Program.cs:

using Microsoft.AspNetCore.Identity;

using Microsoft.EntityFrameworkCore;

using TaskManagementPlatform.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddDbContext<ApplicationDbContext>(options =>

options.UseInMemoryDatabase("TaskManagementDB"));

builder.Services.AddIdentity<ApplicationUser, IdentityRole>(options =>

{

options.Password.RequiredLength = 8;

options.Password.RequireNonAlphanumeric = false;

options.Password.RequireUppercase = true;

options.Password.RequireLowercase = true;

options.Password.RequireDigit = true;

// Lockout settings

options.Lockout.DefaultLockoutTimeSpan = TimeSpan.FromMinutes(5);

options.Lockout.MaxFailedAccessAttempts = 5;

options.Lockout.AllowedForNewUsers = true;

// User settings

options.User.RequireUniqueEmail = true;

})

.AddEntityFrameworkStores<ApplicationDbContext>()

.AddDefaultTokenProviders();

builder.Services.ConfigureApplicationCookie(options =>

{

options.Cookie.HttpOnly = true;

options.Cookie.SecurePolicy = CookieSecurePolicy.Always;

options.ExpireTimeSpan = TimeSpan.FromMinutes(15);

options.SlidingExpiration = true;

options.LoginPath = "/Account/Login";

options.LogoutPath = "/Account/Logout";

options.AccessDeniedPath = "/Account/AccessDenied";

});

builder.Services.AddAuthorization(options =>

{

options.AddPolicy("RequireAdminRole", policy => policy.RequireRole("Admin"));

options.AddPolicy("RequireUserRole", policy => policy.RequireRole("User"));

options.AddPolicy("CanEditTask", policy => policy.RequireClaim("CanEditTask", "true"));

});

builder.Services.AddControllersWithViews(options =>

{

options.Filters.Add(new AutoValidateAntiforgeryTokenAttribute());

});

builder.Services.AddAntiforgery(options =>

{

options.HeaderName = "X-CSRF-TOKEN";

options.SuppressXFrameOptionsHeader = false;

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

// Seed the database with initial data

using (var scope = app.Services.CreateScope())

{

var roleManager = scope.ServiceProvider.GetRequiredService<RoleManager<IdentityRole>>();

var userManager = scope.ServiceProvider.GetRequiredService<UserManager<ApplicationUser>>();

// Create roles

if (!await roleManager.RoleExistsAsync("Admin"))

await roleManager.CreateAsync(new IdentityRole("Admin"));

if (!await roleManager.RoleExistsAsync("User"))

await roleManager.CreateAsync(new IdentityRole("User"));

// Create admin user

var adminUser = new ApplicationUser { UserName = "admin@example.com", Email = "admin@example.com" };

var result = await userManager.CreateAsync(adminUser, "Admin123!");

if (result.Succeeded)

{

await userManager.AddToRoleAsync(adminUser, "Admin");

await userManager.AddClaimAsync(adminUser, new System.Security.Claims.Claim("CanEditTask", "true"));

}

// Create regular user

var regularUser = new ApplicationUser { UserName = "user@example.com", Email = "user@example.com" };

result = await userManager.CreateAsync(regularUser, "User123!");

if (result.Succeeded)

{

await userManager.AddToRoleAsync(regularUser, "User");

await userManager.AddClaimAsync(regularUser, new System.Security.Claims.Claim("CanEditTask", "true"));

}

}

app.Run();

1.) ApplicationUser.cs:

using Microsoft.AspNetCore.Identity;

namespace TaskManagementPlatform.Models

{

public class ApplicationUser : IdentityUser

{

public string FirstName { get; set; } = string.Empty;

public string LastName { get; set; } = string.Empty;

}

}

**ApplicationDbContext.cs:** **using Microsoft.AspNetCore.Identity.EntityFrameworkCore;**

**using Microsoft.EntityFrameworkCore;**

**namespace TaskManagementPlatform.Models**

**{**

**public class ApplicationDbContext : IdentityDbContext<ApplicationUser>**

**{**

**public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)**

**: base(options)**

**{**

**}**

**public DbSet<Task> Tasks { get; set; }**

**protected override void OnModelCreating(ModelBuilder builder)**

**{**

**base.OnModelCreating(builder);**

**}**

**}**

**}**

-Models:

**1.)LoginViewModel.cs:**

**using System.ComponentModel.DataAnnotations;**

**namespace TaskManagementPlatform.Models**

**{**

**public class LoginViewModel**

**{**

**[Required]**

**[EmailAddress]**

**[Display(Name = "Email")]**

**public string Email { get; set; } = string.Empty;**

**[Required]**

**[DataType(DataType.Password)]**

**[Display(Name = "Password")]**

**public string Password { get; set; } = string.Empty;**

**[Display(Name = "Remember me?")]**

**public bool RememberMe { get; set; }**

**}**

**}**

**2.)RegisterViewModel.cs:**

**using System.ComponentModel.DataAnnotations;**

**namespace TaskManagementPlatform.Models**

**{**

**public class RegisterViewModel**

**{**

**[Required]**

**[Display(Name = "First Name")]**

**[StringLength(50, ErrorMessage = "The {0} must be at least {2} and at max {1} characters long.", MinimumLength = 2)]**

**public string FirstName { get; set; } = string.Empty;**

**[Required]**

**[Display(Name = "Last Name")]**

**[StringLength(50, ErrorMessage = "The {0} must be at least {2} and at max {1} characters long.", MinimumLength = 2)]**

**public string LastName { get; set; } = string.Empty;**

**[Required]**

**[EmailAddress]**

**[Display(Name = "Email")]**

**public string Email { get; set; } = string.Empty;**

**[Required]**

**[StringLength(100, ErrorMessage = "The {0} must be at least {2} and at max {1} characters long.", MinimumLength = 8)]**

**[DataType(DataType.Password)]**

**[Display(Name = "Password")]**

**public string Password { get; set; } = string.Empty;**

**[DataType(DataType.Password)]**

**[Display(Name = "Confirm password")]**

**[Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]**

**public string ConfirmPassword { get; set; } = string.Empty;**

**}**

**}**

**3.)Task.cs:**

**using System.ComponentModel.DataAnnotations;**

**namespace TaskManagementPlatform.Models**

**{**

**public class Task**

**{**

**public int Id { get; set; }**

**[Required]**

**[StringLength(100, ErrorMessage = "Title cannot be longer than 100 characters.")]**

**public string Title { get; set; } = string.Empty;**

**[StringLength(500, ErrorMessage = "Description cannot be longer than 500 characters.")]**

**public string Description { get; set; } = string.Empty;**

**public DateTime CreatedDate { get; set; } = DateTime.Now;**

**public DateTime? DueDate { get; set; }**

**public bool IsCompleted { get; set; }**

**public string UserId { get; set; } = string.Empty;**

**public ApplicationUser? User { get; set; }**

**}**

**}**

**-Controllers:**

#### 1.) AccountController.cs:

#### using Microsoft.AspNetCore.Authentication;

#### using Microsoft.AspNetCore.Authorization;

#### using Microsoft.AspNetCore.Identity;

#### using Microsoft.AspNetCore.Mvc;

#### using TaskManagementPlatform.Models;

#### namespace TaskManagementPlatform.Controllers

#### {

#### public class AccountController : Controller

#### {

#### private readonly UserManager<ApplicationUser> \_userManager;

#### private readonly SignInManager<ApplicationUser> \_signInManager;

#### private readonly ILogger<AccountController> \_logger;

#### public AccountController(

#### UserManager<ApplicationUser> userManager,

#### SignInManager<ApplicationUser> signInManager,

#### ILogger<AccountController> logger)

#### {

#### \_userManager = userManager;

#### \_signInManager = signInManager;

#### \_logger = logger;

#### }

#### [HttpGet]

#### [AllowAnonymous]

#### public IActionResult Register(string? returnUrl = null)

#### {

#### ViewData["ReturnUrl"] = returnUrl;

#### return View();

#### }

#### [HttpPost]

#### [AllowAnonymous]

#### [ValidateAntiForgeryToken]

#### public async Task<IActionResult> Register(RegisterViewModel model, string? returnUrl = null)

#### {

#### ViewData["ReturnUrl"] = returnUrl;

#### if (ModelState.IsValid)

#### {

#### var user = new ApplicationUser

#### {

#### UserName = model.Email,

#### Email = model.Email,

#### FirstName = model.FirstName,

#### LastName = model.LastName

#### };

#### 

#### var result = await \_userManager.CreateAsync(user, model.Password);

#### if (result.Succeeded)

#### {

#### \_logger.LogInformation("User created a new account with password.");

#### 

#### // Add user to the User role

#### await \_userManager.AddToRoleAsync(user, "User");

#### 

#### // Add CanEditTask claim

#### await \_userManager.AddClaimAsync(user, new System.Security.Claims.Claim("CanEditTask", "true"));

#### 

#### await \_signInManager.SignInAsync(user, isPersistent: false);

#### \_logger.LogInformation("User logged in after registration.");

#### return RedirectToLocal(returnUrl);

#### }

#### AddErrors(result);

#### }

#### return View(model);

#### }

#### [HttpGet]

#### [AllowAnonymous]

#### public IActionResult Login(string? returnUrl = null)

#### {

#### ViewData["ReturnUrl"] = returnUrl;

#### return View();

#### }

#### [HttpPost]

#### [AllowAnonymous]

#### [ValidateAntiForgeryToken]

#### public async Task<IActionResult> Login(LoginViewModel model, string? returnUrl = null)

#### {

#### ViewData["ReturnUrl"] = returnUrl;

#### if (ModelState.IsValid)

#### {

#### var result = await \_signInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe, lockoutOnFailure: false);

#### if (result.Succeeded)

#### {

#### \_logger.LogInformation("User logged in.");

#### return RedirectToLocal(returnUrl);

#### }

#### if (result.IsLockedOut)

#### {

#### \_logger.LogWarning("User account locked out.");

#### return RedirectToAction(nameof(Lockout));

#### }

#### else

#### {

#### ModelState.AddModelError(string.Empty, "Invalid login attempt.");

#### return View(model);

#### }

#### }

#### return View(model);

#### }

#### [HttpPost]

#### [ValidateAntiForgeryToken]

#### public async Task<IActionResult> Logout()

#### {

#### await \_signInManager.SignOutAsync();

#### \_logger.LogInformation("User logged out.");

#### return RedirectToAction(nameof(HomeController.Index), "Home");

#### }

#### [HttpGet]

#### [AllowAnonymous]

#### public IActionResult Lockout()

#### {

#### return View();

#### }

#### [HttpGet]

#### public IActionResult AccessDenied()

#### {

#### return View();

#### }

#### private void AddErrors(IdentityResult result)

#### {

#### foreach (var error in result.Errors)

#### {

#### ModelState.AddModelError(string.Empty, error.Description);

#### }

#### }

#### private IActionResult RedirectToLocal(string? returnUrl)

#### {

#### if (Url.IsLocalUrl(returnUrl))

#### {

#### return Redirect(returnUrl);

#### }

#### else

#### {

#### return RedirectToAction(nameof(HomeController.Index), "Home");

#### }

#### }

#### }

#### }

#### 2.) HomeController.cs: using Microsoft.AspNetCore.Authorization;

#### using Microsoft.AspNetCore.Mvc;

#### namespace TaskManagementPlatform.Controllers

#### {

#### public class HomeController : Controller

#### {

#### private readonly ILogger<HomeController> \_logger;

#### public HomeController(ILogger<HomeController> logger)

#### {

#### \_logger = logger;

#### }

#### public IActionResult Index()

#### {

#### return View();

#### }

#### [Authorize]

#### public IActionResult Dashboard()

#### {

#### return View();

#### }

#### [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

#### public IActionResult Error()

#### {

#### return View();

#### }

#### }

#### }

#### 3.)AdminController.cs:

#### using Microsoft.AspNetCore.Authorization;

#### using Microsoft.AspNetCore.Mvc;

#### using Microsoft.EntityFrameworkCore;

#### using TaskManagementPlatform.Models;

#### namespace TaskManagementPlatform.Controllers

#### {

#### [Authorize(Policy = "RequireAdminRole")]

#### public class AdminController : Controller

#### {

#### private readonly ApplicationDbContext \_context;

#### public AdminController(ApplicationDbContext context)

#### {

#### \_context = context;

#### }

#### public async Task<IActionResult> ManageTasks()

#### {

#### var tasks = await \_context.Tasks

#### .Include(t => t.User)

#### .ToListAsync();

#### 

#### return View(tasks);

#### }

#### }

#### }

#### 4.) UserController.cs: using Microsoft.AspNetCore.Authorization;

#### using Microsoft.AspNetCore.Identity;

#### using Microsoft.AspNetCore.Mvc;

#### using Microsoft.EntityFrameworkCore;

#### using TaskManagementPlatform.Models;

#### namespace TaskManagementPlatform.Controllers

#### {

#### [Authorize(Policy = "RequireUserRole")]

#### public class UserController : Controller

#### {

#### private readonly ApplicationDbContext \_context;

#### private readonly UserManager<ApplicationUser> \_userManager;

#### public UserController(ApplicationDbContext context, UserManager<ApplicationUser> userManager)

#### {

#### \_context = context;

#### \_userManager = userManager;

#### }

#### public async Task<IActionResult> TaskList()

#### {

#### var user = await \_userManager.GetUserAsync(User);

#### if (user == null)

#### {

#### return NotFound();

#### }

#### var tasks = await \_context.Tasks

#### .Where(t => t.UserId == user.Id)

#### .ToListAsync();

#### 

#### return View(tasks);

#### }

#### [Authorize(Policy = "CanEditTask")]

#### [HttpGet]

#### public IActionResult CreateTask()

#### {

#### return View();

#### }

#### [Authorize(Policy = "CanEditTask")]

#### [HttpPost]

#### [ValidateAntiForgeryToken]

#### public async Task<IActionResult> CreateTask(Task task)

#### {

#### if (ModelState.IsValid)

#### {

#### var user = await \_userManager.GetUserAsync(User);

#### if (user == null)

#### {

#### return NotFound();

#### }

#### task.UserId = user.Id;

#### \_context.Tasks.Add(task);

#### await \_context.SaveChangesAsync();

#### 

#### return RedirectToAction(nameof(TaskList));

#### }

#### 

#### return View(task);

#### }

#### }

#### }

#### -VIEW:

#### 1.) Shared/\_Layout.cshtml: <!DOCTYPE html>

#### <html lang="en">

#### <head>

#### <meta charset="utf-8" />

#### <meta name="viewport" content="width=device-width, initial-scale=1.0" />

#### <title>@ViewData["Title"] - Task Management Platform</title>

#### <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

#### <style>

#### .navbar-brand {

#### font-weight: bold;

#### }

#### .footer {

#### position: absolute;

#### bottom: 0;

#### width: 100%;

#### height: 60px;

#### line-height: 60px;

#### background-color: #f5f5f5;

#### }

#### </style>

#### @RenderSection("Styles", required: false)

#### </head>

#### <body>

#### <header>

#### <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">

#### <div class="container">

#### <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">Task Management Platform</a>

#### <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse" aria-controls="navbarSupportedContent"

#### aria-expanded="false" aria-label="Toggle navigation">

#### <span class="navbar-toggler-icon"></span>

#### </button>

#### <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">

#### <ul class="navbar-nav flex-grow-1">

#### @if (User.Identity.IsAuthenticated)

#### {

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

#### <a class="nav-link text-dark" asp-controller="Home" asp-action="Dashboard">Dashboard</a>

#### </li>

#### @if (User.IsInRole("Admin"))

#### {

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

#### <a class="nav-link text-dark" asp-controller="Admin" asp-action="ManageTasks">Manage Tasks</a>

#### </li>

#### }

#### @if (User.IsInRole("User"))

#### {

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

#### <a class="nav-link text-dark" asp-controller="User" asp-action="TaskList">My Tasks</a>

#### </li>

#### }

#### }

#### </ul>

#### <partial name="\_LoginPartial" />

#### </div>

#### </div>

#### </nav>

#### </header>

#### <div class="container">

#### <main role="main" class="pb-3">

#### @RenderBody()

#### </main>

#### </div>

#### <footer class="border-top footer text-muted">

#### <div class="container">

#### &copy; 2023 - Task Management Platform - <a asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>

#### </div>

#### </footer>

#### <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

#### @RenderSection("Scripts", required: false)

#### </body>

#### </html>

**:**

-