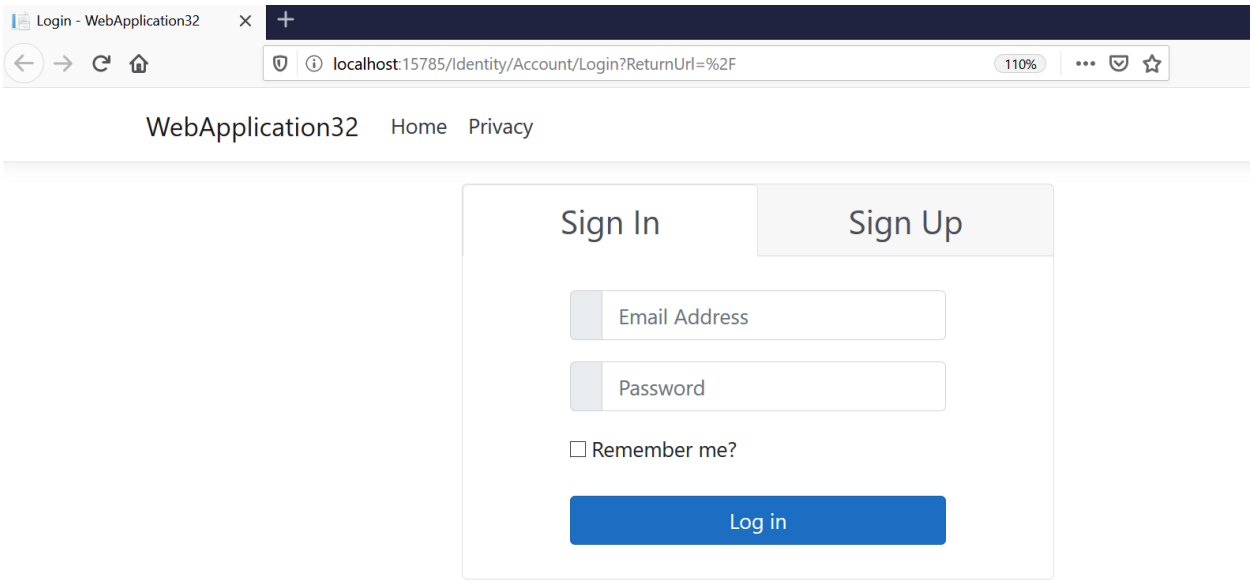
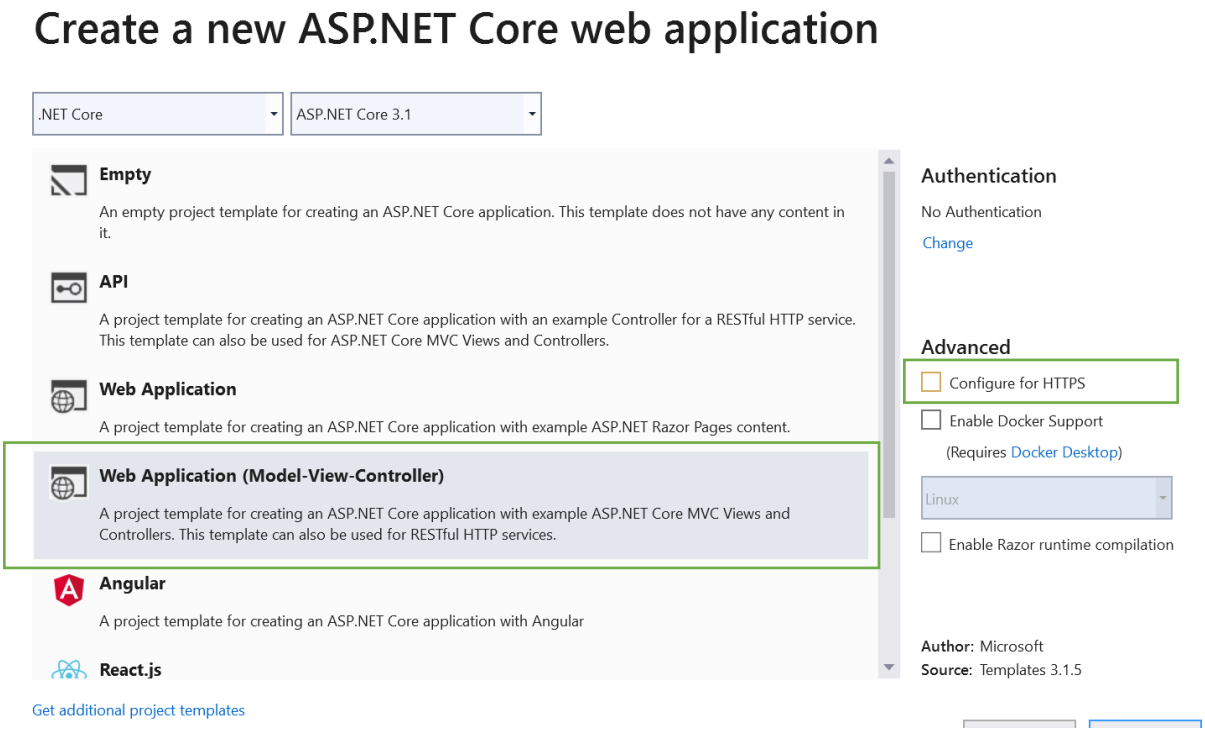
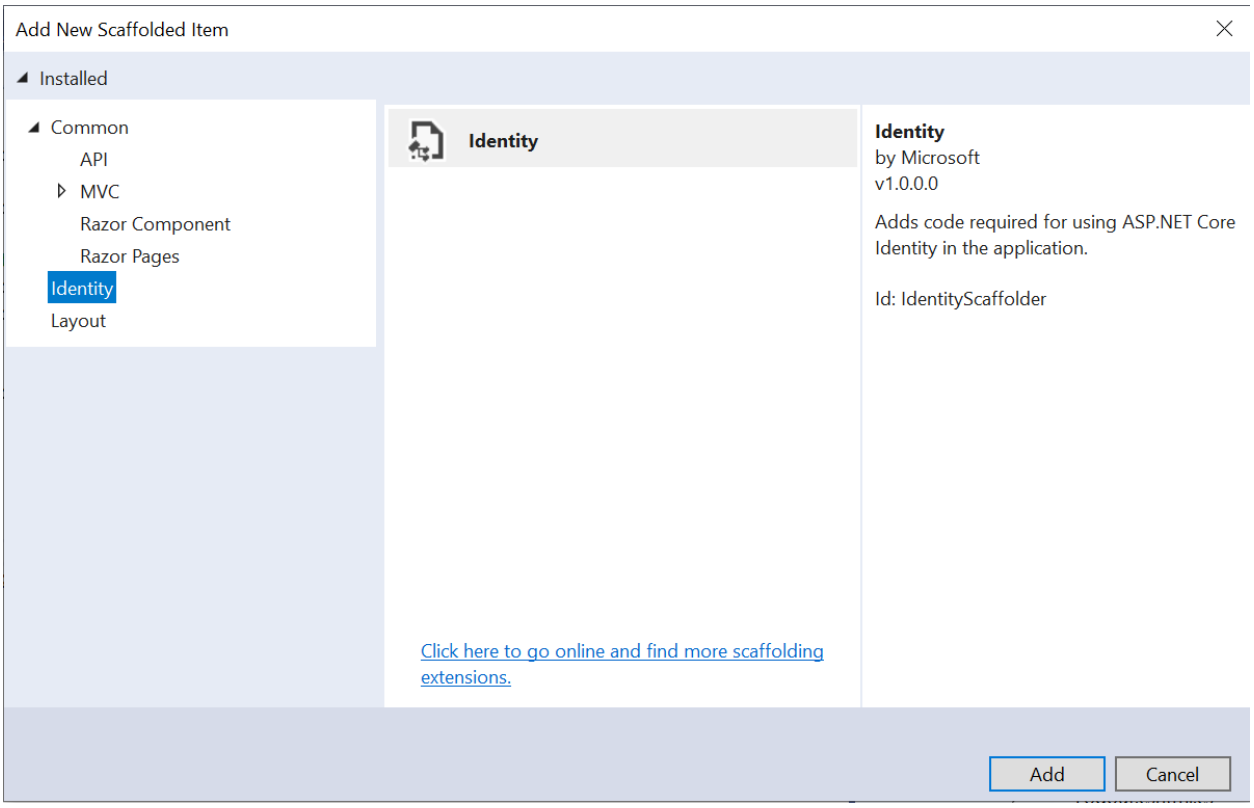
Create a new Project Without Authentication



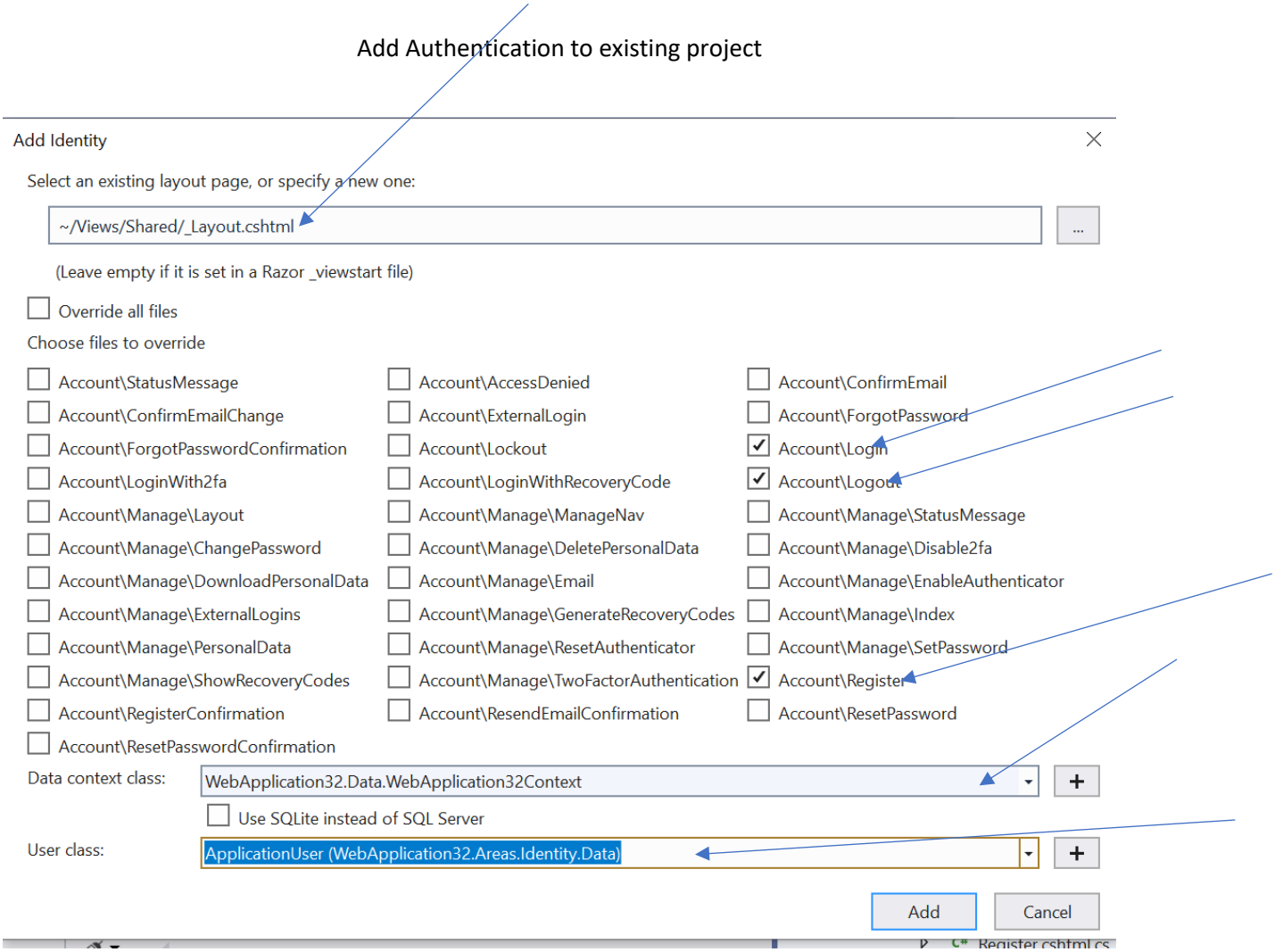


Now add Identity to Existing Project

To add ASP.NET Core Identity to the existing project, right-click on the project. Add > New Scaffolded Item. from the opened window, select Identity from the left panel and click on Add.



Add Authentication to existing project



- Update the startup.cs file, to add the following code

```
// This method gets called by the runtime. Use this method to add services to the container.
0 references
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();

    services.AddRazorPages();
}
```

- In **void configure** add the following code

```
//to be added
app.UseAuthentication();
app.UseAuthorization();


app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id?}");
    //to be added
    endpoints.MapRazorPages();
});
```

- Go to /Areas/Identity/Data/ApplicationUser.cs. to modify the file and add two columns

```
public class ApplicationUser : IdentityUser
{
    [PersonalData]
    [Column(TypeName = "nvarchar(100)")]
    1 reference
    public string FirstName { get; set; }

    [PersonalData]
    [Column(TypeName = "nvarchar(100)")]
    1 reference
    public string LastName { get; set; }
}
```

- go to Tools>NuGet Package Manager > Package Manager Console. This will helps to modify the table **AspNetUsers**, you could see columns for First Name and Last Name.
  Add-Migration "InitialCreate"
  Update-Database

- Now I will work with the Design in order to customize it . go to Views/Shared/_Layout.cshml.

```
<a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">WebApplication32</a>
<button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse" aria-controls="navbarSu
        aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
</button>
<div class="navbar-collapse collapse d-sm-inline-flex flex-sm-row-reverse">
    <ul class="navbar-nav flex-grow-1">
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
        </li>
    </ul>
    <partial name="_LoginPartial" />
</div>
```

- In order to remove the login and register link in the header. We update _LoginPartial as shown below

```
@using Microsoft.AspNetCore.Identity
@using WebApplication32.Areas.Identity.Data

@inject SignInManager<ApplicationUser> SignInManager
@inject UserManager<ApplicationUser> UserManager

<ul class="navbar-nav">
    @if (SignInManager.IsSignedIn(User))
    {
        <li class="nav-item">
            <a id="manage" class="nav-link text-dark" asp-area="Identity" asp-page="/Account/Manage/Index" title="Manage">Hello @UserManager.GetUserName(User)!</a>
        </li>
        <li class="nav-item">
            <form id="logoutForm" class="form-inline" asp-area="Identity" asp-page="/Account/Logout" asp-route-returnUrl="@Url.Action("Index", "Home", new { area = "" })">
                <button id="logout" type="submit" class="nav-link btn btn-link text-dark">Logout</button>
            </form>
        </li>
    }
</ul>
```

- Go to site.css

- Update the css file site.css, by adding the following code

```
body {
    font-family: 'Roboto', sans-serif;
}

/*for tab control*/
div.login-logout-tab div.card-header {
    padding: 0px 0px 12px 0px;
```

```
}

div.login-logout-tab ul.nav-tabs {
   margin: 0px 0px -12px 0px;
}

div.login-logout-tab li.nav-item {
   width: 50%;
}

div.login-logout-tab a.nav-link {
   font-size: 25px;
   color: #495057;
   text-align:center;
}

div.card-content{
   padding : 10px 20px;
}

/*login form*/
div.login-form-icon{
   text-align:center;
}
```

Tab control

- Now we are going to show login and registration form side by side in a **tab control**.
- We need a nested layout page for login and register page.
- Create _AuthLayout.cshtml in /Areas/Identity/Pages.
- For that right-click on Pages folder, Add>NewItem. Select Razor Layout, name the file as _AuthLayout.cshtml. Replace the file as shown 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'>Sign 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="col-md-12">
                @RenderBody()
            </div>
        </div>
    </div>
  </div>
</div>

@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> }
```

User Registration

Go to /Areas/Identity/Pages/Account/*Register.cshtml*.
We've to update that for adding controls for first-name and last-name in user registration form.

```
public class InputModel
{

  [Required]
  [DataType(DataType.Text)]
  [Display(Name = "First Name")]
  public string FirstName { get; set; }

  [Required]
  [DataType(DataType.Text)]
  [Display(Name = "Last Name")]
```

```
    public string LastName { get; set; }

    ...
}
```

- Update ***OnPostAsync***, save the first name and last name to ***ApplicationUser*** instance before inserting into the table.

```
var user = new ApplicationUser { UserName = Input.Email, Email = Input.Email,
    FirstName = Input.FirstName,
    LastName = Input.LastName
};
var result = await _userManager.CreateAsync(user, Input.Password);
```

- Update register.cshtml

```
@page
@model RegisterModel

@{ ViewData["Title"] = "Register"; }

@{ Layout = "~/Areas/Identity/Pages/_AuthLayout.cshtml"; }

<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>
    </div>
    <div class="col-md-6">
      <div class="form-group">
        <label asp-for="Input.LastName"></label>
        <input asp-for="Input.LastName" class="form-control" />
        <span asp-validation-for="Input.LastName" class="text-danger"></span>
      </div>
    </div>
  </div>
  <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="row">
```

```
        <div class="col-md-6">
            <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>
        <div class="col-md-6">
            <div class="form-group">
                <label asp-for="Input.ConfirmPassword"></label>
                <input asp-for="Input.ConfirmPassword" class="form-control" />
                <span asp-validation-for="Input.ConfirmPassword" class="text-danger"></span>
            </div>
        </div>
    </div>
    <button type="submit" class="btn btn-primary">Register</button>
</form>

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

Login Form

User authentication or login is handled inside – /Areas/Identity/Pages/Account/Login.cshtml. Now you can update the razor html as follows.

```
@page
@model LoginModel

@{ViewData["Title"] = "Login";}


@{
    Layout = "~/Areas/Identity/Pages/_AuthLayout.cshtml";
}
<div class="col-md-10 offset-1">
    <div class="login-form-icon">
        <i class="fas fa-user-circle fa-9x text-secondary"></i>
    </div>
    <form id="account" method="post">
        <div asp-validation-summary="All" class="text-danger"></div>
        <div class="form-group">
            <div class="input-group">
```

```
                <div class="input-group-prepend">
                    <div class="input-group-text">
                        <i class="fas fa-envelope"></i>
                    </div>
                </div>
                <input asp-for="Input.Email" class="form-control" placeholder="Email Address" />
            </div>
            <span asp-validation-for="Input.Email" class="text-danger"></span>
        </div>
        <div class="form-group">
            <div class="input-group">
                <div class="input-group-prepend">
                    <div class="input-group-text">
                        <i class="fas fa-lock"></i>
                    </div>
                </div>
                <input asp-for="Input.Password" class="form-control" placeholder="Password" />
            </div>
            <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>
</div>

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

- if you want to redirect the user back to home from login or registration when he/ she is already logged in. you just need to add following if statement in login and register page, inside the function *onGetAsync* as shown below.

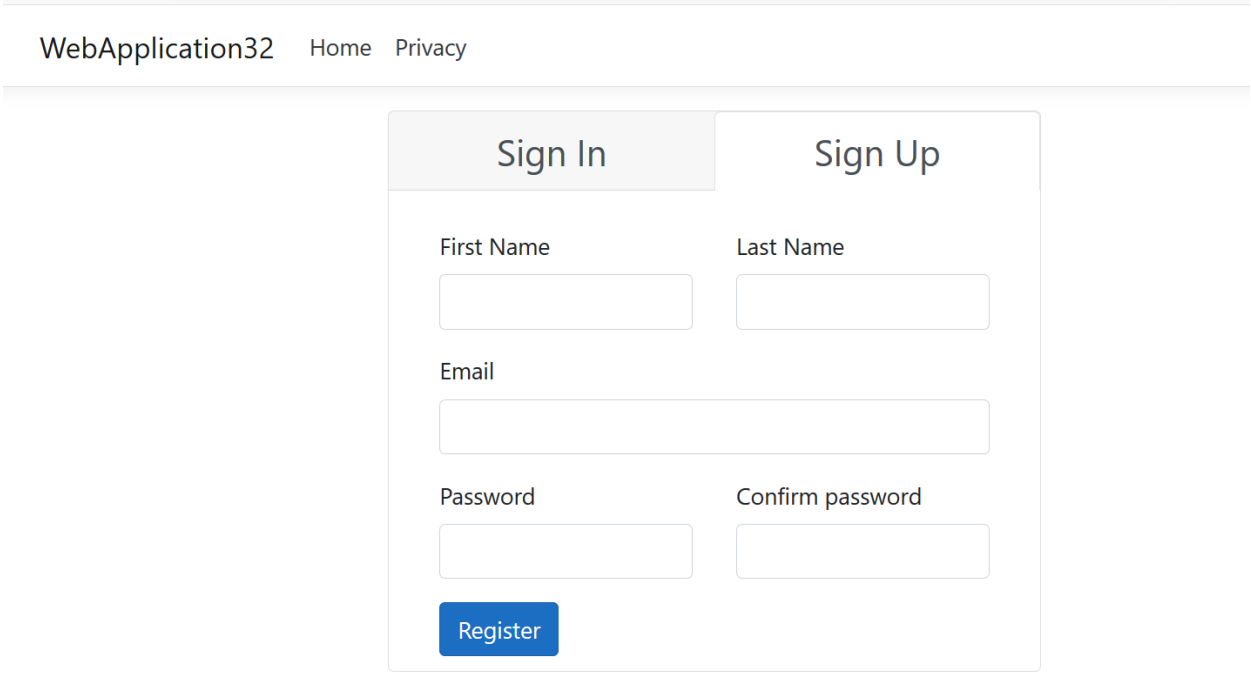
```
0 references
public async Task OnGetAsync(string returnUrl = null)
{
    if (User.Identity.IsAuthenticated)
    {
        Response.Redirect("/Home");
    }
    if (!string.IsNullOrEmpty(ErrorMessage))
    {
        ModelState.AddModelError(string.Empty, ErrorMessage);
    }
```

- add authorize to home controller. you could protect controllers or actions from unauthorized request using *Authorize* attribute If there is any unauthorized request made, user will be redirected to login page.

```
namespace WebApplication32.Controllers
{
    [Authorize]
    3 references
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;

        0 references
        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }

        0 references
        public IActionResult Index()
```


WebApplication32
Home
Privacy
Sign In
Sign Up
First Name
Last Name
Email
Password
Confirm password
Register