Models:

1.)ApplicationUser.cs

```
using Microsoft.AspNetCore.Identity;


namespace SecureTaskApp.Models
{
    public class ApplicationUser : IdentityUser
    {
    }
}
```

2.)TaskItem.cs

```
using System.ComponentModel.DataAnnotations;


namespace SecureTaskApp.Models
{
    public class TaskItem
    {
        public int Id { get; set; }


        [Required, StringLength(200)]
        public string Title { get; set; }


        [Required, StringLength(500)]
        public string Description { get; set; }
```

```
        public string UserId { get; set; } // Link to ApplicationUser
    }
}
```

-Db Context File:

Application Db Context.cs:

```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using SecureTaskApp.Models;

namespace SecureTaskApp.Data
{
    public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
    {
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
            : base(options) { }

        public DbSet<TaskItem> Tasks { get; set; }
    }
}
```

Program.cs

```
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using SecureTaskApp.Data;
```

```
using SecureTaskApp.Models;

var builder = WebApplication.CreateBuilder(args);

// Database (SQLite for simplicity)
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlite("Data Source=securetaskapp.db"));

// Identity
builder.Services.AddIdentity<ApplicationUser, IdentityRole>(options =>
{
    options.Password.RequireDigit = true;
    options.Password.RequiredLength = 6;
    options.Lockout.DefaultLockoutTimeSpan = TimeSpan.FromMinutes(5);
})
.AddEntityFrameworkStores<ApplicationDbContext>()
.AddDefaultTokenProviders();

// Session & Cookie settings
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";
});

builder.Services.AddControllersWithViews();

var app = builder.Build();

app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();

app.UseAuthentication();
app.UseAuthorization();

app.MapDefaultControllerRoute();

app.Run();
```

Controllers:

1.)Account Controller.cs

```
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using SecureTaskApp.Models;
using SecureTaskApp.ViewModels;
```

```
namespace SecureTaskApp.Controllers
{
    public class AccountController : Controller
    {
        private readonly UserManager<ApplicationUser> _userManager;
        private readonly SignInManager<ApplicationUser> _signInManager;
        private readonly RoleManager<IdentityRole> _roleManager;

        public AccountController(UserManager<ApplicationUser> userManager,
            SignInManager<ApplicationUser> signInManager,
            RoleManager<IdentityRole> roleManager)
        {
            _userManager = userManager;
            _signInManager = signInManager;
            _roleManager = roleManager;
        }

        [HttpGet]
        public IActionResult Register() => View();

        [HttpPost, ValidateAntiForgeryToken]
        public async Task<IActionResult> Register(RegisterViewModel model)
        {
            if (ModelState.IsValid)
```

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

        if (result.Succeeded)
        {
            // Assign default role "User"
            if (!await _roleManager.RoleExistsAsync("User"))
                await _roleManager.CreateAsync(new IdentityRole("User"));

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

            await _signInManager.SignInAsync(user, isPersistent: false);
            return RedirectToAction("TaskList", "User");
        }
        foreach (var error in result.Errors)
            ModelState.AddModelError("", error.Description);
    }
    return View(model);
}

[HttpGet]
public IActionResult Login() => View();

[HttpPost, ValidateAntiForgeryToken]
```

```
    public async Task<IActionResult> Login(LoginViewModel model)
    {
        if (ModelState.IsValid)
        {
            var result = await _signInManager.PasswordSignInAsync(model.Email, model.Password, false,
false);

            if (result.Succeeded)
            {
                var user = await _userManager.FindByEmailAsync(model.Email);
                if (await _userManager.IsInRoleAsync(user, "Admin"))
                    return RedirectToAction("ManageTasks", "Admin");

                return RedirectToAction("TaskList", "User");
            }
            ModelState.AddModelError("", "Invalid login attempt.");
        }
        return View(model);
    }

    [HttpPost, ValidateAntiForgeryToken]
    public async Task<IActionResult> Logout()
    {
        await _signInManager.SignOutAsync();
        return RedirectToAction("Login");
    }
```

```
    }
}
```

2.)Admin Controller.cs:

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace SecureTaskApp.Controllers
{
    [Authorize(Roles = "Admin")]
    public class AdminController : Controller
    {
        public IActionResult ManageTasks()
        {
            return View();
        }
    }
}
```

3.)Uer Controller.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace SecureTaskApp.Controllers
{
    [Authorize(Roles = "User")]
    public class UserController : Controller
```

```
    {
        public IActionResult TaskList()
        {
            return View();
        }
    }
}
```

View Models

1.)RegisterViewModel.cs:

```
using System.ComponentModel.DataAnnotations;

namespace SecureTaskApp.ViewModels
{
    public class RegisterViewModel
    {
        [Required, EmailAddress]
        public string Email { get; set; }

        [Required, DataType(DataType.Password)]
        public string Password { get; set; }

        [Compare("Password", ErrorMessage = "Passwords do not match.")]
        public string ConfirmPassword { get; set; }
    }
```

```
}
```

2.)LoginViewModel.cs:

```
using System.ComponentModel.DataAnnotations;


namespace SecureTaskApp.ViewModels
{
    public class LoginViewModel
    {
        [Required, EmailAddress]
        public string Email { get; set; }


        [Required, DataType(DataType.Password)]
        public string Password { get; set; }
    }
}
```

Views:

Register.cshtml:

```
@model SecureTaskApp.ViewModels.RegisterViewModel
<form asp-action="Register" method="post">
    @Html.AntiForgeryToken()
    <input asp-for="Email" placeholder="Email" />
    <input asp-for="Password" type="password" placeholder="Password" />
    <input asp-for="ConfirmPassword" type="password" placeholder="Confirm Password" />
    <button type="submit">Register</button>
```

```
</form>
```

2.)Login.cshtml

```
@model SecureTaskApp.ViewModels.LoginViewModel

<form asp-action="Login" method="post">

    @Html.AntiForgeryToken()

    <input asp-for="Email" placeholder="Email" />

    <input asp-for="Password" type="password" placeholder="Password" />

    <button type="submit">Login</button>

</form>
```

3.)ManageTask.cshtml(Admin)

```
<h2>Admin - Manage Tasks</h2>

<p>Only admins can see this page.</p>
```

4)TaskList.cshtml(User):

```
<h2>User - Task List</h2>

<p>Only users can see this page.</p>
```

Register

Email

Password

Confirm Password

Register