Project Report: Secure Online Shopping Platform in ASP.NET Core MVC

---

1. Introduction

Brief Description:
This project is a **basic online shopping platform** built with **ASP.NET Core MVC**. It allows users to **register, log in, view products, and make purchases**. The application follows **secure coding practices** to prevent common vulnerabilities such as **SQL Injection, Cross-Site Scripting (XSS), and CSRF attacks**.

Purpose:

- Ensure security with input validation and output encoding.
- Use **Entity Framework Core** (ORM) for safe database queries.
- Provide **role-based access** (Admin and Customer).

---

2. Tools and Technologies

- **ASP.NET Core MVC (7/8)**
- **Visual Studio 2022**
- **C#**
- **Entity Framework Core (EF Core)**
- **AntiForgeryToken** for CSRF prevention
- **HTTPS** enforcement
- **Identity Framework** for authentication and roles

---

3. Project Structure

Important Files & Folders:

- **Controllers**
  - AccountController.cs → Handles Registration, Login, Logout
  - ProductsController.cs → Displays products
  - AdminController.cs → Admin dashboard
- **Models**

    - ApplicationUser.cs → Extends IdentityUser (stores user details)
    - Product.cs → Product details (Name, Price, Description)
    - LoginViewModel.cs, RegisterViewModel.cs
- **Views**
    - Account/Login.cshtml
    - Account/Register.cshtml
    - Products/Index.cshtml (product listing)
    - Admin/Dashboard.cshtml
    - Shared/AccessDenied.cshtml
- **Program.cs / Startup.cs** → Configures Identity, EF Core, Middleware

---

4. Implementation Details

4.1 Secure Input Validation & Output Encoding

- **Validation**: DataAnnotations ([Required], [EmailAddress], [RegularExpression]) used in models.
- **SQL Injection Prevention**: EF Core parameterized queries.
- **XSS Prevention**: Razor automatically encodes @Model.Property in .cshtml.

```
public class RegisterViewModel {
  [Required, EmailAddress]
  public string Email { get; set; }

  [Required, MinLength(8)]
  [RegularExpression("^(?=.*[A-Z])(?=.*[0-9])(?=.*[@$!%*?&]).+$",
    ErrorMessage = "Password must contain uppercase, number, and special character")]
  public string Password { get; set; }
}
```

4.2 Authentication & Role-Based Authorization

- **Roles**: Admin, Customer
- **Authorization**: [Authorize(Roles = "Admin")] on AdminController.
- **Unauthorized Access**: Redirects to /Account/Login or AccessDenied.cshtml.

```
[Authorize(Roles = "Admin")]
public class AdminController : Controller {
   public IActionResult Dashboard() {
      return View();
   }
}
```

4.3 Security Measures

- **CSRF**: All forms use @Html.AntiForgeryToken() with [ValidateAntiForgeryToken].
- **HTTPS**: Enforced in Program.cs with app.UseHttpsRedirection();.
- **Password Hashing**: Identity automatically hashes and salts passwords.

```
services.AddDefaultIdentity<ApplicationUser>()
      .AddRoles<IdentityRole>()
      .AddEntityFrameworkStores<AppDbContext>();
```

---

5. Views

5.1 Register Page (Register.cshtml)

- Fields: Email, Password, Confirm Password
- Includes AntiForgeryToken

```
<form asp-action="Register" method="post">
   @Html.AntiForgeryToken()
   <input asp-for="Email" />
   <input asp-for="Password" type="password" />
   <button type="submit">Register</button>
</form>
```

5.2 Login Page (Login.cshtml)

- Accepts username & password
- Redirects user based on role

5.3 Product Listing (Products/Index.cshtml)

- Displays list of products with safe output encoding (@Html.DisplayFor).

5.4 Admin Dashboard (Admin/Dashboard.cshtml)

- Restricted to Admins. Shows “Welcome Admin, you can manage products.”

5.5 Access Denied (AccessDenied.cshtml)

```
<h2>Access Denied</h2>

<p>You do not have permission to view this page.</p>
```

---

6. Sample Input & Output

Input 1 (Admin Login):

- Username: admin@shop.com
- Password: Admin@123

Output: Redirected to /Admin/Dashboard → “Welcome Admin!”

Input 2 (Customer Login):

- Username: user@shop.com
- Password: User@1234

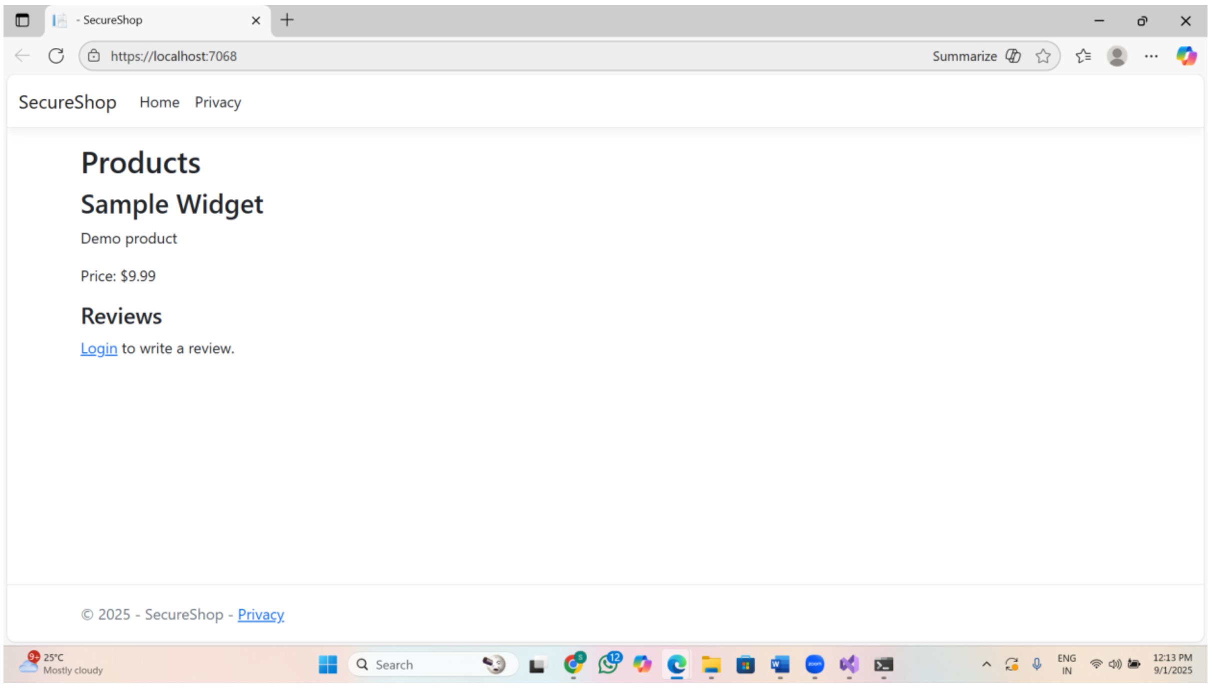
Output: Redirected to /Products/Index → Product list visible.

Input 3 (Unauthorized Access):

- Customer tries /Admin/Dashboard

Output: Redirected to /AccessDenied.

Screenshot of Output:



Products

Sample Widget

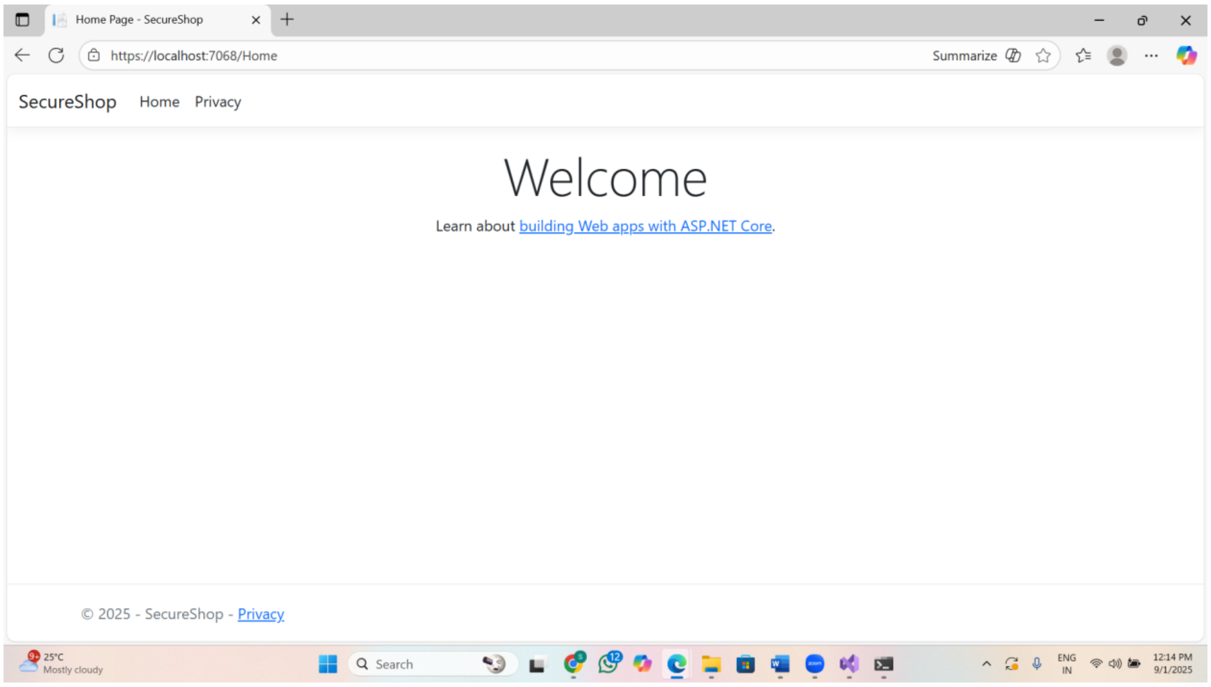
Demo product

Price: $9.99

Reviews

Login to write a review.

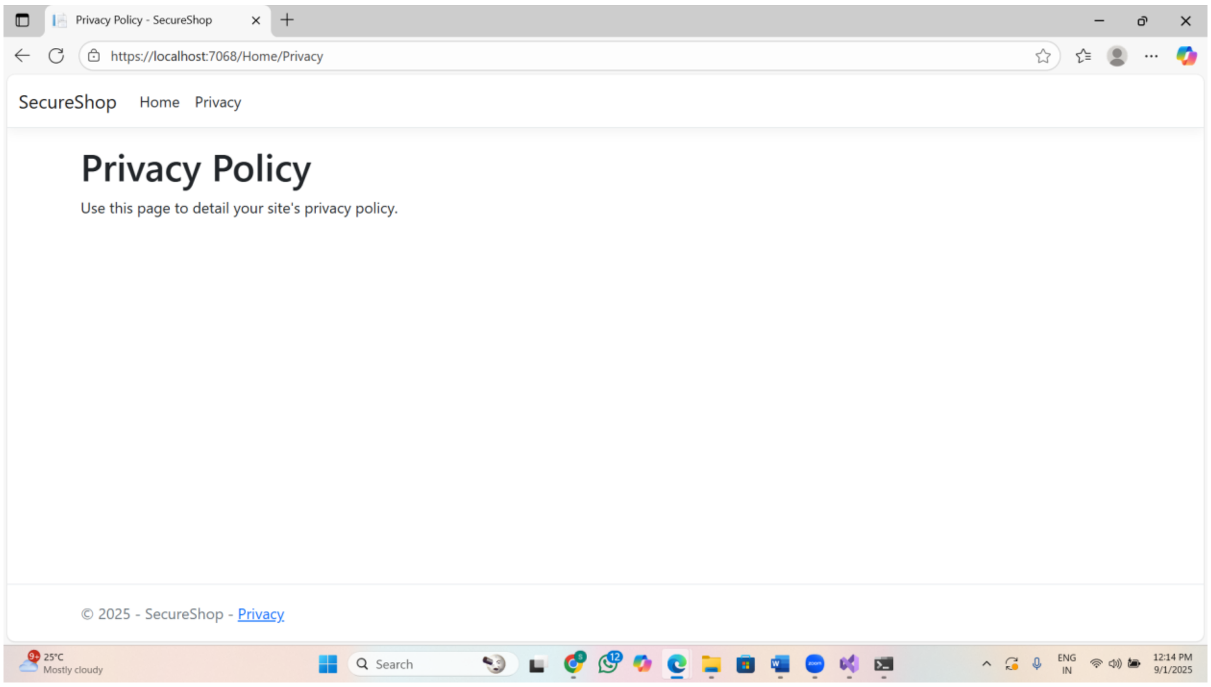
© 2025 - SecureShop - Privacy



Welcome

Learn about building Web apps with ASP.NET Core.

© 2025 - SecureShop - Privacy

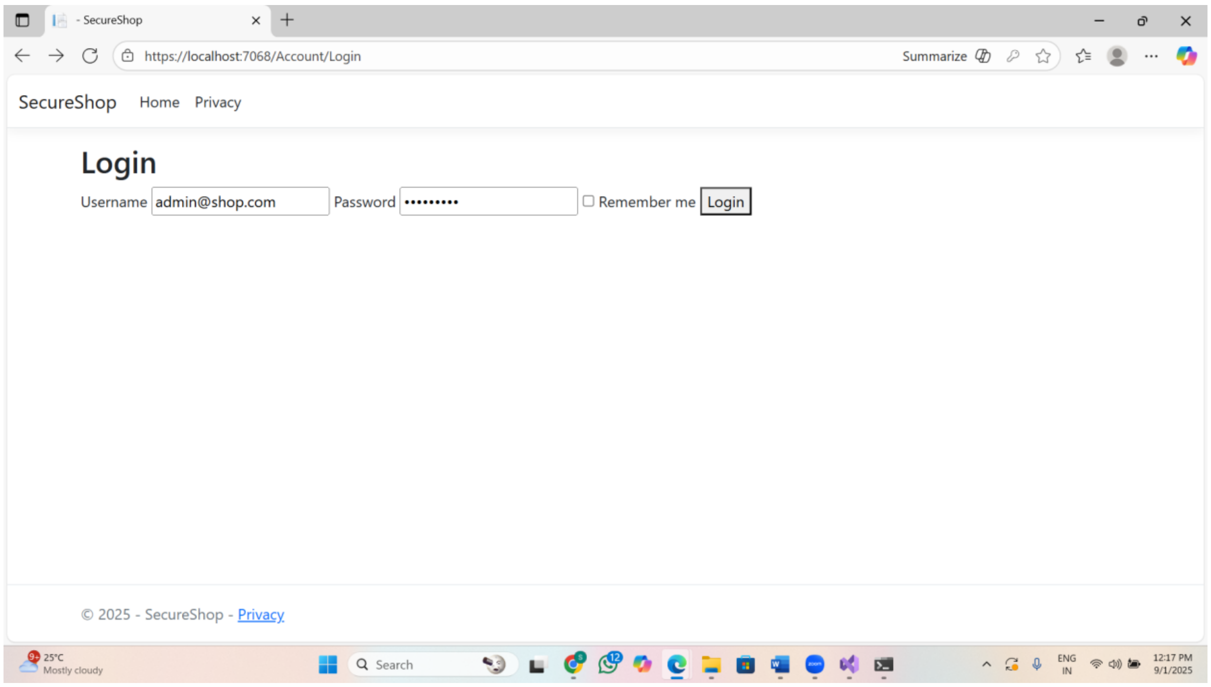

https://localhost:7068/Home/Privacy

SecureShop Home Privacy

Privacy Policy

Use this page to detail your site's privacy policy.

© 2025 - SecureShop - Privacy


https://localhost:7068/Account/Login

SecureShop Home Privacy

Login

Username admin@shop.com Password ••••••••• ☐ Remember me Login

© 2025 - SecureShop - Privacy

7. Conclusion

- Implemented **secure login, registration, and role-based access**.
- Protected against **SQL Injection, XSS, and CSRF attacks**.
- Learned use of **ASP.NET Core Identity + EF Core + Secure Coding Practices**.