**Sectoin02 Identity Authentication**

**Notes: -**

**1-user Identity is another abstraction that you can hold up the claims identities and you can control on them**

**Create new project > Asp.Net Core web application > Empty Project**

**(manage NUGET packages and we install the following packages)**

**Entity Framework Core**

**Entity Framework Core. In memory**

**Microsoft.AspNetCore.Identity.EntityFrameworkCore**

**Steps:-**

**1-on the startup.cs we set the following code in configuration of using Identity in .net core**

**using IdentityExample.Data;**

**using Microsoft.AspNetCore.Builder;**

**using Microsoft.AspNetCore.Hosting;**

**using Microsoft.AspNetCore.Identity;**

**using Microsoft.EntityFrameworkCore;**

**using Microsoft.Extensions.DependencyInjection;**

**using Microsoft.Extensions.Hosting;**

**using System;**

**namespace IdentityExample{**

**public class Startup{**

**// This method gets called by the runtime. Use this method to add services to the container.**

**// For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?LinkID=398940**

**public void ConfigureServices(IServiceCollection services){**

**//Entity Framework Part**

**//to inject AppDbContext on the whole application to initialize database on the memory**

**services.AddDbContext<AppDbContext>(config =>{config.UseInMemoryDatabase("Memory");});**

**//Identity bridge Part(used to generate repositories that is collection of abstraction methods)**

**//AddDefaultTokenProviders: used to generate Token provider to generate Tokens for reset passwords**

**//AddEntityFrameworkStores: used to link the Identity layer with database to communicate**

**//AddIdentity : registers the services**

**services.AddIdentity<IdentityUser, IdentityRole>()**

**.AddEntityFrameworkStores<AppDbContext>()**

**.AddDefaultTokenProviders();**

**services.ConfigureApplicationCookie(options =>{**

**options.AccessDeniedPath = "/Home/AccessDenied";**

**options.Cookie.Name = "IdentityCookie";**

**options.Cookie.HttpOnly = true;**

**options.ExpireTimeSpan = new TimeSpan(0, 15, 0);**

**options.LoginPath = "/Account/Login";**

**options.ReturnUrlParameter = "RedirectUrl";**

**options.LogoutPath = "/Account/Logout";});**

**//to apply controllers with views**

**services.AddControllersWithViews();}**

**// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.**

**public void Configure(IApplicationBuilder app, IWebHostEnvironment env){**

**if (env.IsDevelopment()){app.UseDeveloperExceptionPage();}**

**else{app.UseExceptionHandler("/Home/Error");**

**// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.**

**app.UseHsts();}**

**app.UseHttpsRedirection();**

**app.UseStaticFiles();**

**//we apply the routing middleware to activate routing to whicle endpoint we want**

**app.UseRouting();**

**//means how are you?**

**app.UseAuthentication();**

**//we have to make sure that the authorization middleware must set after the routing middleware**

**//we have also to inject the authenticaiton cookie to allow pass the autorization middleware**

**//means : are you allowed?**

**app.UseAuthorization();**

**app.UseEndpoints(endpoints =>{endpoints.MapDefaultControllerRoute();});}}}**

**2-on the AccountController we set the following code as below**

**using IdentityExample.Models;**

**using Microsoft.AspNetCore.Authorization;**

**using Microsoft.AspNetCore.Identity;**

**using Microsoft.AspNetCore.Mvc;**

**using System.Linq;**

**using System.Security.Claims;**

**using System.Threading.Tasks;**

**namespace IdentityExample.Controllers{**

**public class AccountController : Controller{**

**private readonly UserManager<IdentityUser> userManager;**

**private readonly SignInManager<IdentityUser> signInManager;**

**public AccountController(UserManager<IdentityUser> userManager,SignInManager<IdentityUser> signInManager){**

**this.userManager = userManager;**

**this.signInManager = signInManager;}**

**[HttpGet]**

**public IActionResult Register(){**

**var model = new UserRegistrationDto();**

**return View(model);}**

**[HttpPost]**

**public async Task<IActionResult> Register(UserRegistrationDto request){**

**if (ModelState.IsValid){**

**var userCheck = await userManager.FindByEmailAsync(request.Email);**

**if (userCheck == null){**

**var user = new IdentityUser{**

**UserName = request.Email,**

**NormalizedUserName = request.Email,**

**Email = request.Email,**

**PhoneNumber = request.PhoneNumber,**

**EmailConfirmed = false,**

**PhoneNumberConfirmed = false,};**

**var result = await userManager.CreateAsync(user, request.Password);**

**if (result.Succeeded){return RedirectToAction("Login");}**

**else{**

**if (result.Errors.Count() > 0){**

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

**return View(request);}}**

**else{**

**ModelState.AddModelError("message", "Email already exists.");**

**return View(request);}}**

**return View(request);}**

**[HttpGet]**

**public IActionResult Login(){**

**var model = new UserLoginDto();**

**return View(model);}**

**[HttpPost]**

**public async Task<IActionResult> Login(UserLoginDto model){**

**if (ModelState.IsValid){**

**var user = await userManager.FindByEmailAsync(model.Email);**

**if (user != null && !user.EmailConfirmed){**

**ModelState.AddModelError("message", "Email not confirmed yet");**

**return View(model);}**

**if (await userManager.CheckPasswordAsync(user, model.Password) == false){**

**ModelState.AddModelError("message", "Invalid credentials");**

**return View(model);}**

**var result = await signInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe, true);**

**if (result.Succeeded){**

**await userManager.AddClaimAsync(user, new Claim("UserRole", "Admin"));**

**return RedirectToAction("Dashboard");}**

**else if (result.IsLockedOut){return View("AccountLocked");}**

**else{**

**ModelState.AddModelError("message", "Invalid login attempt");**

**return View(model);}}**

**return View(model);}**

**[Authorize(Roles ="Admin,admin")]**

**public IActionResult Dashboard(){return View();}**

**public async Task<IActionResult> Logout(){**

**await signInManager.SignOutAsync();**

**return RedirectToAction("login", "account");}}}**

**3-on the HomeController we set the following code as below**

**using Microsoft.AspNetCore.Authorization;**

**using Microsoft.AspNetCore.Identity;**

**using Microsoft.AspNetCore.Mvc;**

**namespace Basics.Controllers{**

**public class HomeController : Controller{**

**//used to handle user information such as get , update , delete**

**private readonly UserManager<IdentityUser> \_userManager;**

**private readonly SignInManager<IdentityUser> \_signInManager;**

**public HomeController(UserManager<IdentityUser> userManager,SignInManager<IdentityUser> signInManager){**

**\_userManager = userManager;**

**\_signInManager = signInManager;}**

**public IActionResult Index(){return View();}**

**public IActionResult AccessDenied(){return View();}**

**//mainly to used guard the action**

**[Authorize(Roles = "UserRole")]**

**public IActionResult Secret(){return View();}}}**

**4-on the Login.cshtml page we set the following code**

**@model IdentityExample.Models.UserLoginDto**

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

**<div class="container">**

**<h1>Login</h1>**

**@Html.ValidationSummary()**

**<form asp-action="Login" asp-controller="Account" method="post">**

**<div class="logindiv">**

**<div class="form-group col-sm-12">**

**<label asp-for="@Model.Email" class="col-lg-3 control-label"></label>**

**<br>**

**<div class="col-lg-9">**

**@Html.TextBoxFor(m => m.Email, new { placeholder = "Email", @class = "form-control" })**

**@Html.ValidationMessageFor(m => m.Email)</div></div>**

**<div class="form-group col-sm-12">**

**<label asp-for="@Model.Password" class="col-lg-3 control-label"></label><br>**

**<div class="col-lg-9">**

**@Html.PasswordFor(m => m.Password, new { placeholder = "Password", @class = "form-control" })**

**@Html.ValidationMessageFor(m => m.Password)</div></div>**

**<div class="form-group">**

**<div class="col-sm-12 btn-submit">**

**<button type="submit" class="btn btn-success">Login</button>**

**<a asp-action="Register" asp-controller="account">Don't have an account?Click to Register</a>**

**</div></div></div>**

**</form></div>**

**5-on the Register.cshtml we set the following code**

**@model IdentityExample.Models.UserRegistrationDto**

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

**<div class="container">**

**<h1>Register</h1>**

**@Html.ValidationSummary()**

**<form asp-action="Register" asp-controller="Account" method="post">**

**<div class="registerdiv">**

**<div class="form-group col-sm-12">**

**<label asp-for="@Model.Name" class="col-lg-3 control-label"></label><br>**

**<div class="col-lg-9">**

**@Html.TextBoxFor(m => m.Name, new { placeholder = "First Name", @class = "form-control" })**

**@Html.ValidationMessageFor(m => m.Name)</div></div>**

**<div class="form-group col-sm-12">**

**<label asp-for="@Model.Email" class="col-lg-3 control-label"></label><br>**

**<div class="col-lg-9">**

**@Html.TextBoxFor(m => m.Email, new { placeholder = "Email", @class = "form-control" })**

**@Html.ValidationMessageFor(m => m.Email)</div></div>**

**<div class="form-group col-sm-12">**

**<label asp-for="@Model.PhoneNumber" class="col-lg-3 control-label"></label><br>**

**<div class="col-lg-9">**

**@Html.TextBoxFor(m => m.PhoneNumber, new { placeholder = "Phone Number", @class = "form-control" })@Html.ValidationMessageFor(m => m.PhoneNumber)</div></div>**

**<div class="form-group col-sm-12">**

**<label asp-for="@Model.Password" class="col-lg-3 control-label"></label><br>**

**<div class="col-lg-9">**

**@Html.PasswordFor(m => m.Password, new { placeholder = "Password", @class = "form-control" })**

**@Html.ValidationMessageFor(m => m.Password)</div></div>**

**<div class="form-group col-sm-12">**

**<label asp-for="@Model.ConfirmPassword" class="col-lg-3 control-label"></label><br>**

**<div class="col-lg-9">**

**@Html.PasswordFor(m => m.ConfirmPassword, new { placeholder = "Confirm Password", @class = "form-control" })**

**@Html.ValidationMessageFor(m => m.ConfirmPassword)</div></div>**

**<div class="form-group">**

**<div class="col-sm-12 btn-submit">**

**<button type="submit" class="btn btn-success">Sign Up</button></div></div></div></form></div>**

**6-in the LoginPartial.cshtml we set the following code as below**

**(This partial view will be use in the \_layout as menu as below)**

**@using Microsoft.AspNetCore.Identity**

**@inject SignInManager<IdentityUser> SignInManager**

**@inject UserManager<IdentityUser> UserManager**

**<ul class="navbar-nav">**

**@if (SignInManager.IsSignedIn(User)){**

**<li class="nav-item">**

**<a class="nav-link text-dark" title="Manage">Hello @User.Identity.Name!</a></li>**

**<li class="nav-item">**

**<a class="nav-link btn btn-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="Register">Register</a>**

**</li>**

**<li class="nav-item">**

**<a class="nav-link text-dark" asp-controller="account" asp-action="login">Login</a>**

**</li>}**

**</ul>**