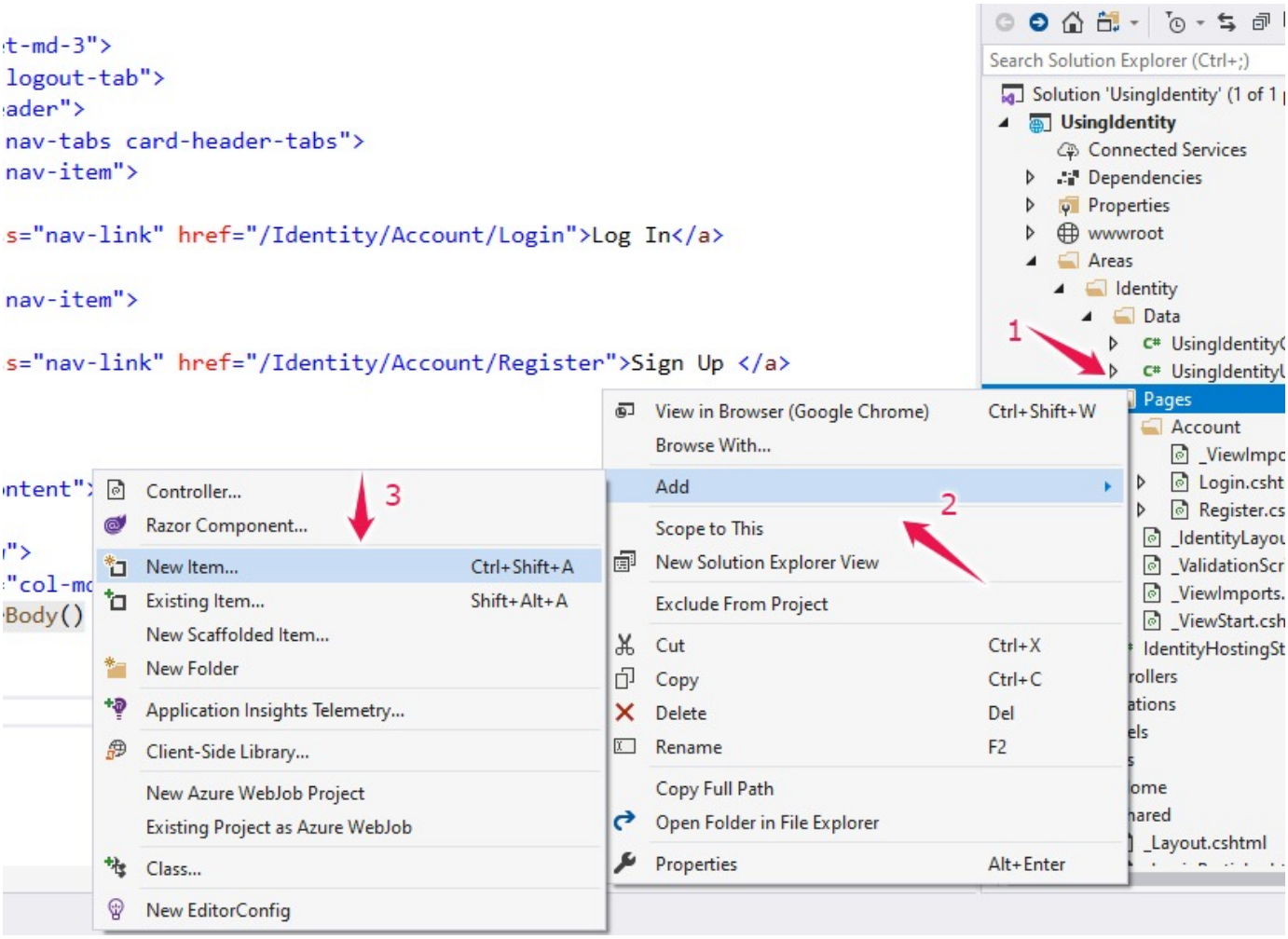


Finally, execute the command Update-Database and you can find the new database as shown below.

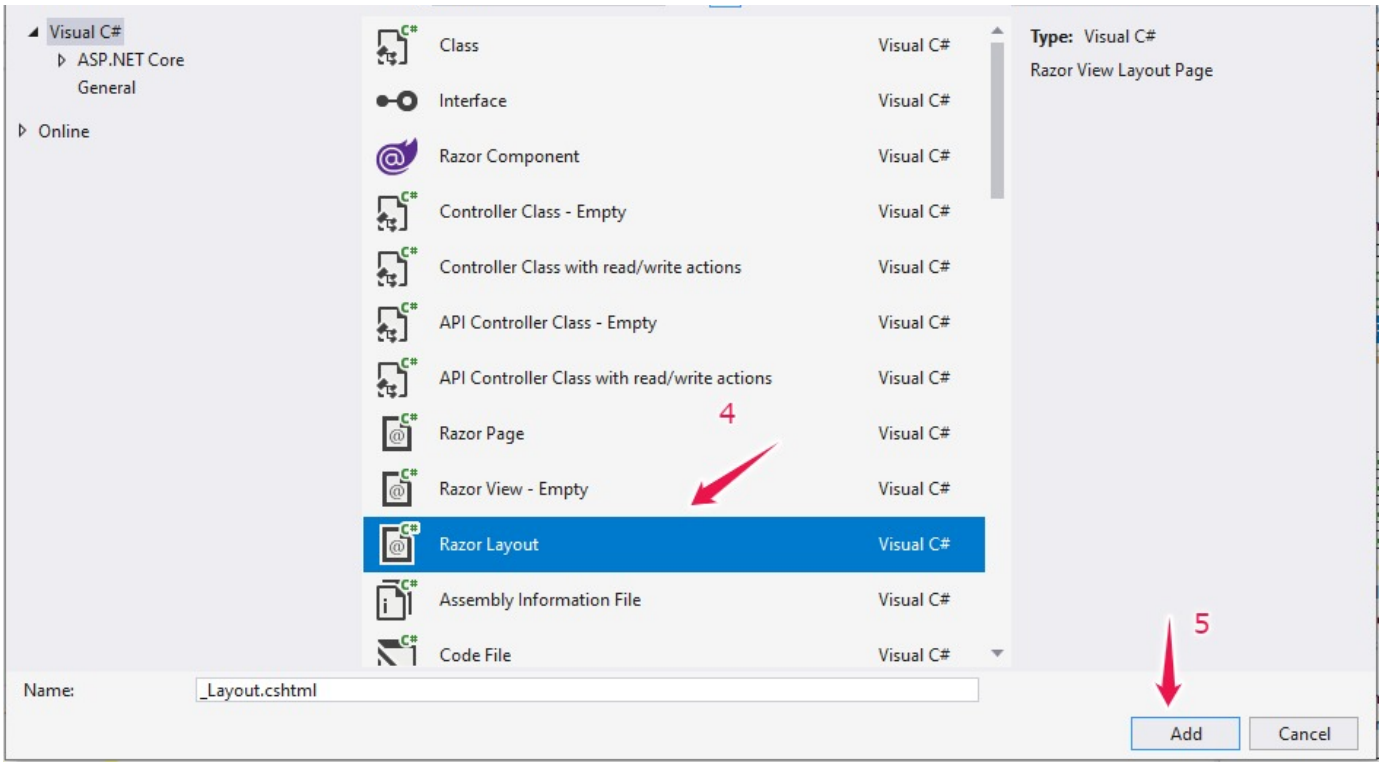


Column Name	Data Type	Allow Nulls
Id	nvarchar(450)	☐
UserName	nvarchar(256)	☑
NormalizedUserName	nvarchar(256)	☑
Email	nvarchar(256)	☑
NormalizedEmail	nvarchar(256)	☑
EmailConfirmed	bit	☐
PasswordHash	nvarchar(MAX)	☑
SecurityStamp	nvarchar(MAX)	☑
ConcurrencyStamp	nvarchar(MAX)	☑
PhoneNumber	nvarchar(MAX)	☑
PhoneNumberConfirmed	bit	☐
TwoFactorEnabled	bit	☐
LockoutEnd	datetimeoffset(7)	☑
LockoutEnabled	bit	☐
AccessFailedCount	int	☐
Firstname	nvarchar(100)	☑
LastName	nvarchar(100)	☑

Now let's start customizing the application. Let's add a Nested layout for tab control headers.



Select Razor layout and click on Add



Customize the layout as below

```
@{
    Layout = "/Views/Shared/_Layout.cshtml";
}
<div class="row">
    <div class="col-md-6  offset-md-3">
        <div class="card login-logout-tab">
            <div class="card-header">
                <ul class="nav nav-tabs card-header-tabs">
                    <li class="nav-item">

                        <a class="nav-link" href="/Identity/Account/Login">Log In</a>
                    </li>
                    <li class="nav-item">

                        <a class="nav-link" href="/Identity/Account/Register">Sign Up </a>
                    </li>
                </ul>
            </div>
            <div class="card-content">

                <div class="row">
                    <div class="col-md-12 ">
                        @RenderBody()
                    </div>
                </div>
            </div>

        </div>
```

```
31. @section Scripts{
32.
33.     @RenderSection("Scripts", required: false)
34.     <script>
35.         $(function () {
36.
37.             var current = location.pathname;
38.             $('.nav-tabs li a').each(function () {
39.                 var $this = $(this);
40.                 if (current.indexOf($this.attr('href')) !== -1) {
41.                     $this.addClass('active');
42.                 }
43.             })
44.         })
45.     </script>
46. }
```

Open site.css and use the below styles

```
01. /*for Tab control*/
02. div.login-logout-tab div.card.header{
03.     padding: 0px 0px 12px 0px;
04. }
05. div.login-logout-tab li.nav-tabs{
06.
07.     margin: 0px 0px -12px 0px;
08. }
09. div.login-logout-tab li.nav-item{
10.
11.     width:50%
12. }
13. div.login-logout-tab a.nav-link{
14.     font-size:25px;
15.     color:#495057;
16.     text-align:center;
17. }
```

Modify the Login page and register the HTML pages with our new layout.

```
01. @page
02. @model LoginModel
03.
04. @{
05.     ViewData["Title"] = "Log in";
06.     Layout = "~/Areas/Identity/Pages/_IdentityLayout.cshtml";
07. }
08.
09. <div class="col-md-10 offset-md-1">
10.     <section>
11.         @*<div class="login-form-icon">
12.             <i class="fas fa-user-circle fa-5x text-secondary">
    </i>
13.         </div>*@
14.         @*fas fa-user-circle fa-9x text-secondary*@
15.         <form id="account" method="post">
```

```
            <div class="form-group">
                <label asp-for="Input.Email"></label>
                <input asp-for="Input.Email" class="form-control" />
                <span asp-validation-for="Input.Email" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Input.Password"></label>
                <input asp-for="Input.Password" class="form-control" />
                <span asp-validation-for="Input.Password" class="text-danger"></span>
            </div>
            <div class="form-group">
                <div class="checkbox">
                    <label asp-for="Input.RememberMe">
                        <input asp-for="Input.RememberMe" />
                        @Html.DisplayNameFor(m => m.Input.RememberMe)
                    </label>
                </div>
            </div>
            <div class="form-group">
                <button type="submit" class="btn btn-primary btn-block">Log in</button>
            </div>
        </form>

    </section>
</div>

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}

@page
@model RegisterModel
@{
    ViewData["Title"] = "Register";
    Layout = "~/Areas/Identity/Pages/_IdentityLayout.cshtml";
}

<h1>@ViewData["Title"]</h1>


<form asp-route-returnUrl="@Model.ReturnUrl" method="post">
    <div asp-validation-summary="All" class="text-danger"></div>
    <div class="row">
        <div class="col-md-6">
            <div class="form-group">
                <label asp-for="Input.FirstName"></label>
                <input asp-for="Input.FirstName" class="form-control" />
                <span asp-validation-for="Input.FirstName" class="text-danger"></span>
            </div>
```

```
68.         <div class="col-md-6">
69.             <div class="form-group">
70.                 <label asp-for="Input.LastName"></label>
71.                 <input asp-for="Input.LastName" class="form-
control" />
72.                 <span asp-validation-for="Input.LastName" class="text-
danger"></span>
73.             </div>
74.         </div>
75.     </div>
76.     <div class="row">
77.         <div class="col-md-6">
78.             <div class="form-group">
79.                 <label asp-for="Input.Email"></label>
80.                 <input asp-for="Input.Email" class="form-
control" />
81.                 <span asp-validation-for="Input.Email" class="text-
danger"></span>
82.             </div>
83.         </div>
84.         <div class="col-md-6">
85.             <div class="form-group">
86.                 <label asp-for="Input.PhoneNumber"></label>
87.                 <input asp-for="Input.PhoneNumber" class="form-
control" />
88.                 <span asp-validation-
for="Input.PhoneNumber" class="text-danger"></span>
89.             </div>
90.         </div>
91.     </div>
92.
93.     <div class="row">
94.         <div class="col-md-6">
95.             <div class="form-group">
96.                 <label asp-for="Input.Password"></label>
97.                 <input asp-for="Input.Password" class="form-
control" />
98.                 <span asp-validation-for="Input.Password" class="text-
danger"></span>
99.             </div>
100.        </div>
101.        <div class="col-md-6">
102.            <div class="form-group">
103.                <label asp-for="Input.ConfirmPassword"></label>
104.                <input asp-for="Input.ConfirmPassword" class="form-
control" />
105.                <span asp-validation-
for="Input.ConfirmPassword" class="text-danger"></span>
106.            </div>
107.        </div>
108.    </div>
109.
110.    <button type="submit" class="btn btn-
primary">Register</button>
111. </form>
112.
```

```
}
```

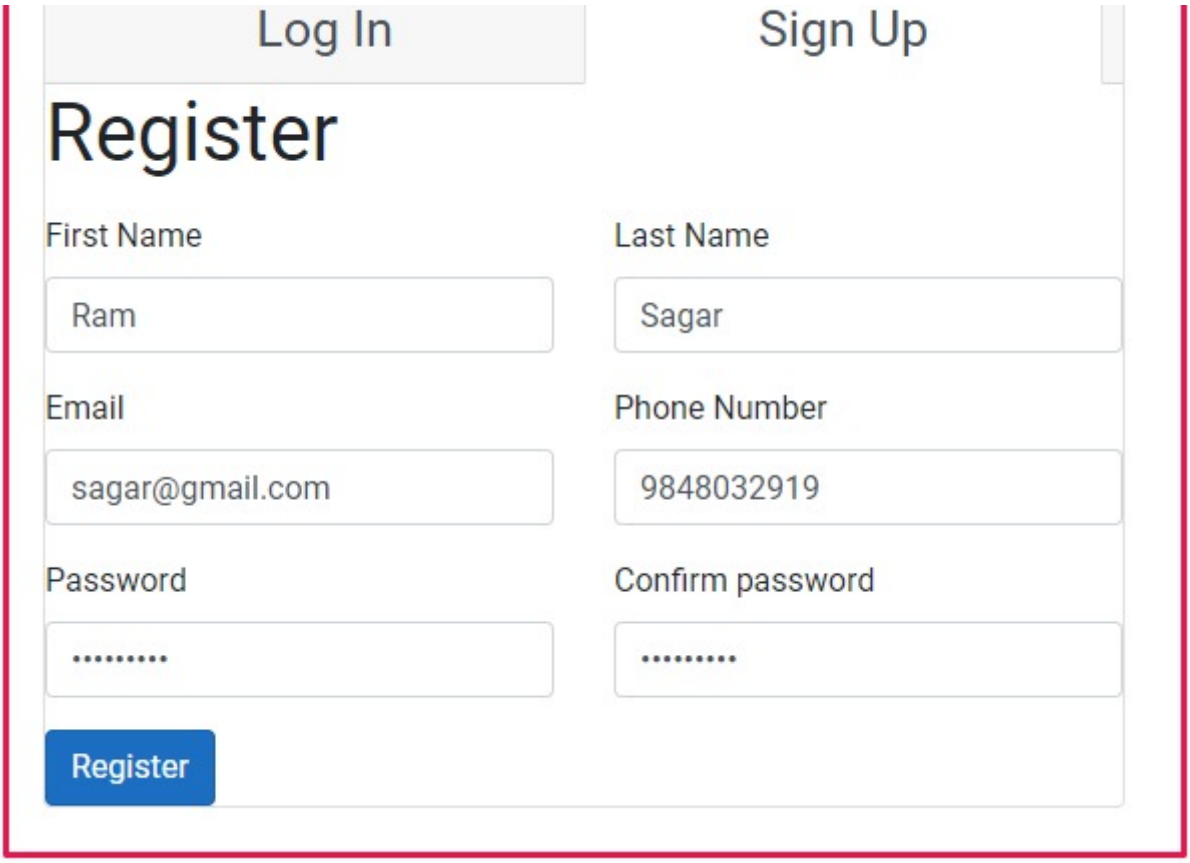
In order to display the active tab, we need to add an active class, as shown below.

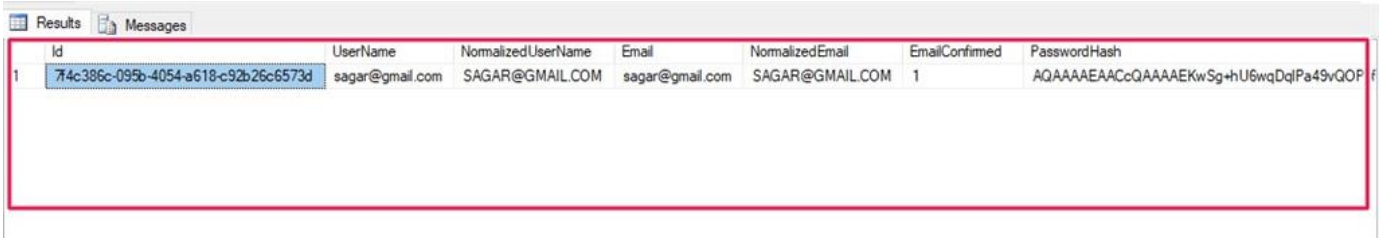
```
@section Scripts{

    @RenderSection("Scripts", required: false)
    <script>
        $(function () {

            var current = location.pathname;
            $('.nav-tabs li a').each(function () {
                var $this = $(this);
                if (current.indexOf($this.attr('href')) !== -1) {
                    $this.addClass('active');
                }
            })
        })
    </script>
}
```

Let's see the output and register a user.

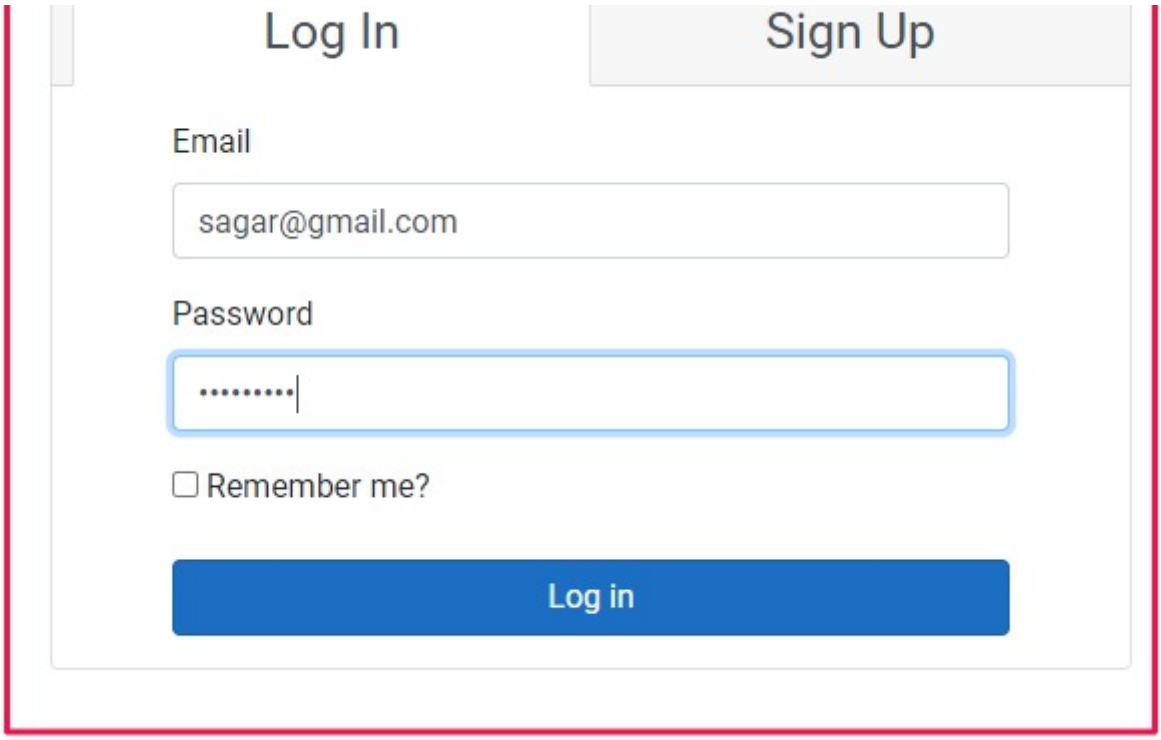


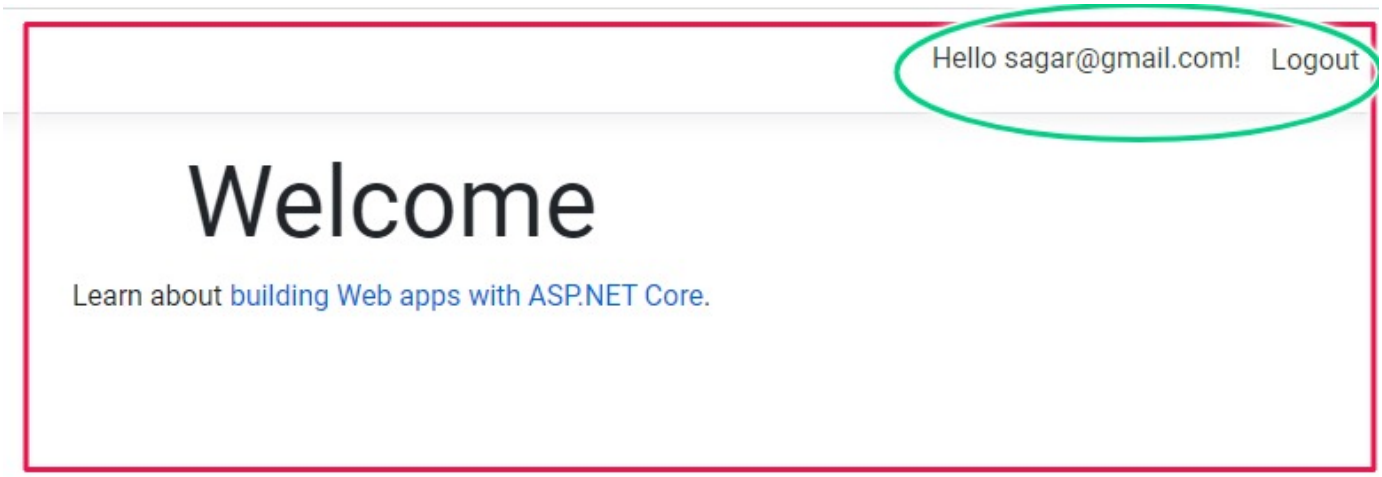


Results | Messages

	Id	UserName	NormalizedUserName	Email	NormalizedEmail	EmailConfirmed	PasswordHash
1	7f4c386c-095b-4054-a618-c92b26c6573d	sagar@gmail.com	SAGAR@GMAIL.COM	sagar@gmail.com	SAGAR@GMAIL.COM	1	AQAAAAEAACcQAAAAEKwSg+hU6wqDqlPa49vQOP

Now let's login with the registered user.





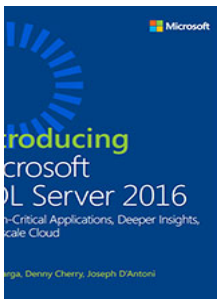
Conclusion

In this article, we discussed how to use Identity UI in ASP.NET Core MVC application by creating a database using the package manager console & commands. I hope you all enjoyed reading this and learned from it.

.Net core | ASP.NET.CORE | Authentication using Identity | Identity

Next Recommended Reading
ASP.NET Core 2.0 JWT Authentication Example

OUR BOOKS



Ramasagar Pulidindi TOP 500

"A Journey of Thousand miles must start with a single Step"I started my programming career with C/C++, Java in my B. Tech days. Later, I got a chance to develop Windows Forms applications using C# and Web appli... Read more

https://www.c-sharpcorner.com/members/ramasagar-pulidindi

198 1.4m 3

9 1

Type your comment here and press Enter Key (Minimum 10 characters)

After LogIn, the Hello @Username and logout doesn't appear

FPL Pramit Mar 22, 2021

NA 71 0 0 0 Reply

FEATURED ARTICLES

Test Coverage For PowerShell Scripts With Pester

Dynamic CSS Values In Blazor

Migrating a SQL IaaS Database To Azure SQL Using DMA (Data Migration Assistant)

Create a Redis Cache with Express Node JS

Symmetrical Repository Pattern - Data Access Made Easy In .NET

TRENDING UP

01 Azure Duration Functions - How To Use And Implement It

03 Log Correlation In Microservices

04 Create A PowerApps Component Framework (PCFx) Using Custom Code In PowerAPPs

05 How To Handle Nullable Reference In .NET 6

06 Import PowerApps Component Framework (PCFx) Into Model Driven PowerApps

07 Growth Mindset Show Ep. 11 - 2022

08 Safest Way To Convert String To Int In C#

09 Creating Various Layouts For Different Razor Pages In Blazor

10 Top Three VS Code Extensions Worth The Money

Learn JavaScript

CHALLENGE YOURSELF

Blockchain Basics Skill

GET CERTIFIED

HTML5 Developer