# Project Files

**SecureShop.csproj**

<Project Sdk="Microsoft.NET.Sdk.Web">  
  
 <PropertyGroup>  
 <TargetFramework>net8.0</TargetFramework>  
 <Nullable>enable</Nullable>  
 <ImplicitUsings>enable</ImplicitUsings>  
 </PropertyGroup>  
  
 <ItemGroup>  
 <PackageReference Include="Microsoft.AspNetCore.Identity.EntityFrameworkCore" Version="8.0.8" />  
 <PackageReference Include="Microsoft.EntityFrameworkCore.Sqlite" Version="8.0.8" />  
 <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="8.0.8">  
 <PrivateAssets>all</PrivateAssets>  
 <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>  
 </PackageReference>  
 <PackageReference Include="Microsoft.AspNetCore.Mvc.Razor.RuntimeCompilation" Version="8.0.8" />  
 </ItemGroup>  
  
</Project>

**appsettings.json**

{  
 "ConnectionStrings": {  
 "DefaultConnection": "Data Source=secure\_shop.db"  
 },  
 "Logging": {  
 "LogLevel": {  
 "Default": "Information",  
 "Microsoft.AspNetCore": "Warning"  
 }  
 },  
 "AllowedHosts": "\*"  
}

**Program.cs**

using Microsoft.AspNetCore.Identity;  
using Microsoft.EntityFrameworkCore;  
using SecureShop.Data;  
using SecureShop.Models;  
using SecureShop.Services;  
  
var builder = WebApplication.CreateBuilder(args);  
  
  
builder.Services.AddDbContext<AppDbContext>(options =>  
 options.UseSqlite(builder.Configuration.GetConnectionString("DefaultConnection")));  
  
builder.Services.AddIdentity<ApplicationUser, IdentityRole>(options =>  
{  
 options.Password.RequiredLength = 8;  
 options.Password.RequireUppercase = True;  
 options.Password.RequireDigit = True;  
 options.Password.RequireNonAlphanumeric = True; // special char  
 options.Lockout.MaxFailedAccessAttempts = 5;  
 options.Lockout.DefaultLockoutTimeSpan = TimeSpan.FromMinutes(10);  
 options.User.RequireUniqueEmail = true;  
})  
.AddEntityFrameworkStores<AppDbContext>()  
.AddDefaultTokenProviders();  
  
builder.Services.ConfigureApplicationCookie(opt =>  
{  
 opt.LoginPath = "/Account/Login";  
 opt.AccessDeniedPath = "/Account/AccessDenied";  
 opt.SlidingExpiration = true;  
 opt.Cookie.HttpOnly = true;  
 opt.Cookie.SecurePolicy = CookieSecurePolicy.Always;  
 opt.Cookie.SameSite = SameSiteMode.Lax;  
});  
  
builder.Services.AddControllersWithViews()  
 .AddRazorRuntimeCompilation();

builder.Services.AddScoped<IEmailValidator, EmailValidator>();  
builder.Services.AddRateLimiter(options =>  
{  
 options.AddFixedWindowLimiter("loginLimiter", opt =>  
 {  
 opt.Window = TimeSpan.FromMinutes(1);  
 opt.PermitLimit = 10; // at most 10 login attempts/min per IP  
 opt.QueueLimit = 0;  
 });  
});  
  
var app = builder.Build();  
  
  
using (var scope = app.Services.CreateScope())  
{  
 var services = scope.ServiceProvider;  
 var db = services.GetRequiredService<AppDbContext>();  
 await db.Database.MigrateAsync();  
 await SeedData.InitializeAsync(services);  
}  
  
if (!app.Environment.IsDevelopment())  
{  
 app.UseExceptionHandler("/Home/Error");  
 app.UseHsts();  
}  
  
app.UseHttpsRedirection();  
app.UseStaticFiles();  
  
app.UseRouting();  
app.UseRateLimiter();   
  
app.UseAuthentication();  
app.UseAuthorization();  
  
app.MapControllerRoute(  
 name: "areas",  
 pattern: "{area:exists}/{controller=Dashboard}/{action=Index}/{id?}");  
  
app.MapControllerRoute(  
 name: "default",  
 pattern: "{controller=Products}/{action=Index}/{id?}");  
  
app.Run();

**Models/ApplicationUser.cs**

using Microsoft.AspNetCore.Identity;  
using System.ComponentModel.DataAnnotations;  
  
namespace SecureShop.Models  
{  
 public class ApplicationUser : IdentityUser  
 {  
   
 [StringLength(50)]  
 public string? FullName { get; set; }  
 }  
}

**Models/Product.cs**

using System.ComponentModel.DataAnnotations;  
  
namespace SecureShop.Models  
{  
 public class Product  
 {  
 public int Id { get; set; }  
  
 [Required, StringLength(80)]  
 public string Name { get; set; } = string.Empty;  
  
 [Required, StringLength(500)]  
 public string Description { get; set; } = string.Empty;   
  
 [Range(0.01, 1000000)]  
 public decimal Price { get; set; }  
 }  
}

**Models/Order.cs**

using System.ComponentModel.DataAnnotations;  
using System.ComponentModel.DataAnnotations.Schema;  
  
namespace SecureShop.Models  
{  
 public class Order  
 {  
 public int Id { get; set; }  
  
 [Required]  
 public string UserId { get; set; } = string.Empty;  
  
 [Required]  
 public int ProductId { get; set; }  
  
 [Range(1, 100)]  
 public int Quantity { get; set; }  
  
 public DateTime CreatedUtc { get; set; } = DateTime.UtcNow;  
 }  
}

**Models/Review.cs**

using System.ComponentModel.DataAnnotations;  
  
namespace SecureShop.Models  
{  
 public class Review  
 {  
 public int Id { get; set; }  
  
 [Required]  
 public int ProductId { get; set; }  
  
 [Required, StringLength(200)]  
 public string Title { get; set; } = string.Empty;  
  
 [Required, StringLength(1000)]  
 public string Content { get; set; } = string.Empty;  
  
 [Range(1,5)]  
 public int Rating { get; set; }  
 }  
}

**Data/AppDbContext.cs**

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;  
using Microsoft.EntityFrameworkCore;  
using SecureShop.Models;  
  
namespace SecureShop.Data  
{  
 public class AppDbContext : IdentityDbContext<ApplicationUser>  
 {  
 public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }  
  
 public DbSet<Product> Products => Set<Product>();  
 public DbSet<Order> Orders => Set<Order>();  
 public DbSet<Review> Reviews => Set<Review>();  
  
 protected override void OnModelCreating(ModelBuilder builder)  
 {  
 base.OnModelCreating(builder);  
  
 builder.Entity<Product>().Property(p => p.Price).HasColumnType("decimal(18,2)");  
 builder.Entity<Product>().HasData(  
 new Product { Id = 1, Name = "Wireless Mouse", Description = "Reliable 2.4GHz wireless mouse.", Price = 999.00m },  
 new Product { Id = 2, Name = "Mechanical Keyboard", Description = "Compact keyboard with tactile switches.", Price = 3499.00m }  
 );  
 }  
 }  
}

**Data/SeedData.cs**

using Microsoft.AspNetCore.Identity;  
using Microsoft.Extensions.DependencyInjection;  
using SecureShop.Models;  
  
namespace SecureShop.Data  
{  
 public static class SeedData  
 {  
 public static async Task InitializeAsync(IServiceProvider services)  
 {  
 var roleManager = services.GetRequiredService<RoleManager<IdentityRole>>();  
 var userManager = services.GetRequiredService<UserManager<ApplicationUser>>();  
  
 string[] roles = new[] { "Admin", "Customer" };  
 foreach (var role in roles)  
 {  
 if (!await roleManager.RoleExistsAsync(role))  
 {  
 await roleManager.CreateAsync(new IdentityRole(role));  
 }  
 }  
  
 var adminEmail = "admin@shop.test";  
 var admin = await userManager.FindByEmailAsync(adminEmail);  
 if (admin == null)  
 {  
 admin = new ApplicationUser  
 {  
 UserName = adminEmail,  
 Email = adminEmail,  
 EmailConfirmed = true,  
 FullName = "System Admin"  
 };  
 var result = await userManager.CreateAsync(admin, "Admin@123!");  
 if (result.Succeeded)  
 {  
 await userManager.AddToRoleAsync(admin, "Admin");  
 }  
 }  
 }  
 }  
}

**Services/EmailValidator.cs**

using System.ComponentModel.DataAnnotations;  
using System.Globalization;  
using System.Text.RegularExpressions;  
  
namespace SecureShop.Services  
{  
 public interface IEmailValidator  
 {  
 bool IsValidEmail(string email);  
 }  
  
 public class EmailValidator : IEmailValidator  
 {  
 private static readonly Regex \_emailRegex =  
 new(@"^[^@\s]+@[^@\s]+\.[^@\s]+$", RegexOptions.Compiled | RegexOptions.IgnoreCase);  
  
 public bool IsValidEmail(string email)  
 {  
 if (string.IsNullOrWhiteSpace(email)) return false;  
 email = email.Trim();  
 if (!\_emailRegex.IsMatch(email)) return false;  
  
 try  
 {  
   
 var atIndex = email.LastIndexOf('@');  
 var domain = email[(atIndex + 1)..];  
 var idn = new IdnMapping();  
 var asciiDomain = idn.GetAscii(domain);  
 }  
 catch  
 {  
 return false;  
 }  
 return true;  
 }  
 }  
}

**Controllers/AccountController.cs**

using Microsoft.AspNetCore.Authorization;  
using Microsoft.AspNetCore.Identity;  
using Microsoft.AspNetCore.Mvc;  
using SecureShop.Models;  
using SecureShop.Services;  
using System.ComponentModel.DataAnnotations;  
  
namespace SecureShop.Controllers  
{  
 public class RegisterViewModel  
 {  
 [Required, EmailAddress, StringLength(100)]  
 public string Email { get; set; } = string.Empty;  
  
 [Required, StringLength(50)]  
 public string FullName { get; set; } = string.Empty;  
  
 [Required, DataType(DataType.Password)]  
 [StringLength(100, MinimumLength = 8)]  
 [RegularExpression(@"^(?=.\*[A-Z])(?=.\*\d)(?=.\*[^A-Za-z0-9]).+$",  
 ErrorMessage = "Password must have uppercase, number, and special char.")]  
 public string Password { get; set; } = string.Empty;  
  
 [Required, DataType(DataType.Password), Compare(nameof(Password))]  
 public string ConfirmPassword { get; set; } = string.Empty;  
 }  
  
 public class LoginViewModel  
 {  
 [Required, EmailAddress]  
 public string Email { get; set; } = string.Empty;  
  
 [Required, DataType(DataType.Password)]  
 public string Password { get; set; } = string.Empty;  
  
 public bool RememberMe { get; set; }  
 }  
  
 public class AccountController : Controller  
 {  
 private readonly UserManager<ApplicationUser> \_userManager;  
 private readonly SignInManager<ApplicationUser> \_signInManager;  
 private readonly IEmailValidator \_emailValidator;  
  
 public AccountController(UserManager<ApplicationUser> userManager,  
 SignInManager<ApplicationUser> signInManager,  
 IEmailValidator emailValidator)  
 {  
 \_userManager = userManager;  
 \_signInManager = signInManager;  
 \_emailValidator = emailValidator;  
 }  
  
 [HttpGet]  
 public IActionResult Register() => View();  
  
 [HttpPost]  
 [ValidateAntiForgeryToken]  
 public async Task<IActionResult> Register(RegisterViewModel vm)  
 {  
 if (!\_emailValidator.IsValidEmail(vm.Email))  
 {  
 ModelState.AddModelError(nameof(vm.Email), "Invalid email address.");  
 }  
  
 if (!ModelState.IsValid) return View(vm);  
  
 var user = new ApplicationUser { UserName = vm.Email, Email = vm.Email, FullName = vm.FullName };  
  
 var result = await \_userManager.CreateAsync(user, vm.Password);  
 if (result.Succeeded)  
 {  
 await \_userManager.AddToRoleAsync(user, "Customer");  
 await \_signInManager.SignInAsync(user, isPersistent: false);  
 return RedirectToAction("Index", "Products");  
 }  
  
 foreach (var error in result.Errors)  
 ModelState.AddModelError("", error.Description);  
  
 return View(vm);  
 }  
  
 [HttpGet]  
 public IActionResult Login(string? returnUrl = null)  
 {  
 ViewData["ReturnUrl"] = returnUrl;  
 return View();  
 }  
  
 [HttpPost]  
 [ValidateAntiForgeryToken]  
 [EnableRateLimiting("loginLimiter")]  
 public async Task<IActionResult> Login(LoginViewModel vm, string? returnUrl = null)  
 {  
 if (!ModelState.IsValid) return View(vm);  
  
 var user = await \_userManager.FindByEmailAsync(vm.Email);  
 if (user == null)  
 {  
 ModelState.AddModelError("", "Invalid login attempt.");  
 return View(vm);  
 }  
  
 var result = await \_signInManager.PasswordSignInAsync(user, vm.Password, vm.RememberMe, lockoutOnFailure: true);  
 if (result.Succeeded)  
 {  
 if (!string.IsNullOrEmpty(returnUrl) && Url.IsLocalUrl(returnUrl))  
 return Redirect(returnUrl);  
 return RedirectToAction("Index", "Products");  
 }  
 if (result.IsLockedOut)  
 {  
 ModelState.AddModelError("", "Account locked due to multiple failed attempts. Try again later.");  
 return View(vm);  
 }  
  
 ModelState.AddModelError("", "Invalid login attempt.");  
 return View(vm);  
 }  
  
 [Authorize]  
 [HttpPost]  
 [ValidateAntiForgeryToken]  
 public async Task<IActionResult> Logout()  
 {  
 await \_signInManager.SignOutAsync();  
 return RedirectToAction("Login", "Account");  
 }  
  
 [HttpGet]  
 public IActionResult AccessDenied() => View();  
 }  
}

**Controllers/ProductsController.cs**

using Microsoft.AspNetCore.Authorization;  
using Microsoft.AspNetCore.Mvc;  
using Microsoft.EntityFrameworkCore;  
using SecureShop.Data;  
using SecureShop.Models;  
using System.ComponentModel.DataAnnotations;  
  
namespace SecureShop.Controllers  
{  
 public class ProductsController : Controller  
 {  
 private readonly AppDbContext \_db;  
 public ProductsController(AppDbContext db) => \_db = db;  
  
 [HttpGet]  
 public async Task<IActionResult> Index()  
 {  
 var products = await \_db.Products.AsNoTracking().ToListAsync();  
 return View(products);  
 }  
  
 [HttpGet]  
 public async Task<IActionResult> Details(int id)  
 {  
 var product = await \_db.Products.AsNoTracking().FirstOrDefaultAsync(p => p.Id == id);  
 if (product == null) return NotFound();  
  
 ViewBag.Reviews = await \_db.Reviews.AsNoTracking().Where(r => r.ProductId == id).ToListAsync();  
 return View(product);  
 }  
  
 public class ReviewVm  
 {  
 [Required] public int ProductId { get; set; }  
 [Required, StringLength(200)] public string Title { get; set; } = string.Empty;  
 [Required, StringLength(1000)] public string Content { get; set; } = string.Empty;  
 [Range(1,5)] public int Rating { get; set; }  
 }  
  
 [Authorize]  
 [HttpPost]  
 [ValidateAntiForgeryToken]  
 public async Task<IActionResult> AddReview(ReviewVm vm)  
 {  
 if (!ModelState.IsValid)  
 {  
 return RedirectToAction(nameof(Details), new { id = vm.ProductId });  
 }  
  
   
 var review = new Review  
 {  
 ProductId = vm.ProductId,  
 Title = vm.Title.Trim(),  
 Content = vm.Content.Trim(),  
 Rating = vm.Rating  
 };  
 \_db.Reviews.Add(review);  
 await \_db.SaveChangesAsync();  
 return RedirectToAction(nameof(Details), new { id = vm.ProductId });  
 }  
 }  
}

**Controllers/OrdersController.cs**

using Microsoft.AspNetCore.Authorization;  
using Microsoft.AspNetCore.Identity;  
using Microsoft.AspNetCore.Mvc;  
using Microsoft.EntityFrameworkCore;  
using SecureShop.Data;  
using SecureShop.Models;  
using System.ComponentModel.DataAnnotations;  
  
namespace SecureShop.Controllers  
{  
 [Authorize]  
 public class OrdersController : Controller  
 {  
 private readonly AppDbContext \_db;  
 private readonly UserManager<ApplicationUser> \_userManager;  
  
 public OrdersController(AppDbContext db, UserManager<ApplicationUser> userManager)  
 {  
 \_db = db;  
 \_userManager = userManager;  
 }  
  
 public class CreateOrderVm  
 {  
 [Required] public int ProductId { get; set; }  
 [Range(1,100)] public int Quantity { get; set; } = 1;  
 }  
  
 [HttpGet]  
 public async Task<IActionResult> Create(int productId)  
 {  
 var product = await \_db.Products.AsNoTracking().FirstOrDefaultAsync(p => p.Id == productId);  
 if (product == null) return NotFound();  
 ViewBag.Product = product;  
 return View(new CreateOrderVm { ProductId = productId });  
 }  
  
 [HttpPost]  
 [ValidateAntiForgeryToken]  
 public async Task<IActionResult> Create(CreateOrderVm vm)  
 {  
 if (!ModelState.IsValid)  
 {  
 var product = await \_db.Products.AsNoTracking().FirstOrDefaultAsync(p => p.Id == vm.ProductId);  
 ViewBag.Product = product;  
 return View(vm);  
 }  
  
 var user = await \_userManager.GetUserAsync(User);  
 if (user == null) return Challenge();  
  
 var order = new Order  
 {  
 ProductId = vm.ProductId,  
 Quantity = vm.Quantity,  
 UserId = user.Id  
 };  
 \_db.Orders.Add(order);  
 await \_db.SaveChangesAsync();  
 return RedirectToAction("Index", "Products");  
 }  
 }  
}

**Areas/Admin/Controllers/DashboardController.cs**

using Microsoft.AspNetCore.Authorization;  
using Microsoft.AspNetCore.Mvc;  
using Microsoft.EntityFrameworkCore;  
using SecureShop.Data;  
  
namespace SecureShop.Areas.Admin.Controllers  
{  
 [Area("Admin")]  
 [Authorize(Roles = "Admin")]  
 public class DashboardController : Controller  
 {  
 private readonly AppDbContext \_db;  
 public DashboardController(AppDbContext db) => \_db = db;  
  
 public async Task<IActionResult> Index()  
 {  
 var stats = new  
 {  
 Products = await \_db.Products.CountAsync(),  
 Orders = await \_db.Orders.CountAsync(),  
 Reviews = await \_db.Reviews.CountAsync()  
 };  
 return View(stats);  
 }  
 }  
}

**\_ViewImports.cshtml**

@addTagHelper \*, Microsoft.AspNetCore.Mvc.TagHelpers  
@using SecureShop.Models

**Views/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>SecureShop - @ViewData["Title"]</title>  
 <link rel="stylesheet" href="~/css/site.css" />  
</head>  
<body>  
 <header>  
 <nav>  
 <a asp-controller="Products" asp-action="Index">Products</a>  
 @if (User.Identity?.IsAuthenticated ?? false)  
 {  
 <span>Hello, @User.Identity!.Name</span>  
 <form asp-controller="Account" asp-action="Logout" method="post" style="display:inline;">  
 <button type="submit">Logout</button>  
 @Html.AntiForgeryToken()  
 </form>  
 @if (User.IsInRole("Admin"))  
 {  
 <a asp-area="Admin" asp-controller="Dashboard" asp-action="Index">Admin</a>  
 }  
 }  
 else  
 {  
 <a asp-controller="Account" asp-action="Login">Login</a>  
 <a asp-controller="Account" asp-action="Register">Register</a>  
 }  
 </nav>  
 </header>  
 <div class="container">  
 @RenderBody()  
 </div>  
 <partial name="\_ValidationScriptsPartial" />  
</body>  
</html>

**Views/Shared/\_ValidationScriptsPartial.cshtml**

<environment include="Development">  
 <script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>  
 <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-validate/1.19.5/jquery.validate.min.js"></script>  
 <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-validation-unobtrusive/3.2.12/jquery.validate.unobtrusive.min.js"></script>  
</environment>

**Views/Shared/Error.cshtml**

@{  
 Layout = "\_Layout";  
 ViewData["Title"] = "Error";  
}  
<h2>Something went wrong.</h2>  
<p>Please try again later.</p>

**Views/Account/Login.cshtml**

@model SecureShop.Controllers.LoginViewModel  
@{  
 Layout = "\_Layout";  
 ViewData["Title"] = "Login";  
}  
<h2>Login</h2>  
<form asp-action="Login" method="post">  
 <div>  
 <label asp-for="Email"></label>  
 <input asp-for="Email" />  
 <span asp-validation-for="Email"></span>  
 </div>  
 <div>  
 <label asp-for="Password"></label>  
 <input asp-for="Password" type="password" />  
 <span asp-validation-for="Password"></span>  
 </div>  
 <div>  
 <input asp-for="RememberMe" type="checkbox" /> Remember me  
 </div>  
 <button type="submit">Login</button>  
 @Html.AntiForgeryToken()  
</form>  
<div asp-validation-summary="All"></div>

**Views/Account/Register.cshtml**

@model SecureShop.Controllers.RegisterViewModel  
@{  
 Layout = "\_Layout";  
 ViewData["Title"] = "Register";  
}  
<h2>Register</h2>  
<form asp-action="Register" method="post" novalidate>  
 <div>  
 <label asp-for="Email"></label>  
 <input asp-for="Email" />  
 <span asp-validation-for="Email"></span>  
 </div>  
 <div>  
 <label asp-for="FullName"></label>  
 <input asp-for="FullName" />  
 <span asp-validation-for="FullName"></span>  
 </div>  
 <div>  
 <label asp-for="Password"></label>  
 <input asp-for="Password" type="password" />  
 <span asp-validation-for="Password"></span>  
 </div>  
 <div>  
 <label asp-for="ConfirmPassword"></label>  
 <input asp-for="ConfirmPassword" type="password" />  
 <span asp-validation-for="ConfirmPassword"></span>  
 </div>  
 <button type="submit">Register</button>  
 @Html.AntiForgeryToken()  
</form>  
<div asp-validation-summary="All"></div>

**Views/Products/Index.cshtml**

@model IEnumerable<SecureShop.Models.Product>  
@{  
 Layout = "\_Layout";  
 ViewData["Title"] = "Products";  
}  
<h2>Products</h2>  
<table>  
 <thead>  
 <tr>  
 <th>Name</th>  
 <th>Price</th>  
 <th></th>  
 </tr>  
 </thead>  
 <tbody>  
 @foreach (var p in Model)  
 {  
 <tr>  
 <td>@p.Name</td>  
 <td>@p.Price.ToString("C")</td>  
 <td><a asp-action="Details" asp-route-id="@p.Id">View</a></td>  
 </tr>  
 }  
 </tbody>  
</table>

**Views/Products/Details.cshtml**

@model SecureShop.Models.Product  
@{  
 Layout = "\_Layout";  
 ViewData["Title"] = "Details";  
 var reviews = (IEnumerable<SecureShop.Models.Review>)ViewBag.Reviews;  
}  
<h2>@Model.Name</h2>  
<p>@Model.Description</p> <!-- Razor encodes output by default -->  
<p><strong>Price:</strong> @Model.Price.ToString("C")</p>  
  
<a asp-controller="Orders" asp-action="Create" asp-route-productId="@Model.Id">Buy</a>  
  
<section>  
 <h3>Reviews</h3>  
 @if (reviews.Any())  
 {  
 <ul>  
 @foreach (var r in reviews)  
 {  
 <li>  
 <strong>@r.Title</strong> (@r.Rating/5)<br />  
 <span>@r.Content</span> <!-- encoded -->  
 </li>  
 }  
 </ul>  
 }  
 else  
 {  
 <p>No reviews yet.</p>  
 }  
  
 @if (User.Identity?.IsAuthenticated ?? false)  
 {  
 <h4>Add Review</h4>  
 <form asp-action="AddReview" method="post">  
 <input type="hidden" name="ProductId" value="@Model.Id" />  
 <div>  
 <label>Title</label>  
 <input name="Title" maxlength="200" required />  
 </div>  
 <div>  
 <label>Content</label>  
 <textarea name="Content" maxlength="1000" required></textarea>  
 </div>  
 <div>  
 <label>Rating</label>  
 <input name="Rating" type="number" min="1" max="5" value="5" required />  
 </div>  
 <button type="submit">Submit</button>  
 @Html.AntiForgeryToken()  
 </form>  
 }  
 else  
 {  
 <p><a asp-controller="Account" asp-action="Login">Login to add a review</a></p>  
 }  
</section>

**Views/Orders/Create.cshtml**

@model SecureShop.Controllers.OrdersController.CreateOrderVm  
@{  
 Layout = "\_Layout";  
 ViewData["Title"] = "Checkout";  
 var product = (SecureShop.Models.Product)ViewBag.Product;  
}  
<h2>Checkout: @product.Name</h2>  
<form asp-action="Create" method="post">  
 <input type="hidden" asp-for="ProductId" />  
 <div>  
 <label asp-for="Quantity"></label>  
 <input asp-for="Quantity" />  
 <span asp-validation-for="Quantity"></span>  
 </div>  
 <p>Total: @(product.Price \* Model.Quantity)</p>  
 <button type="submit">Place Order</button>  
 @Html.AntiForgeryToken()  
</form>

**Areas/Admin/Views/Dashboard/Index.cshtml**

@model dynamic  
@{  
 Layout = "\_Layout";  
 ViewData["Title"] = "Admin Dashboard";  
}  
<h2>Admin Dashboard</h2>  
<ul>  
 <li>Total Products: @Model.Products</li>  
 <li>Total Orders: @Model.Orders</li>  
 <li>Total Reviews: @Model.Reviews</li>  
</ul>

**Views/Account/AccessDenied.cshtml**

@{  
 Layout = "\_Layout";  
 ViewData["Title"] = "Access Denied";  
}  
<h2>Access Denied</h2>  
<p>You do not have permission to access this page.</p>

**wwwroot/css/site.css**

body { font-family: system-ui, -apple-system, Segoe UI, Roboto, Arial, sans-serif; margin: 0; }  
header nav { display:flex; gap:1rem; padding:1rem; background:#111; color:#fff; align-items:center; }  
header nav a, header nav span, header nav button { color:#fff; text-decoration:none; }  
.container { padding: 1rem; }  
table { border-collapse: collapse; width: 100%; }  
td, th { padding: .5rem; border-bottom: 1px solid #ddd; }  
form div { margin-bottom: .5rem; }