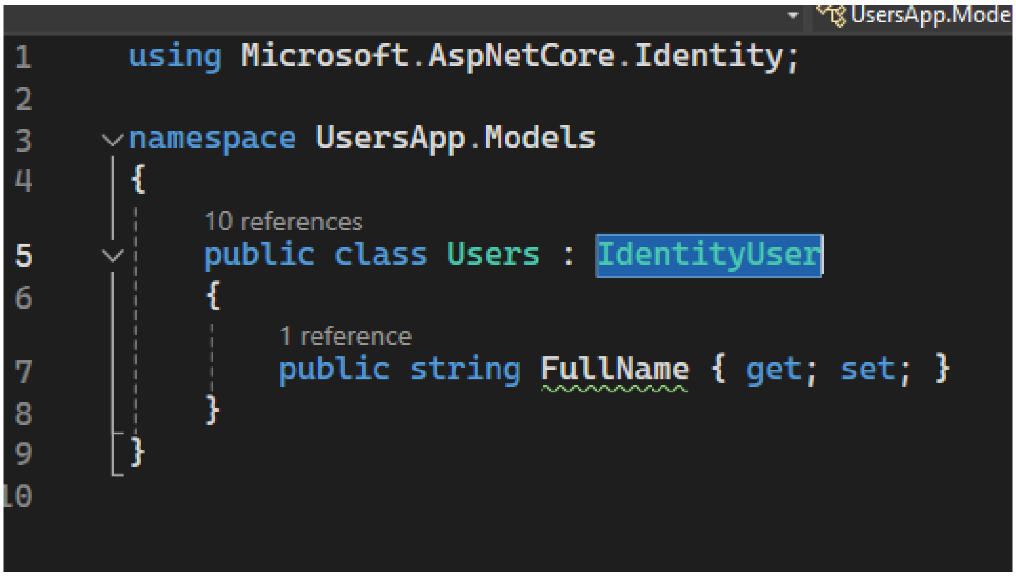
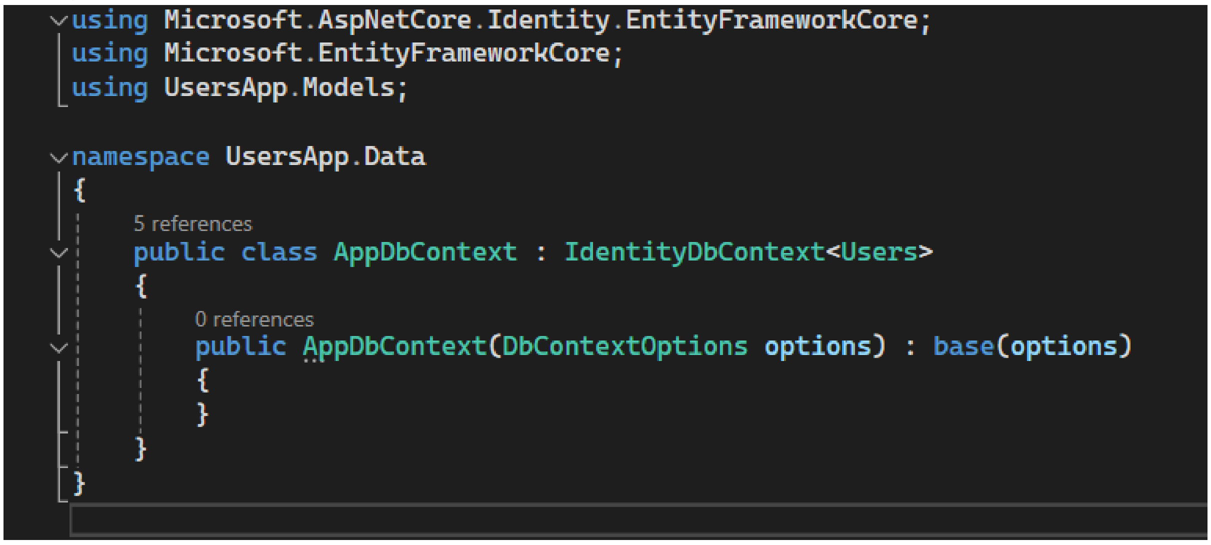
Asp.net with entity framework identity

1. Install vs
2. Create project ASP.net mvc .net 8.0 (long term support)
3. Install in **m.aspnetcore.identity.entityframwork**
4. Install m. **entityframwork**
5. Install **sqlsrver**
6. Install **tools**
7. Create model Users
8. Inherit this class IdentityUser and create class property



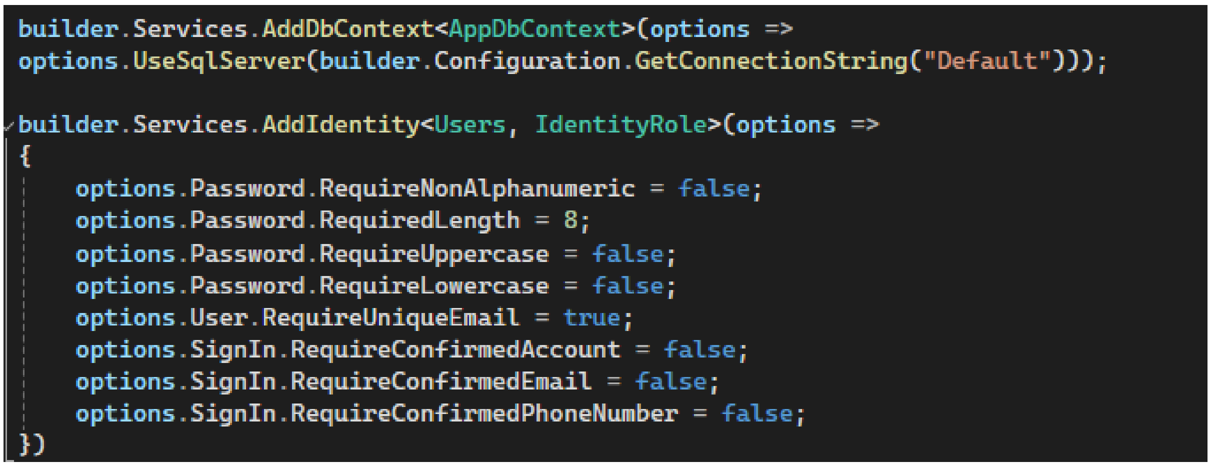
```csharp
using Microsoft.AspNetCore.Identity;

namespace UsersApp.Models
{
    public class Users : IdentityUser
    {
        public string FullName { get; set; }
    }
}
```

9. Now create a folder and Data and inside it create **AppDbContext** class and inherit this class IdentityDbContext and pass Users class it and creation constructor with options



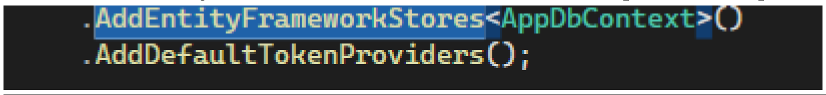
```csharp
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using UsersApp.Models;

namespace UsersApp.Data
{
    public class AppDbContext : IdentityDbContext<Users>
    {
        public AppDbContext(DbContextOptions options) : base(options)
        {
        }
    }
}
```

10. Now go to programm.cs file and add database connection and now use

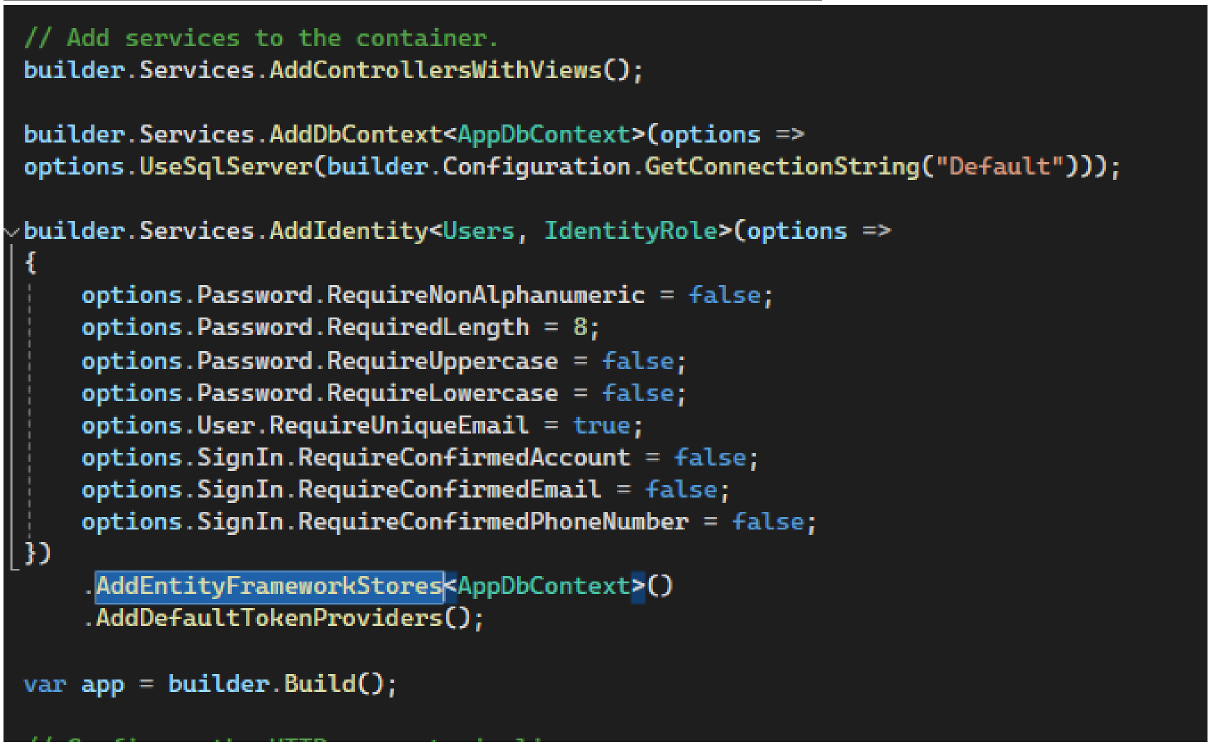


```
builder.Services.AddDbContext<AppDbContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("Default")));

builder.Services.AddIdentity<Users, IdentityRole>(options =>
{
    options.Password.RequireNonAlphanumeric = false;
    options.Password.RequiredLength = 8;
    options.Password.RequireUppercase = false;
    options.Password.RequireLowercase = false;
    options.User.RequireUniqueEmail = true;
    options.SignIn.RequireConfirmedAccount = false;
    options.SignIn.RequireConfirmedEmail = false;
    options.SignIn.RequireConfirmedPhoneNumber = false;
})
```

11. After add AddEntityFrameworkStores method and add tokenprovider and pass AppDbContext



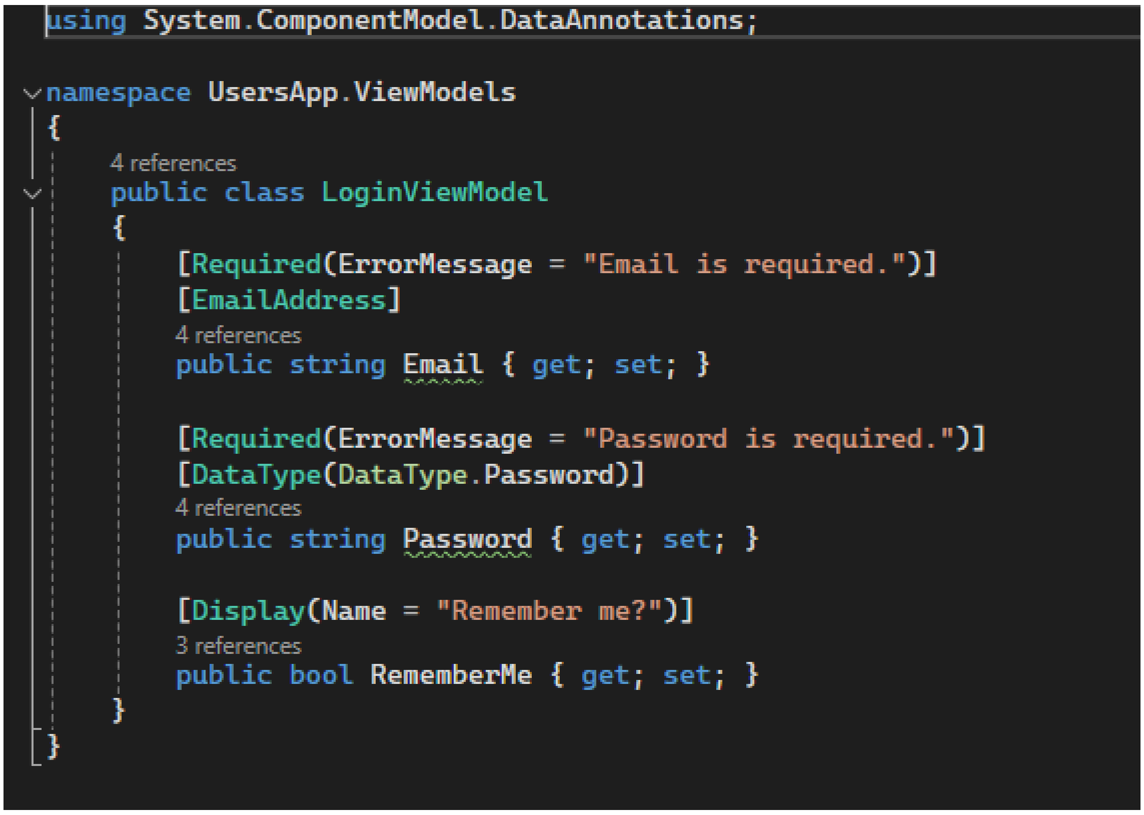
```
    .AddEntityFrameworkStores<AppDbContext>()
    .AddDefaultTokenProviders();
```



```
// Add services to the container.
builder.Services.AddControllersWithViews();

builder.Services.AddDbContext<AppDbContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("Default")));

builder.Services.AddIdentity<Users, IdentityRole>(options =>
{
    options.Password.RequireNonAlphanumeric = false;
    options.Password.RequiredLength = 8;
    options.Password.RequireUppercase = false;
    options.Password.RequireLowercase = false;
    options.User.RequireUniqueEmail = true;
    options.SignIn.RequireConfirmedAccount = false;
    options.SignIn.RequireConfirmedEmail = false;
    options.SignIn.RequireConfirmedPhoneNumber = false;
})
    .AddEntityFrameworkStores<AppDbContext>()
    .AddDefaultTokenProviders();

var app = builder.Build();
```

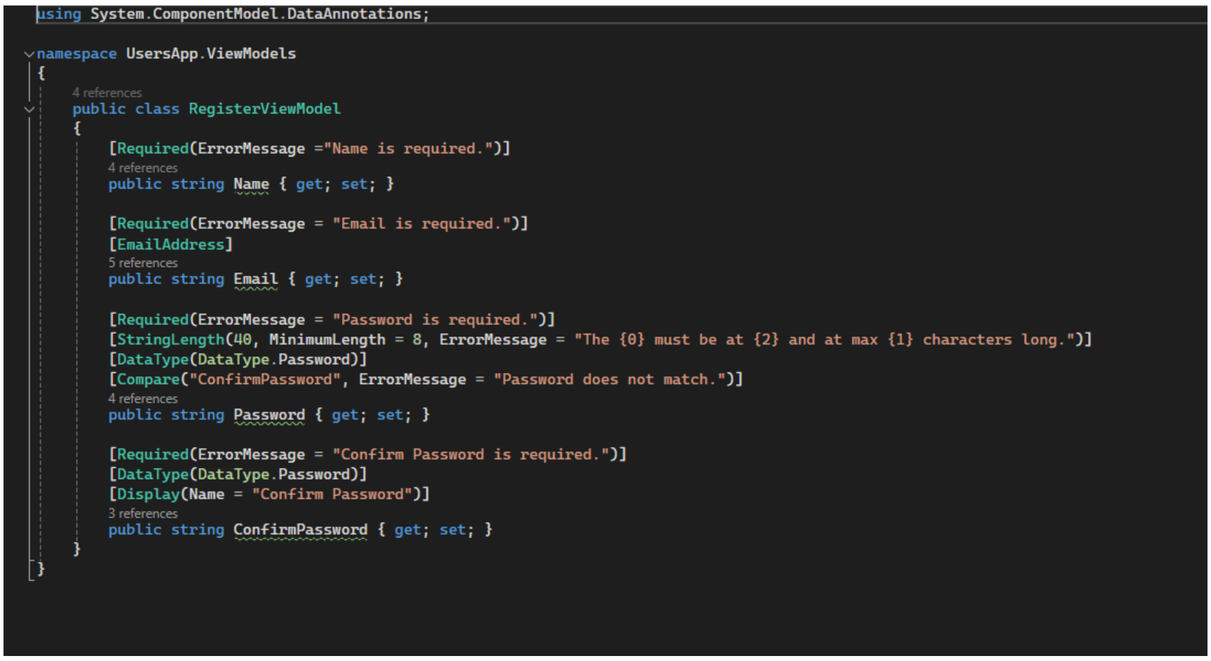
12.
13. Open console and hit add-migration frist than update-database
14. **Now create view models classes**

15. Create folder ViewModels add class LoginViewModel inside ir create property



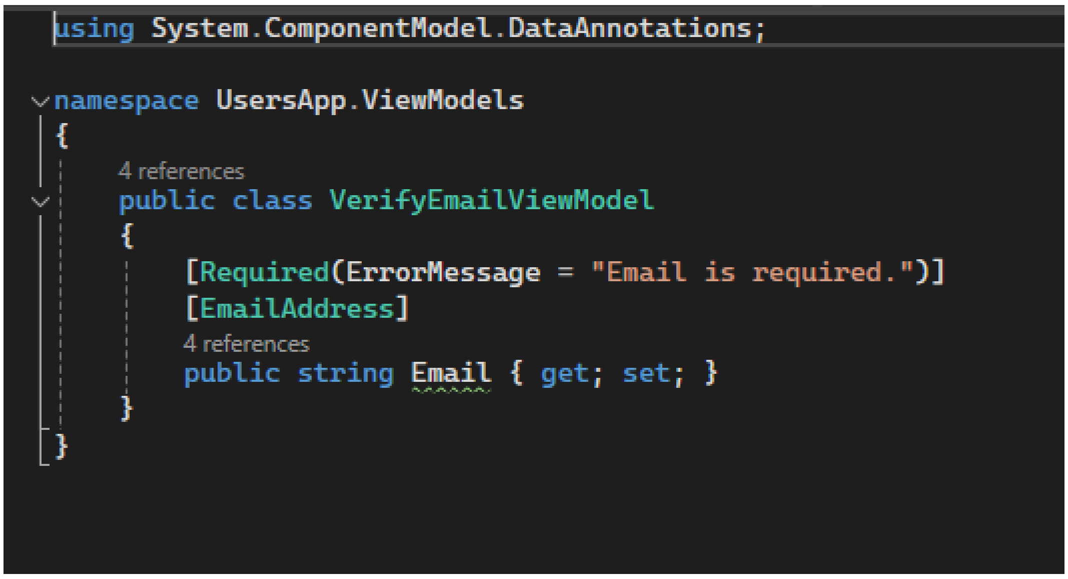
```
using System.ComponentModel.DataAnnotations;

namespace UsersApp.ViewModels
{
    4 references
    public class LoginViewModel
    {
        [Required(ErrorMessage = "Email is required.")]
        [EmailAddress]
        4 references
        public string Email { get; set; }

        [Required(ErrorMessage = "Password is required.")]
        [DataType(DataType.Password)]
        4 references
        public string Password { get; set; }

        [Display(Name = "Remember me?")]
        3 references
        public bool RememberMe { get; set; }
    }
}
```

16. Now create RegisterViewModel



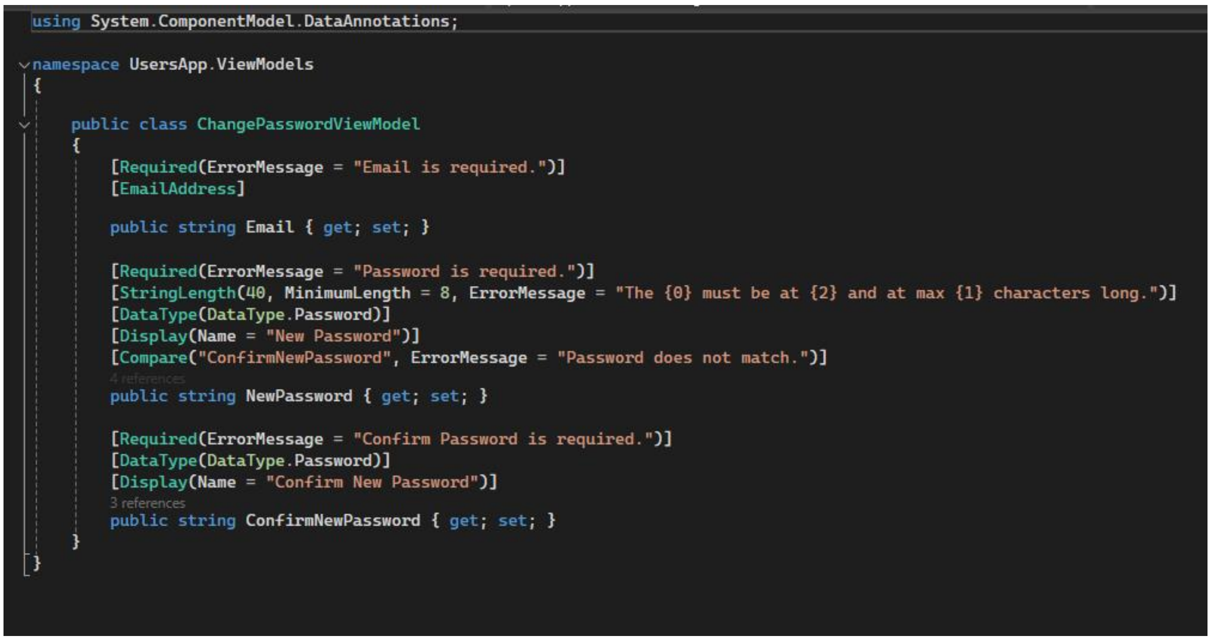
```
using System.ComponentModel.DataAnnotations;

namespace UsersApp.ViewModels
{
    4 references
    public class RegisterViewModel
    {
        [Required(ErrorMessage ="Name is required.")]
        4 references
        public string Name { get; set; }

        [Required(ErrorMessage = "Email is required.")]
        [EmailAddress]
        5 references
        public string Email { get; set; }

        [Required(ErrorMessage = "Password is required.")]
        [StringLength(40, MinimumLength = 8, ErrorMessage = "The {0} must be at {2} and at max {1} characters long.")]
        [DataType(DataType.Password)]
        [Compare("ConfirmPassword", ErrorMessage = "Password does not match.")]
        4 references
        public string Password { get; set; }

        [Required(ErrorMessage = "Confirm Password is required.")]
        [DataType(DataType.Password)]
        [Display(Name = "Confirm Password")]
        3 references
        public string ConfirmPassword { get; set; }
    }
}
```

17. Now create VerifyEmailViewModel



```
using System.ComponentModel.DataAnnotations;

namespace UsersApp.ViewModels
{
    4 references
    public class VerifyEmailViewModel
    {
        [Required(ErrorMessage = "Email is required.")]
        [EmailAddress]
        4 references
        public string Email { get; set; }
    }
}
```

18. Now create ChangePasswordViewModel



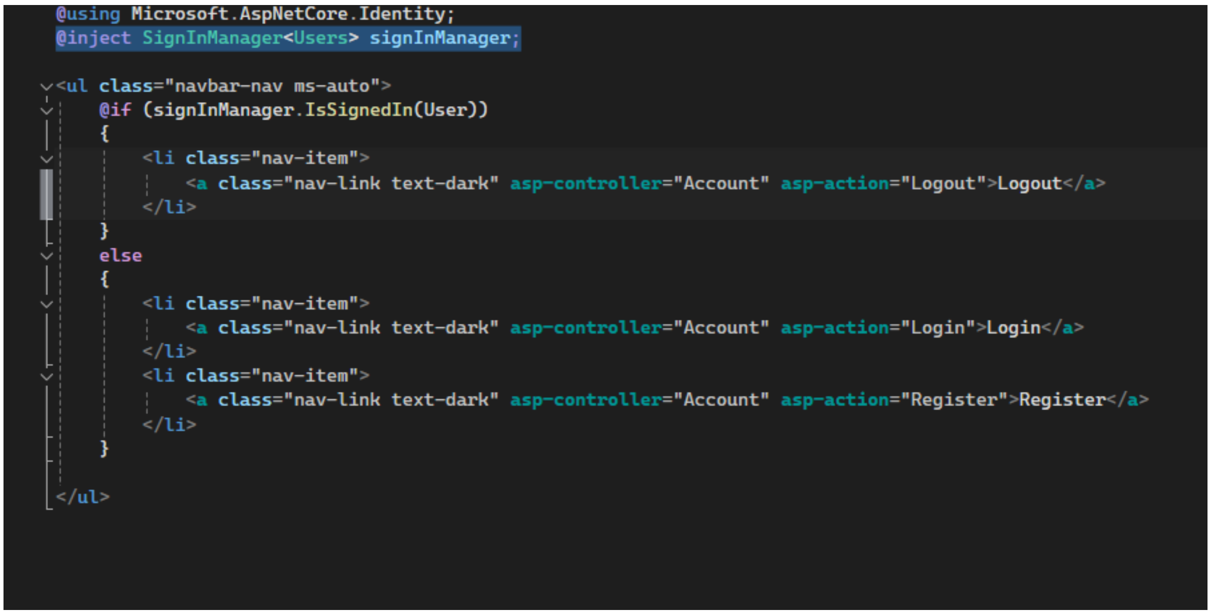
```
using System.ComponentModel.DataAnnotations;

namespace UsersApp.ViewModels
{
    public class ChangePasswordViewModel
    {
        [Required(ErrorMessage = "Email is required.")]
        [EmailAddress]

        public string Email { get; set; }

        [Required(ErrorMessage = "Password is required.")]
        [StringLength(40, MinimumLength = 8, ErrorMessage = "The {0} must be at {2} and at max {1} characters long.")]
        [DataType(DataType.Password)]
        [Display(Name = "New Password")]
        [Compare("ConfirmNewPassword", ErrorMessage = "Password does not match.")]
        4 references
        public string NewPassword { get; set; }

        [Required(ErrorMessage = "Confirm Password is required.")]
        [DataType(DataType.Password)]
        [Display(Name = "Confirm New Password")]
        3 references
        public string ConfirmNewPassword { get; set; }
    }
}
```

19. Now time to create _LoginPartial.cshtml in Shared folder

delete everythings

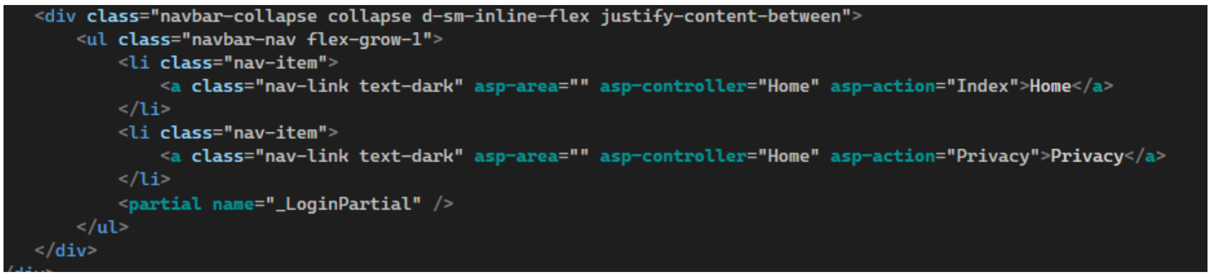
```
@using Microsoft.AspNetCore.Identity;
```

@inject SignInManager<Users> signInManager;



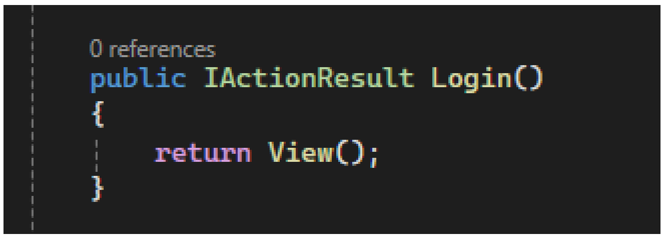
```
@using Microsoft.AspNetCore.Identity;
@inject SignInManager<Users> signInManager;

<ul class="navbar-nav ms-auto">
    @if (signInManager.IsSignedIn(User))
    {
        <li class="nav-item">
            <a class="nav-link text-dark" asp-controller="Account" asp-action="Logout">Logout</a>
        </li>
    }
    else
    {
        <li class="nav-item">
            <a class="nav-link text-dark" asp-controller="Account" asp-action="Login">Login</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-controller="Account" asp-action="Register">Register</a>
        </li>
    }

</ul>
```

20. Than add in _Layout



```
<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
    <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>
        <partial name="_LoginPartial" />
    </ul>
</div>
```

21. Now create controller in controller folder and name AccountController

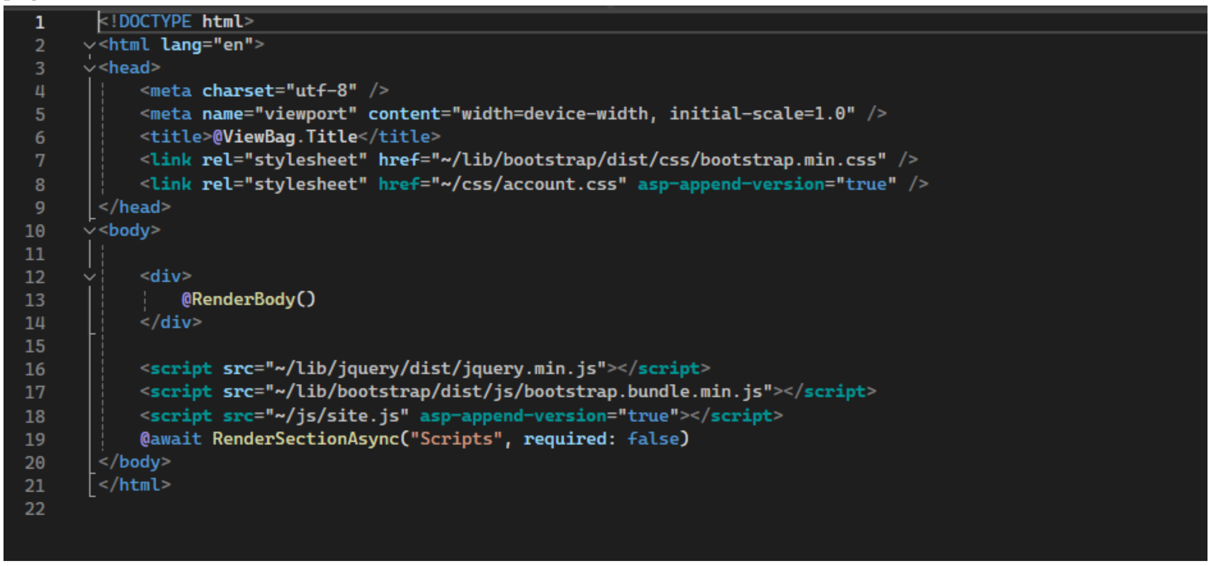


```
0 references
public IActionResult Login()
{
    return View();
}
```

22. Now add view Login

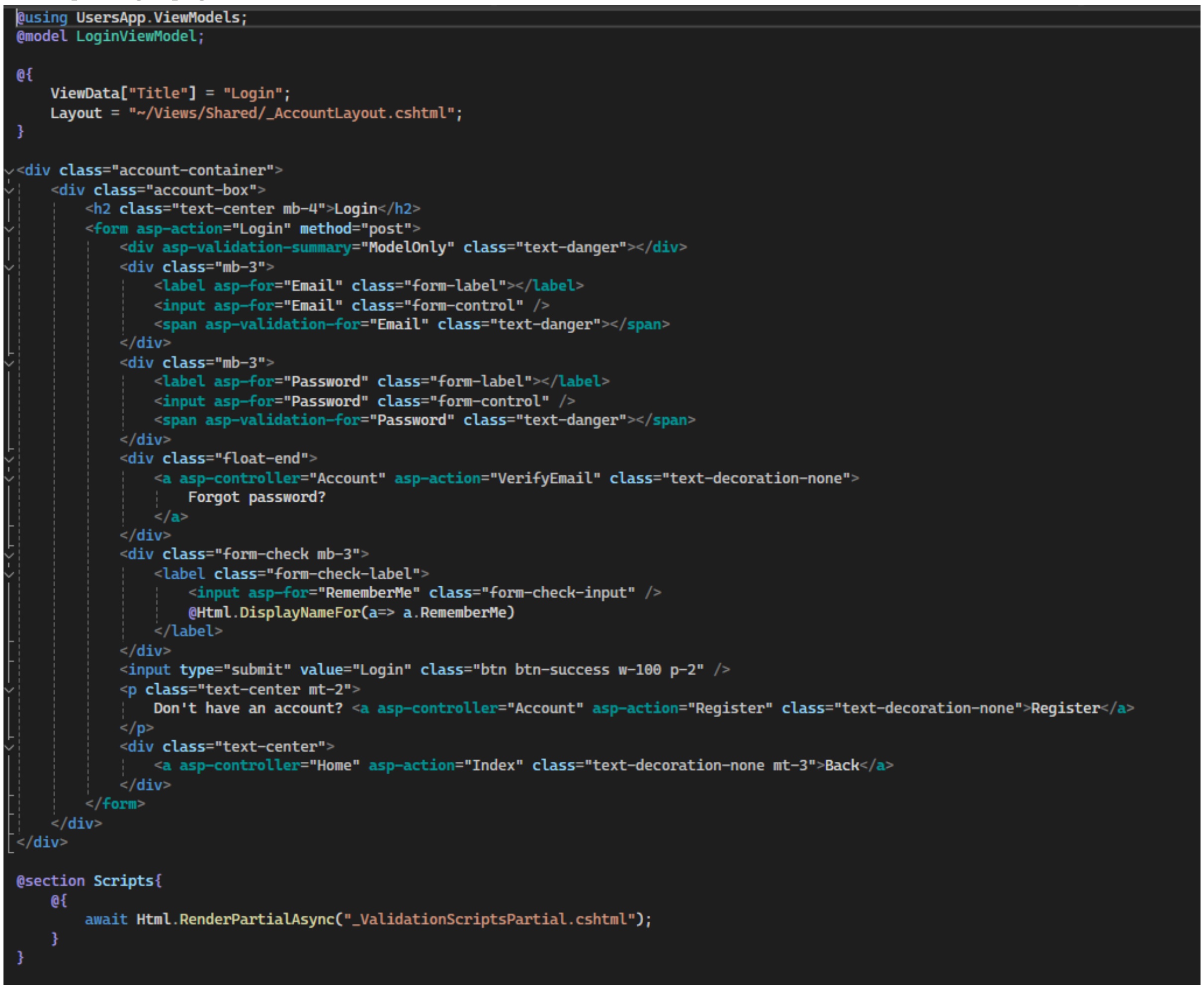
23. Now create _AccountLayout.cshtml in shared folder

24. Open _layout and copy all and pest there in _Accountlayout than remove header and footer also change the page title



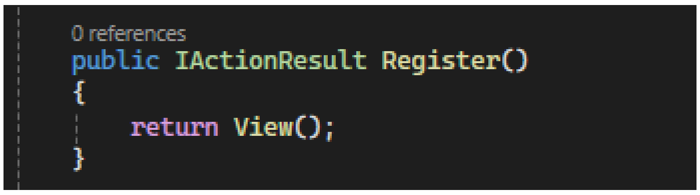
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewBag.Title</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/account.css" asp-append-version="true" />
</head>
<body>

    <div>
        @RenderBody()
    </div>

    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>

```

25. Now open login page



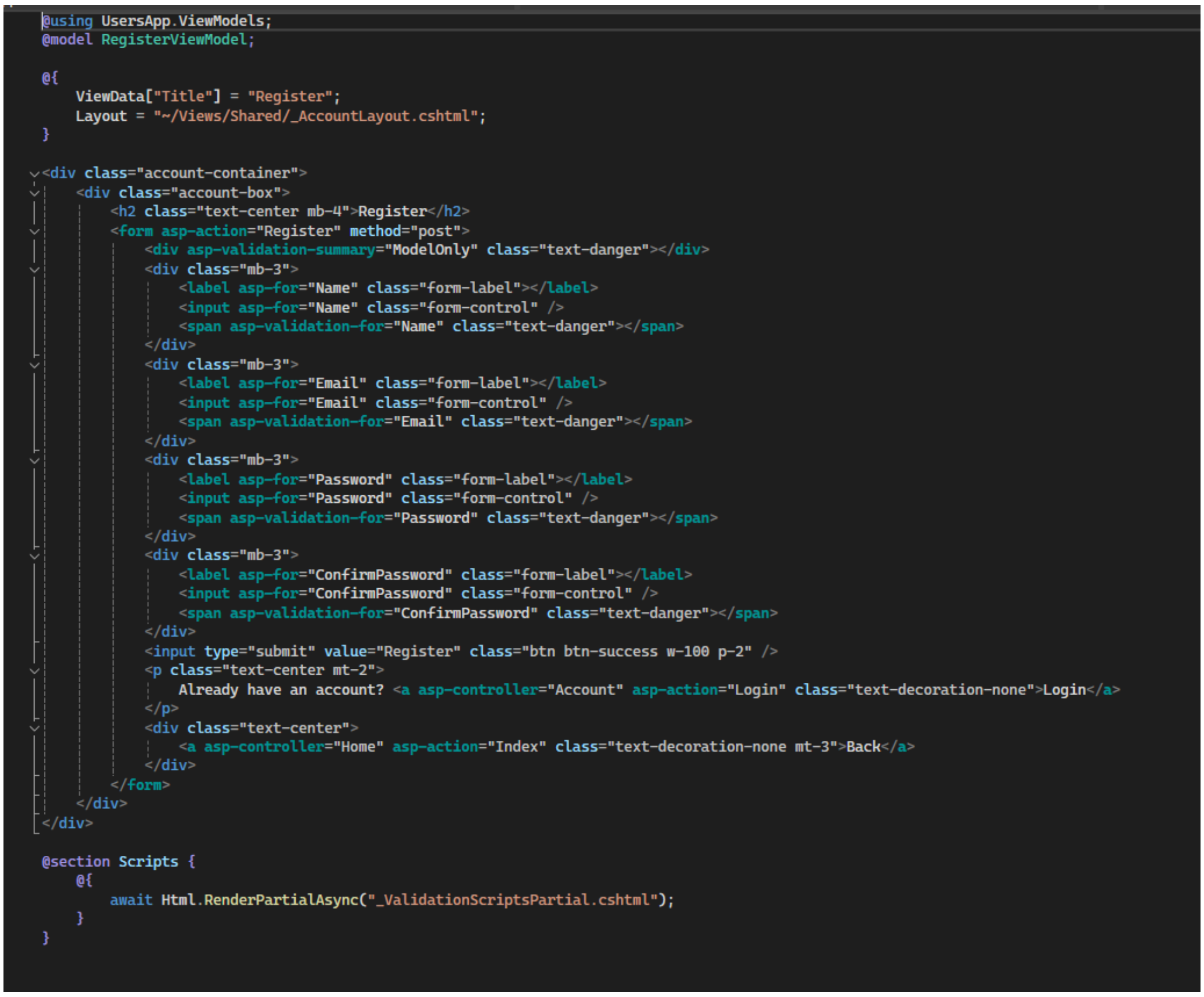
```
@using UsersApp.ViewModels;
@model LoginViewModel;

@{
    ViewData["Title"] = "Login";
    Layout = "~/Views/Shared/_AccountLayout.cshtml";
}

<div class="account-container">
    <div class="account-box">
        <h2 class="text-center mb-4">Login</h2>
        <form asp-action="Login" method="post">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="mb-3">
                <label asp-for="Email" class="form-label"></label>
                <input asp-for="Email" class="form-control" />
                <span asp-validation-for="Email" class="text-danger"></span>
            </div>
            <div class="mb-3">
                <label asp-for="Password" class="form-label"></label>
                <input asp-for="Password" class="form-control" />
                <span asp-validation-for="Password" class="text-danger"></span>
            </div>
            <div class="float-end">
                <a asp-controller="Account" asp-action="VerifyEmail" class="text-decoration-none">
                    Forgot password?
                </a>
            </div>
            <div class="form-check mb-3">
                <label class="form-check-label">
                    <input asp-for="RememberMe" class="form-check-input" />
                    @Html.DisplayNameFor(a=> a.RememberMe)
                </label>
            </div>
            <input type="submit" value="Login" class="btn btn-success w-100 p-2" />
            <p class="text-center mt-2">
                Don't have an account? <a asp-controller="Account" asp-action="Register" class="text-decoration-none">Register</a>
            </p>
            <div class="text-center">
                <a asp-controller="Home" asp-action="Index" class="text-decoration-none mt-3">Back</a>
            </div>
        </form>
    </div>
</div>

@section Scripts{
    @{
        await Html.RenderPartialAsync("_ValidationScriptsPartial.cshtml");
    }
}
```

26. Now create registerpage open account controller



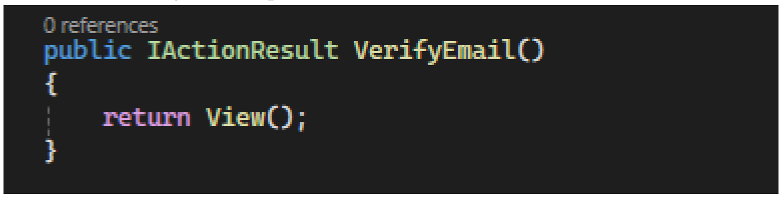
```
0 references
public IActionResult Register()
{
    return View();
}
```

27. Now create register view page



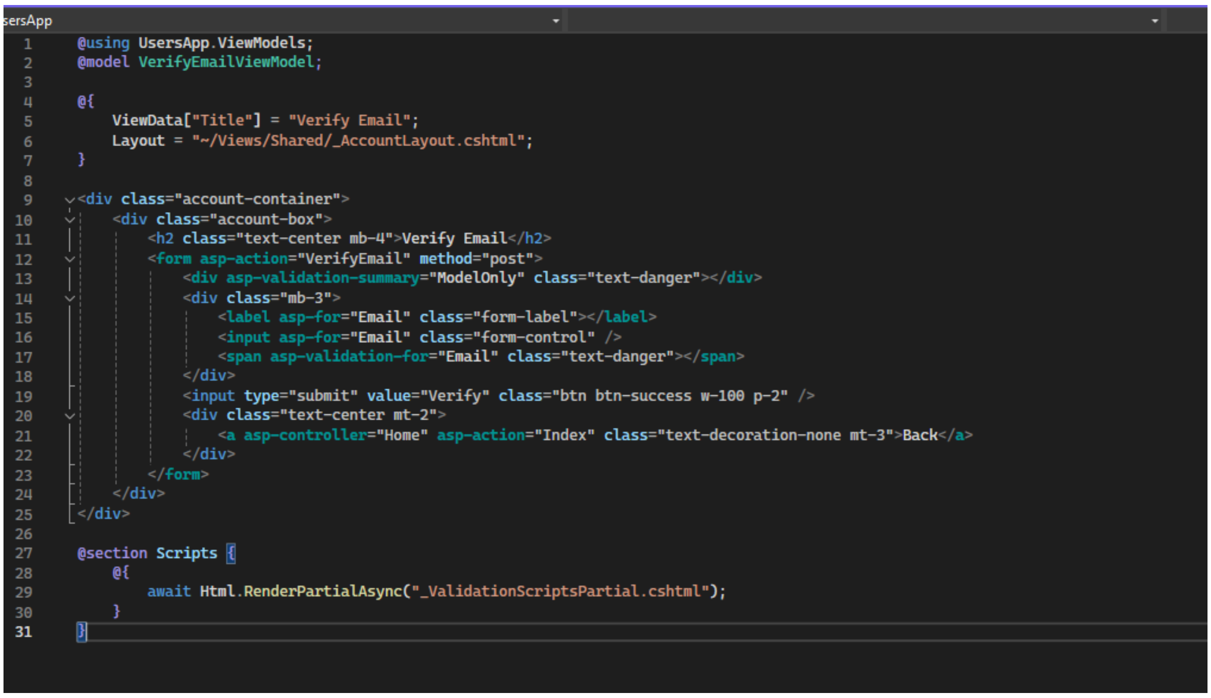
```
@using UsersApp.ViewModels;
@model RegisterViewModel;

@{
    ViewData["Title"] = "Register";
    Layout = "~/Views/Shared/_AccountLayout.cshtml";
}

<div class="account-container">
    <div class="account-box">
        <h2 class="text-center mb-4">Register</h2>
        <form asp-action="Register" method="post">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="mb-3">
                <label asp-for="Name" class="form-label"></label>
                <input asp-for="Name" class="form-control" />
                <span asp-validation-for="Name" class="text-danger"></span>
            </div>
            <div class="mb-3">
                <label asp-for="Email" class="form-label"></label>
                <input asp-for="Email" class="form-control" />
                <span asp-validation-for="Email" class="text-danger"></span>
            </div>
            <div class="mb-3">
                <label asp-for="Password" class="form-label"></label>
                <input asp-for="Password" class="form-control" />
                <span asp-validation-for="Password" class="text-danger"></span>
            </div>
            <div class="mb-3">
                <label asp-for="ConfirmPassword" class="form-label"></label>
                <input asp-for="ConfirmPassword" class="form-control" />
                <span asp-validation-for="ConfirmPassword" class="text-danger"></span>
            </div>
            <input type="submit" value="Register" class="btn btn-success w-100 p-2" />
            <p class="text-center mt-2">
                Already have an account? <a asp-controller="Account" asp-action="Login" class="text-decoration-none">Login</a>
            </p>
            <div class="text-center">
                <a asp-controller="Home" asp-action="Index" class="text-decoration-none mt-3">Back</a>
            </div>
        </form>
    </div>
</div>

@section Scripts {
    @{
        await Html.RenderPartialAsync("_ValidationScriptsPartial.cshtml");
    }
}
```

28. Now create verify email open account controller



```
0 references
public IActionResult VerifyEmail()
{
    return View();
}
```

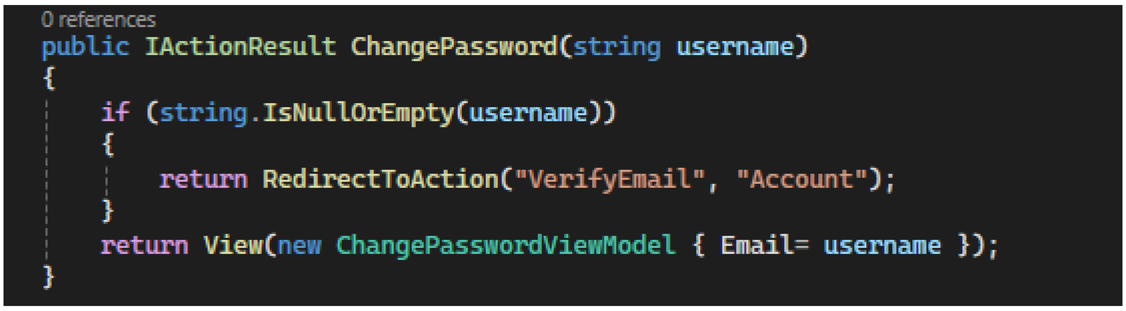
29. Now create verifyemail view page



sersApp

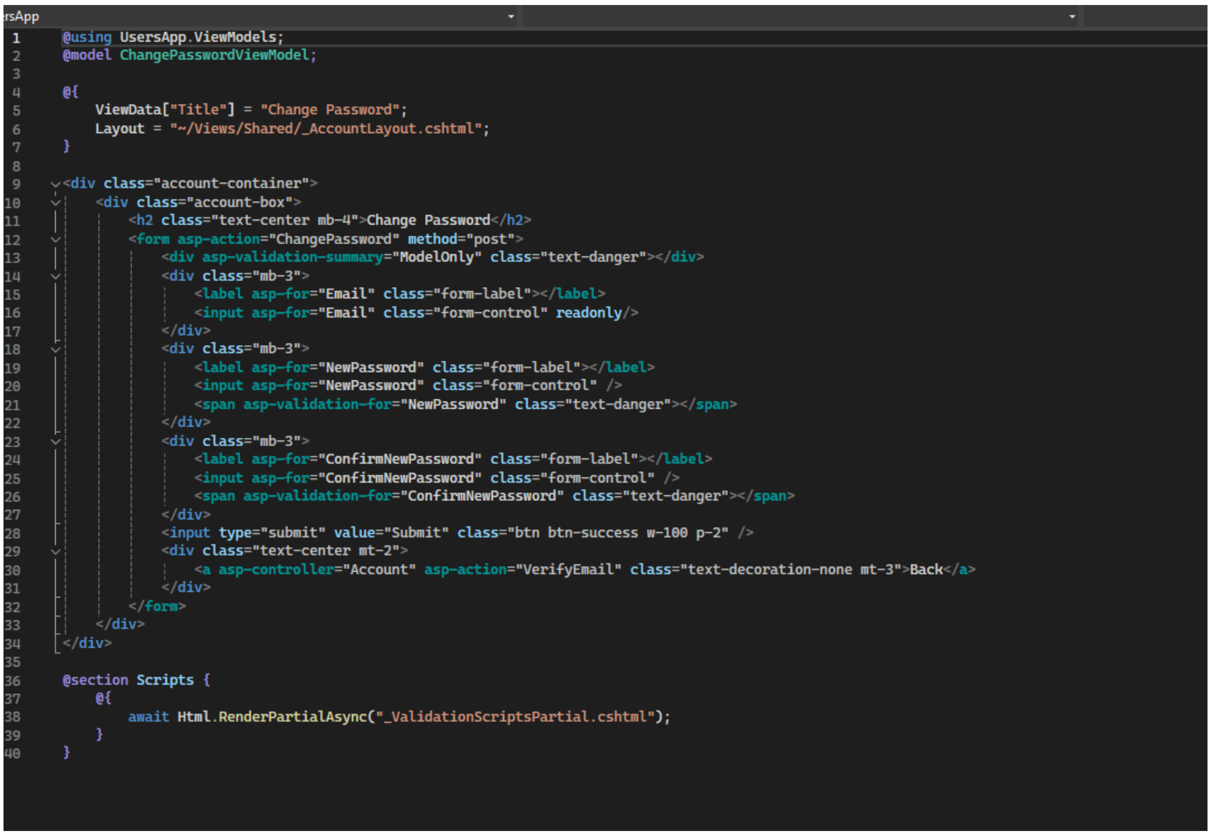
```
@using UsersApp.ViewModels;
@model VerifyEmailViewModel;

@{
    ViewData["Title"] = "Verify Email";
    Layout = "~/Views/Shared/_AccountLayout.cshtml";
}

<div class="account-container">
    <div class="account-box">
        <h2 class="text-center mb-4">Verify Email</h2>
        <form asp-action="VerifyEmail" method="post">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="mb-3">
                <label asp-for="Email" class="form-label"></label>
                <input asp-for="Email" class="form-control" />
                <span asp-validation-for="Email" class="text-danger"></span>
            </div>
            <input type="submit" value="Verify" class="btn btn-success w-100 p-2" />
            <div class="text-center mt-2">
                <a asp-controller="Home" asp-action="Index" class="text-decoration-none mt-3">Back</a>
            </div>
        </form>
    </div>
</div>

@section Scripts {
    @{
        await Html.RenderPartialAsync("_ValidationScriptsPartial.cshtml");
    }
}
```

30. Now time to create change password page



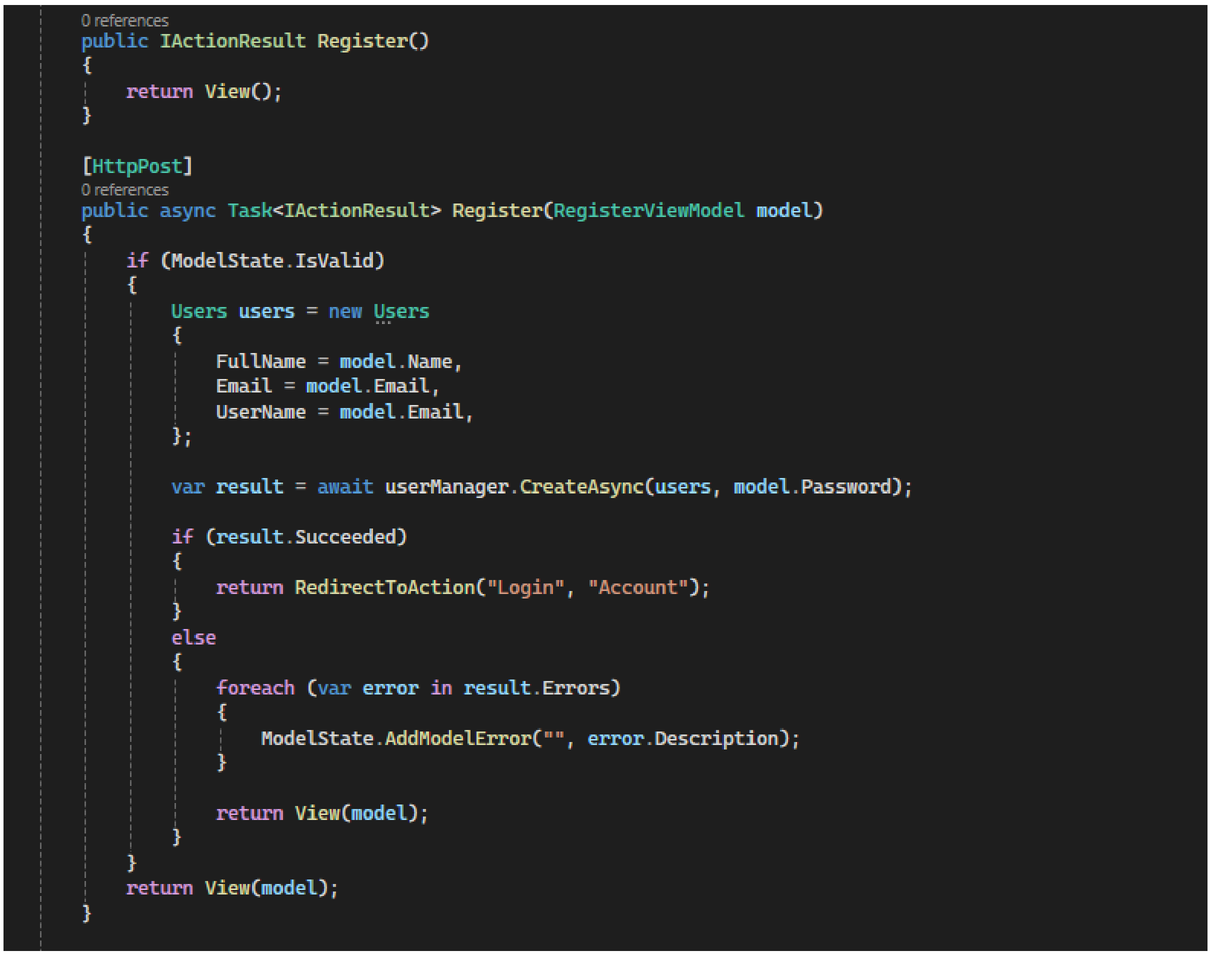
```
0 references
public IActionResult ChangePassword(string username)
{
    if (string.IsNullOrEmpty(username))
    {
        return RedirectToAction("VerifyEmail", "Account");
    }
    return View(new ChangePasswordViewModel { Email= username });
}
```

and view page



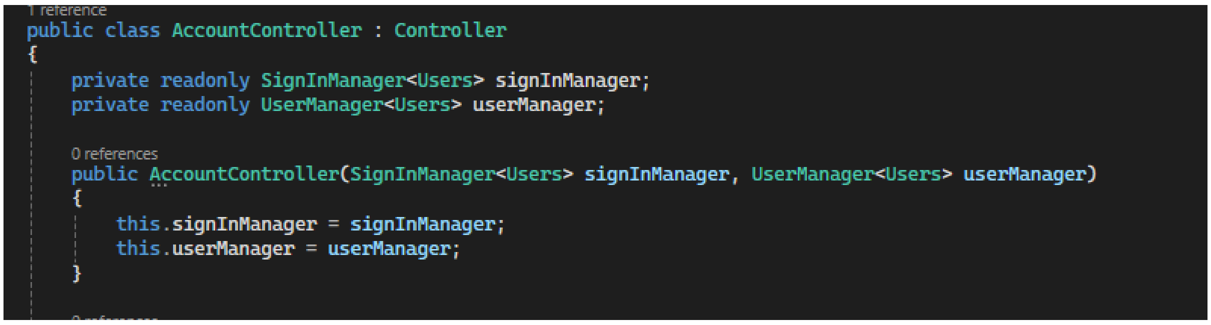
```
@using UsersApp.ViewModels;
@model ChangePasswordViewModel;

@{
    ViewData["Title"] = "Change Password";
    Layout = "~/Views/Shared/_AccountLayout.cshtml";
}

<div class="account-container">
    <div class="account-box">
        <h2 class="text-center mb-4">Change Password</h2>
        <form asp-action="ChangePassword" method="post">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="mb-3">
                <label asp-for="Email" class="form-label"></label>
                <input asp-for="Email" class="form-control" readonly/>
            </div>
            <div class="mb-3">
                <label asp-for="NewPassword" class="form-label"></label>
                <input asp-for="NewPassword" class="form-control" />
                <span asp-validation-for="NewPassword" class="text-danger"></span>
            </div>
            <div class="mb-3">
                <label asp-for="ConfirmNewPassword" class="form-label"></label>
                <input asp-for="ConfirmNewPassword" class="form-control" />
                <span asp-validation-for="ConfirmNewPassword" class="text-danger"></span>
            </div>
            <input type="submit" value="Submit" class="btn btn-success w-100 p-2" />
            <div class="text-center mt-2">
                <a asp-controller="Account" asp-action="VerifyEmail" class="text-decoration-none mt-3">Back</a>
            </div>
        </form>
    </div>
</div>

@section Scripts {
    @{
        await Html.RenderPartialAsync("_ValidationScriptsPartial.cshtml");
    }
}
```

31. Now time to register page data save in database



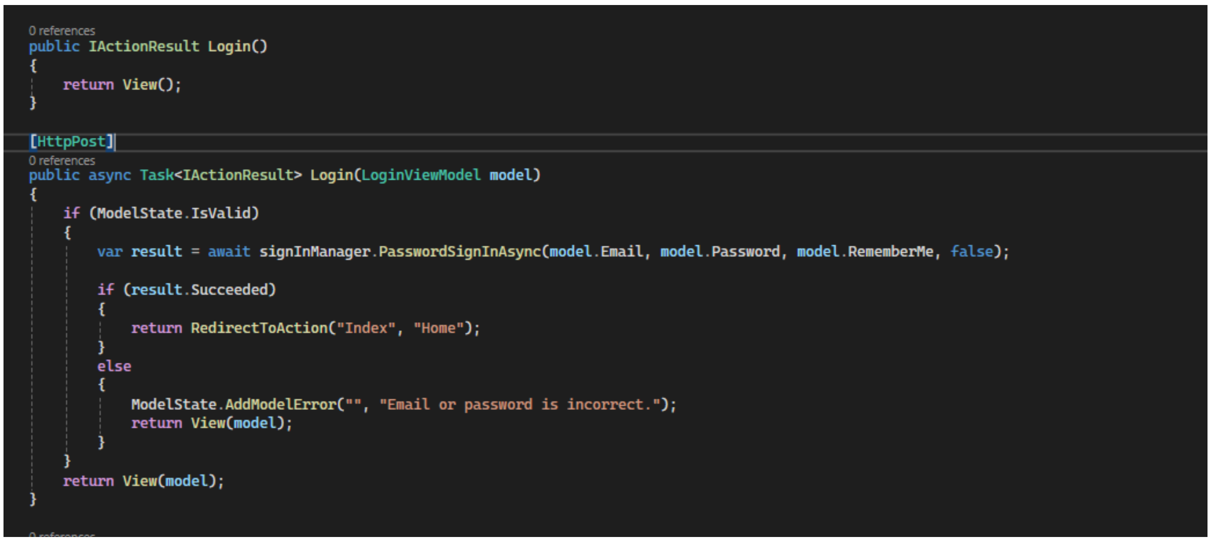
```
0 references
public IActionResult Register()
{
    return View();
}

[HttpPost]
0 references
public async Task<IActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        Users users = new Users
        {
            FullName = model.Name,
            Email = model.Email,
            UserName = model.Email,
        };

        var result = await userManager.CreateAsync(users, model.Password);

        if (result.Succeeded)
        {
            return RedirectToAction("Login", "Account");
        }
        else
        {
            foreach (var error in result.Errors)
            {
                ModelState.AddModelError("", error.Description);
            }

            return View(model);
        }
    }
    return View(model);
}
```

32. But before create



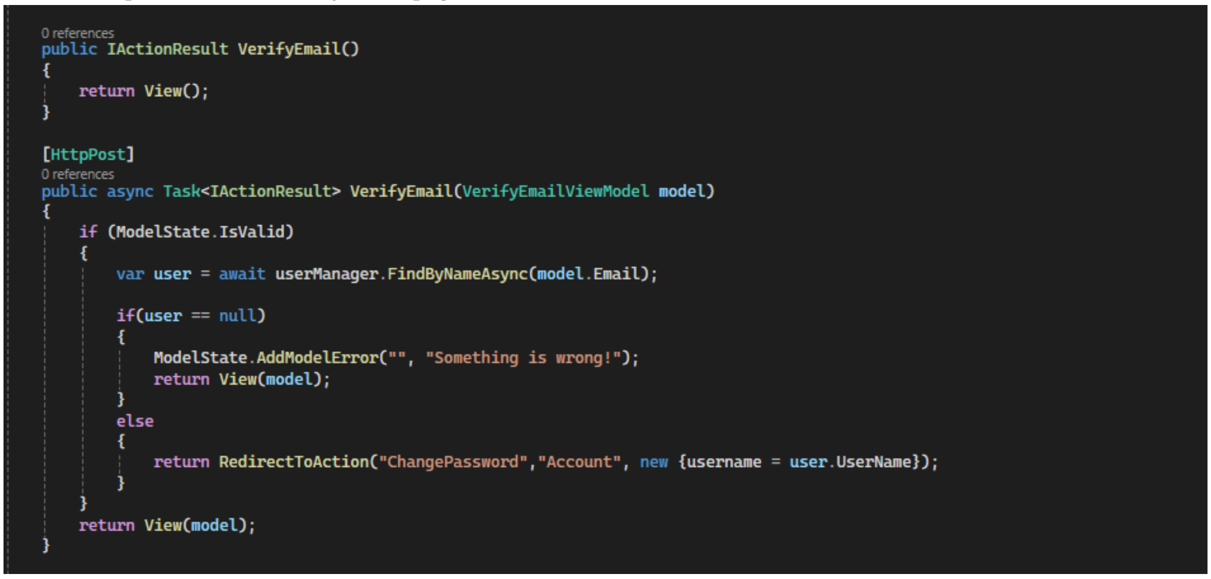
```
1 reference
public class AccountController : Controller
{
    private readonly SignInManager<Users> signInManager;
    private readonly UserManager<Users> userManager;

    0 references
    public AccountController(SignInManager<Users> signInManager, UserManager<Users> userManager)
    {
        this.signInManager = signInManager;
        this.userManager = userManager;
    }

    0 references
```

33. Now create post method of login page



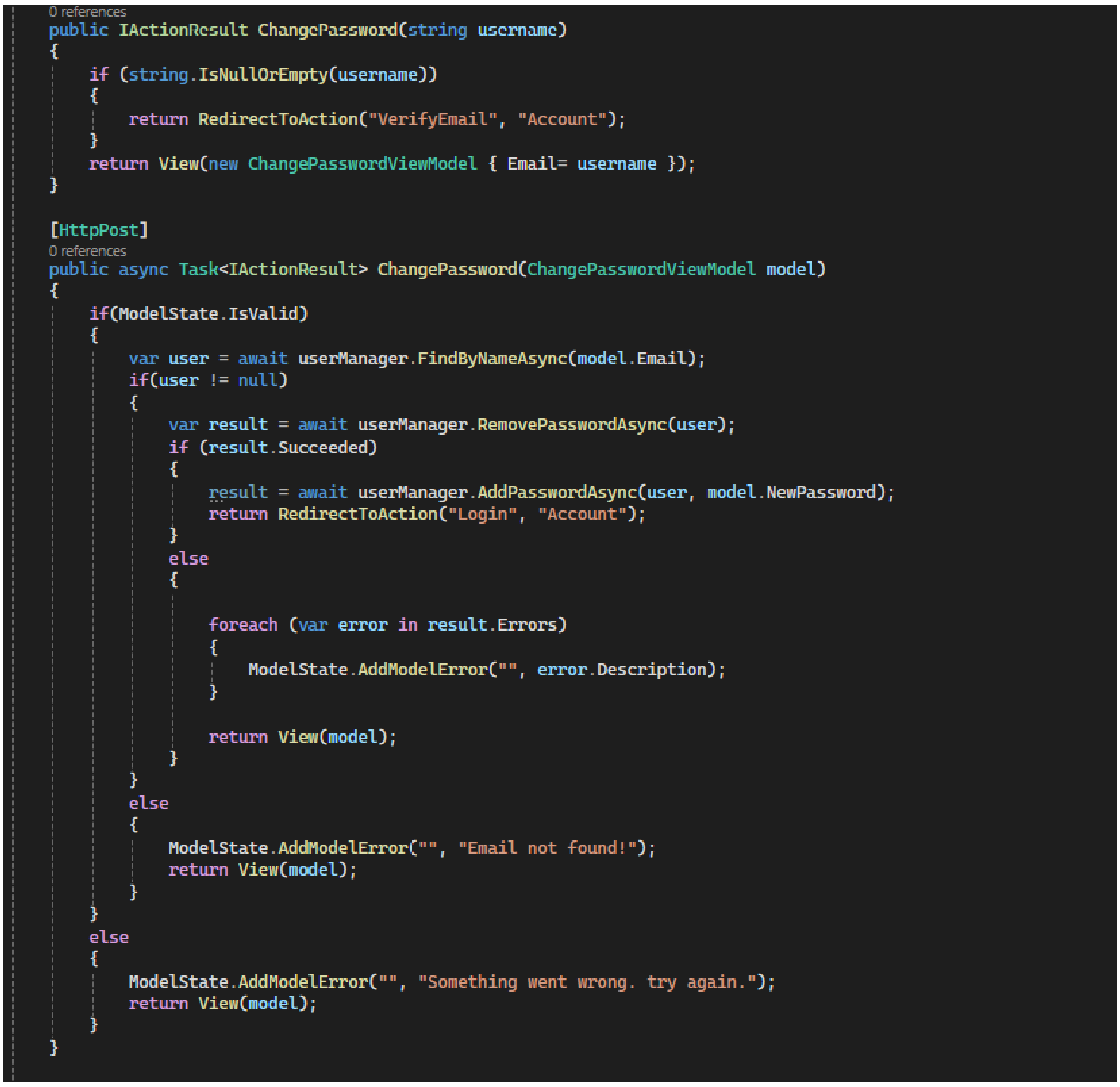
```
0 references
public IActionResult Login()
{
    return View();
}

[HttpPost]
0 references
public async Task<IActionResult> Login(LoginViewModel model)
{
    if (ModelState.IsValid)
    {
        var result = await signInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe, false);

        if (result.Succeeded)
        {
            return RedirectToAction("Index", "Home");
        }
        else
        {
            ModelState.AddModelError("", "Email or password is incorrect.");
            return View(model);
        }
    }
    return View(model);
}

0 references
```

34. Now create post method of verifyemail page



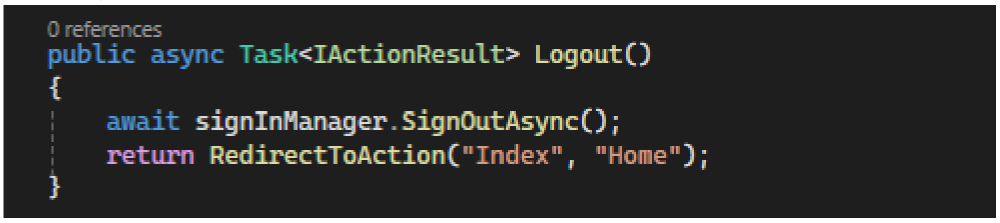
```
0 references
public IActionResult VerifyEmail()
{
    return View();
}

[HttpPost]
0 references
public async Task<IActionResult> VerifyEmail(VerifyEmailViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = await userManager.FindByNameAsync(model.Email);

        if(user == null)
        {
            ModelState.AddModelError("", "Something is wrong!");
            return View(model);
        }
        else
        {
            return RedirectToAction("ChangePassword","Account", new {username = user.UserName});
        }
    }
    return View(model);
}
```

35. Now create post method of change password page



```
0 references
public IActionResult ChangePassword(string username)
{
    if (string.IsNullOrEmpty(username))
    {
        return RedirectToAction("VerifyEmail", "Account");
    }
    return View(new ChangePasswordViewModel { Email= username });
}

[HttpPost]
0 references
public async Task<IActionResult> ChangePassword(ChangePasswordViewModel model)
{
    if(ModelState.IsValid)
    {
        var user = await userManager.FindByNameAsync(model.Email);
        if(user != null)
        {
            var result = await userManager.RemovePasswordAsync(user);
            if (result.Succeeded)
            {
                result = await userManager.AddPasswordAsync(user, model.NewPassword);
                return RedirectToAction("Login", "Account");
            }
            else
            {

                foreach (var error in result.Errors)
                {
                    ModelState.AddModelError("", error.Description);
                }

                return View(model);
            }
        }
        else
        {
            ModelState.AddModelError("", "Email not found!");
            return View(model);
        }
    }
    else
    {
        ModelState.AddModelError("", "Something went wrong. try again.");
        return View(model);
    }
}
```

36. Create logout



```
0 references
public async Task<IActionResult> Logout()
{
    await signInManager.SignOutAsync();
    return RedirectToAction("Index", "Home");
}
```