1. Coding Problem: Secure Login and Role-Based Access in an ASP.NET Core MVC Application

Problem Statement:

You are building an ASP.NET Core MVC application that requires implementing authentication and role-based authorization. Your goal is to create a secure login page that validates user credentials and redirects the user based on their role. Implement the following requirements:

1. **Security Fundamentals and Authentication:**
    - Create a secure login page that accepts a username and password.
    - Implement authentication using `UserManager` and `SignInManager` to validate the credentials.
    - Hash the passwords and store them securely.
2. **Authentication and Authorization in ASP.NET Core:**
    - After successful login, assign roles to users (`Admin` or `User`).
    - Implement role-based authorization for different views:
        - `Admin` role should have access to a dashboard displaying user details.
        - `User` role should only have access to a basic profile page.
    - Secure the `Admin` page using the `[Authorize(Roles = "Admin")]` attribute.
3. **Securing MVC Applications:**
    - Use `AntiForgeryToken` in forms to prevent CSRF attacks.
    - Implement `RequireHttps` in the `Startup.cs` to force the application to use HTTPS for all requests.
    - Secure sensitive data using `Data Protection APIs` where necessary.

Requirements:

- Create the following views: `Login`, `AdminDashboard`, and `UserProfile`.
- Use the `AccountController` for managing login and role assignment.
- Implement `Authorize` attributes to secure the views based on roles.
- Ensure all forms use `AntiForgeryToken` to prevent CSRF attacks.

Sample Input:

1. **User Login:**
   `Username: admin`
   `Password: Admin@123`

2. **Role Assignment:**
    - Username: admin -> Role: Admin

    - Username: user1 -> Role: User
3. **Access URLs:**
    - Admin tries to access `/Admin/Dashboard`
    - User tries to access `/Admin/Dashboard`

Sample Output:

1. **Successful Admin Login:**
    - The admin is redirected to the `AdminDashboard` page with a message:

```
Welcome, Admin! You have access to the Admin Dashboard.
```

2. **User Accessing Admin Page:**
    - If a user with the `User` role tries to access `/Admin/Dashboard`, they should see an error page with the message:

```
Access Denied: You do not have permission to view this page.
```

3. **Successful User Login:**
    - The user is redirected to their `UserProfile` page with a message:

```
Welcome, User! Here is your profile information.
```

Explanation:

- **Login Page** authenticates users using hashed passwords and assigns appropriate roles.
- **Authorization** is enforced using `[Authorize]` attributes to restrict access based on roles.
- **HTTPS and CSRF Protection** are implemented to secure the application against common attacks.