CS526 Enterprise and Cloud Computing
Stevens Institute of Technology—Fall 2020
Assignment Three—Web Security

This assignment requires the use of Visual Studio 2019 and ASP.NET Core MVC 3.1. You are provided with a partially completed Web app, which you should complete. The app includes a banner image and two style sheets (one default, one for ADA clients). You may modify the look and appearance of the Web site (e.g. by changing the banner image), but you should leave the general structure: a header area, a navigation bar with navigation links for the Web site, a sidebar on the left, a content area on the right, and a footer. You will use this template to build a simple social network that is intended to allow people to share photographic images. The layout for all Web pages is specified in the `Layout` view (rather than the default `_Layout` view), as set in `_ViewStart.cshtml`.

Your project should include the models from the previous assignment, and the basic UI. In this project, you will complete this application by adding proper user authentication, and by making the project as secure as you can.

Use the ASP.NET Identity API to manage user credentials and authorization (role membership). Store both this authorization information and your application data in the same database, extending the User entity objects defined for ASP.NET Identity to store your user information. The Entity Framework code for defining the database is generated by the wizard for creating an ASP.NET project with individual authentication, while you are responsible for writing the application initialization code that seeds the database with some initial users and roles.

In addition to user information, the database defined by ASP.NET identity includes information about roles, for which your initialization code should define three roles: `User`, `Approver` and `Administrator`. The `User` role is for general users of the site. The `Approver` role is for administrators who control the approval of images uploaded to the site. We will assume from now on that images do not become available immediately after being uploaded, but must be approved first of all (e.g. this may be a check for indecent content or breach of copyright). The `Administrator` role is for administrators who control user accounts (including unlocking a locked user account, and deleting a user account).

Replace the `Account` controller from the previous assignment with an `Account` controller that performs proper authentication:

- The `Register` action saves user information in a SQL Express database.
- The `Login` action authenticates a user based on the credentials they provide, and based on the information in the authorization database.
- The actions for the `Account` and `Images` controller should require proper authentication. If a user has not authenticated, then they should be redirected to the `Login` action in the Account controller. Once they login, they are redirected back to the page they were originally trying to access. If a user has logged in, but

does not have sufficient authorization to perform an action, then they should be prevented from accessing that action.

There is a default action, `Index`, for the `Home` controller, that displays a welcome message, personalized for a logged-in user.

There is a `Manage` action to the `Account` controller. This action allows an administrator to deactivate a user account: The user can no longer log in, and any images they have uploaded should be deleted (permanently). Implement this by associating a Boolean flag with every user account (in the `User` table), that is initially true to indicate that the user is active. The `Manage` action allows this flag to be set to false (disabling the account) and later back to true to reactivate the account. All images uploaded by a user should be deleted whenever their account is deactivated. The `Manage` form presents a table with all user names, in order to allow several accounts to be disabled or re-enabled at once.

There is an approve action (`Approve`) to the Images controller. This action displays a list of images that have been uploaded and not yet approved. Each image has a check box for approval, and form processing performs the appropriate logic. Only a user with `Approver` authorization should be able to see and submit this form. The form should display a table with all images that have been uploaded, and not yet approved.

It is important that you take steps to ensure that your application is robust against malicious attacks.

1. Access to the application should be restricted to authenticated users, except for the Register and Login actions for the accounts controller. Use the Authorize attribute to restrict access, requiring specific role-based permissions for certain actions (Administrator permission to deactivate/reactivate users, and Approver permission for approving uploaded images).
2. Protect users against CSRF attacks using secret token validation: Any form that uses asp-action or asp-controller tag helpers automatically includes a secret token when it is rendered, and your code must then use ValidateAntiForgerToken to validate the presence of such a token when posted back. You should also protect against open redirect attacks, and avoid over-posting attacks by never binding an entity model using the model binder (use view models instead).
3. ASP.NET Core applications require communication over SSL *as a default* (to protect the session cookie). In general you should protect against XSS attacks using HTML encoding and use measures to protect against cookie stealing (and consider using a library like AntiXSS), but we will not require it for the assignment.

Submission:

Submit your assignment as a zip archive file. This archive file should contain a single folder with your name, with an underscore between first and last name. For example, if your name is Humphrey Bogart, the folder should be named Humphrey_Bogart. This folder should contain a single folder for your solution named `ImageSharingWithSecurity`, with the projects `ImageSharingWithAuth`.

In addition, record mpeg or Quicktime videos demonstrating your deployment working. Demonstrate:

1. Registration of a user.
2. Automatic redirection to login upon an attempt to perform an action that requires authentication.
3. Prevention of an ordinary user from doing a restricted action (i.e., image approval).
4. Non-display of an uploaded image until it is approved (show the upload action).
5. Approval of an image and its subsequent display.
6. Deletion of a user and their images. Show the user being prevented from logging in, and their images no longer available, after they are activated, and the user being able to log in again after being reactivated.

Make sure that your name appears at the beginning of the video, for example as the name of the administration user who manages the Web app. *Do not provide private information such as your email or cwid in the video.* Be careful of any "free" apps that you download to do the recording, there are known cases of such apps containing Trojan horses, including key loggers.