Step-by-Step guide to implement Cookie-Based Security Using ASP.NET Core Identity in MVC Application

1. **Create Project**:
   - Use Visual Studio to create an ASP.NET Core MVC project
2. **Install Packages**:
   - Add required NuGet packages:

```
dotnet add package Microsoft.AspNetCore.Identity.EntityFrameworkCore
dotnet add package Microsoft.EntityFrameworkCore
dotnet add package Microsoft.EntityFrameworkCore.SqlServer
dotnet add package Microsoft.EntityFrameworkCore.Tools
```

3. **Update DbContext**:
   - Modify `ApplicationDbContext.cs` to include Identity tables.
   - Define required Entity classes

```
public class ApplicationDbContext :  IdentityDbContext<IdentityUser>
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext>
options) : base(options) { }
}
```

4. **Configure Identity in Program.cs**:
   - Register Identity services, EF Core, and customize cookie settings:

```
 // Register ApplicationDbContext
 var connectionString =
builder.Configuration.GetConnectionString("DefaultConnection");
 builder.Services.AddDbContext<ApplicationDbContext>(options =>
options.UseSqlServer(connectionString));

// Configure New User Registration constraints
builder.Services.AddIdentity<IdentityUser, IdentityRole>(options =>
{
    options.Password.RequiredLength = 8;
    options.User.RequireUniqueEmail = true;
})
.AddEntityFrameworkStores<ApplicationDbContext>()
.AddDefaultTokenProviders();
```

```
// Configure Cookie settings
builder.Services.ConfigureApplicationCookie(options =>
{
    options.Cookie.HttpOnly = true;
    options.ExpireTimeSpan = TimeSpan.FromMinutes(30);
    options.LoginPath = "/Account/Login";
});


// Add requried middleware
app.UseAuthentication();
app.UseAuthorization();
```

5. **Set Up Database**:
    - Ensure `appsettings.json` has a valid connection string
    - Run migrations to create Identity tables:

```
Add-Migration InitialCreate
Update-Database
```

6. **Add Account Models : Create RegisterViewModel and LoginViewModel. We can create Views based on these model classes**

```
  public class RegisterViewModel
 {
    [Required]
    public string UserName { get; set; }

    [Required]
    [EmailAddress]
    public string Email { get; set; }

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

    [DataType(DataType.Password)]
    [Compare("Password")]
    public string ConfirmPassword { get; set; }
 }


public class LoginViewModel
{
    [Required]
    public string UserName { get; set; }
    [Required]
    [DataType(DataType.Password)]
    public string Password { get; set; }
    public bool RememberMe { get; set; }
    public string? ReturnUrl { get; set; }
 }
```

7. **Add AccountController and Implement required action methods**

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using WebApplication51.Models;

namespace WebApplication51.Controllers
{
    public class AccountController : Controller
    {
        private readonly UserManager<IdentityUser> _userManager;
        private readonly SignInManager<IdentityUser> _signInManager;

        public AccountController(UserManager<IdentityUser> userManager,
SignInManager<IdentityUser> signInManager)
        {
            _userManager = userManager;
            _signInManager = signInManager;
        }

        [HttpGet]
        public IActionResult Register()
        {
            return View();
        }

        [HttpPost]
        [AllowAnonymous]
        public async Task<IActionResult> Register(RegisterViewModel model)
        {
            if (!ModelState.IsValid) return View(model);

            var user = new IdentityUser { UserName = model.UserName, Email =
model.Email };
            var result = await _userManager.CreateAsync(user,
model.Password);
            if (result.Succeeded)
            {
                await _signInManager.SignInAsync(user, isPersistent: false);
                return RedirectToAction("Index", "Home");
            }

            return View(model);
        }
        [HttpGet]
        public IActionResult Login()
        {
            return   View(new LoginViewModel()});
        }


        [HttpPost]
        public async Task<IActionResult> Login(LoginViewModel model)
        {
            if (!ModelState.IsValid) return View(model);

            var result = await
_signInManager.PasswordSignInAsync(model.UserName, model.Password,
model.RememberMe);
            if (result.Succeeded)
            {
                return RedirectToAction("Index", "Home");
            }

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

8. **Create views for AccountController: Login.cshtml and Register.cshtml**

    - Prepare the view files based on the ViewModel class (Refer **Step-6**)

9. **Secure MVC Routes**:

    - Add `[Authorize]` to controllers or actions to secure(e.g., `ProductsController`).

```
namespace WebApplication51.Controllers
 {
     [Authorize]
     public class ProductsController : Controller
     {
        // Add required action methods
     }
 }
```

10. **Update /Views/_ViewImports.cshtml**:

    - Modify `_ViewImports.cshtml` to inject SignInManager and UserManager:

```
@using WebApplication51
@using WebApplication51.Models
@using Microsoft.AspNetCore.Identity
```

```
    @inject SignInManager<IdentityUser> signInManager
    @inject UserManager<IdentityUser> userManager

    @addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

11. **Update Layout for Login Status**:

    - Modify `_Layout.cshtml` to show login/logout links in the nav bar based on authentication status:

    ```
    @if (SignInManager.IsSignedIn(User))
    {
        <a asp-area="Identity" asp-page="/Account/Logout">Logout</a>
    }
    else
    {
        <a asp-area="Identity" asp-page="/Account/Register">Register</a>
        <a asp-area="Identity" asp-page="/Account/Login">Login</a>
    }
    ```

    - Add Identity services to `_ViewImports.cshtml`.

12. **Test the Application**:

    - Run the app (`dotnet run` or F5).
    - Register a user, log in, access protected pages to verify functionality.

Summary Steps

Here are the concise steps to implement the complete Identity + EF Core example:

1. **Create a New MVC Project:**
2. **Add NuGet Packages:**
3. **Create DbContext and Configure Connection String**
4. **Configure Services (Program.cs):**
5. **Setup Database: Add EF Migration and Update Database:**
6. **Add Account Models (Create `RegisterViewModel` and `LoginViewModel`)**
7. **Add AccountController to Implement Register, Login, etc.**
8. **Create Views : Razor views for Register, Login**
9. **Protect Routes using [Authorize] attribute**
10. **Update /Views/_ViewImports.cshtml**
11. **Modify `_Layout.cshtml` to add login/logout links**
12. **Run and Test:**