1. Coding Problem: Role-Based Product Management System in ASP.NET Core MVC

Problem Statement:

You are developing a product management system using ASP.NET Core MVC. The application should have secure access control based on roles (`Admin` and `Manager`). Implement the following requirements to manage product information:

1. **Security Fundamentals and Authentication:**
   - Create a secure registration and login page using ASP.NET Identity.
   - Implement password policies (minimum length, uppercase, special character requirements) for user registration.
   - Implement `SignInManager` and `UserManager` for user authentication and role assignment.
2. **Authentication and Authorization in ASP.NET Core:**
   - Assign the following roles during registration:
     - `Admin`: Full control over product creation, editing, and deletion.
     - `Manager`: Can only view and edit products but cannot delete them.
   - Implement role-based authorization to control access to the product management features.
     - `Admin` should have access to `Create`, `Edit`, and `Delete` actions.
     - `Manager` should have access to `Edit` and `View` actions only.
3. **Securing MVC Applications:**
   - Use the `[Authorize(Roles = "Admin, Manager")]` attribute to protect the product management pages.
   - Implement `AntiForgeryToken` in forms to prevent CSRF attacks.
   - Use HTTPS for secure data transmission and add `Data Protection APIs` to handle sensitive data (e.g., product prices).

Requirements:

- Create the following views: `Register`, `Login`, `ProductList`, `CreateProduct`, `EditProduct`.
- Implement `ProductController` for handling CRUD operations on products.
- Use `Authorize` attributes to enforce access based on roles for each action method.
- Secure the entire application using HTTPS and apply `AntiForgeryToken` for form submissions.

Sample Input:

1. **User Registration:** ○ **Input 1**

```
Username: admin
Password: Admin@123
Role: Admin
```

   - **Input 2**

```
Username: manager1
Password: Manager@123   Role:
Manager
```

2. **Product Operations:**
   - Admin accesses `/Product/Create` and adds a new product:

```
Product Name: Laptop

Price: $1200
```

   - Manager accesses `/Product/Edit/1` and updates the product: `New Price: $1100`

   - Manager tries to access `/Product/Delete/1` but is denied access.

Sample Output:

1. **Admin Creating a Product:**
   - After adding a new product, the admin is redirected to the `ProductList` page with a success message:

```
Product "Laptop" has been successfully created!
```

2. **Manager Editing a Product:**
   - After updating the product price, the manager is redirected to the `ProductList` page with a message:

```
Product "Laptop" has been successfully updated!
```

3. **Manager Trying to Delete a Product:**
   - When a manager attempts to access `/Product/Delete/1`, they see an error message:

```
Access Denied: You do not have permission to delete products.
```

Explanation:

- **Registration and Login:** Secure user registration and login process using ASP.NET Identity with proper password policies.
- **Role-Based Authorization:** Controls user access based on their role (`Admin` or `Manager`).
- **Product Management:** Only `Admin` can create and delete products, while both `Admin` and `Manager` can edit products.
- **Security Measures:** Uses HTTPS, CSRF protection, and data encryption for sensitive information.