CS526 Enterprise and Cloud Computing
Stevens Institute of Technology—Fall 2017
Assignment Four—Cloud Storage

Description:

- This assignment requires the use of Visual Studio 2017.

- You should create an ASP.NET MVC project (MVC5, .NET Framework 4.5, C#). Choose the wizard for creating an Internet Project (i.e., including forms-based authentication). Name this project `ImageSharingWithCloudStorage`. You should also create the unit tests project, `ImageSharingWithCloudStorage.Tests`, and place both in the same solution. The structure of your presentation logic (default layout and styles) should be the same as in the previous assignments, but *you must make this a new project*, because of the way that Windows binds projects in the registry. You should create a new project and copy the relevant files from the earlier project to this new project. Alternatively, you can try copying your old project, editing the project descriptors for the solution, and changing the application guid, but you are responsible if the project fails to run for grading because of e.g. registry problems.

- Your project should include the models from the previous assignment, and the basic UI, including user authentication based on ASP.NET Identity. In this project, you will move the application you have developed in the last three assignments into the cloud. The external API as far as users of your system are concerned will not change over the previous assignments, (aside from on additional piece of functionality), but your application will now run as a cloud application. You can find documentation about the latest StorageClient API here.

- Instead of storing information in SQL Server, you should now use **SQL Database** to store this information in the cloud. Use Entity Framework as before to access the underlying database. Your main task here will be defining a connection string to use SQL Database. You will continue to store the Users, Images and Tags tables in this database, but see below.

- Use **Azure Blob Storage** to save images, instead of storing them on the server file system. Store all of the images in a single blob container. *You have two options here; pick one and document your choice.* One is to simply store images in blob storage as though the container is a file folder, and store information about those images in SQL Database, as you have been doing. This is the easy option, but does not get you far into blob storage. Another option is to use a blob naming convention where a blob name consists of the user name of the user who uploaded the image, followed by a delimiter ("/"), followed by an image identifier that is unique for that user. You can generate this key by storing a sequence identifier in the User table in SQL Database, and incrementing that identifier each time the user uploads an image. Note that Blob Storage does not provide an operation for deleting a virtual directory. Therefore if you delete a user account with this scheme, you will still need to delete each image uploaded by that user individually, rather than issuing a single delete request for all images with that user name's name as their "virtual directory" name. With this option, you should place the information that was in an image record into the metadata for the blob for that image instead. If you pursue this second option, you will still need the Images table, now just containing a database primary key and the URI for the blob in Azure storage, because you still need to represent the one-to-many relationship from tags to images. Although theoretically this relationship could be represented using Azure Table Storage, in fact we would like this relationship to be many-to-many (you have not been asked to support this many-to-many relationship, yet), and doing this efficiently in Azure Table Storage would require secondary indexes. For now, leave the one-to-many relationship from tags to images represented unchanged in SQL Database, no matter which of the two options you choose above. If you choose the second option, you no longer need to represent the one-to-many relationship from users to images in SQL Database, but you still need the Users table for other purposes (e.g. for authentication and authorization).

- Add a table in **Azure Table Storage** which records every time that an image and its details are viewed, and by whom. Call this table `ImageViews`. Assume that the most common query is "What images were viewed today?" so you can use the same key organization as in the example in the lectures. Use a partition key based on date, and use a primary key based on image id. Provide an additional action in the `Account` controller, called `ImageViews`, that displays a report of viewing for a particular day, organized by image id (display each image caption in the report as well). The GET action should provide a drop-down list with dates for the last two weeks, with today's date as the default, and display viewing activity for the date chosen from this drop-down list. Create a separate role, called `Supervisor`, to authorize access to this action.

- Deploy your application using Windows Azure App Service[1]. You can find a tutorial on how to do this here. You should use an Azure 30-day free trial subscription for this and the next assignment. *Use a MS Live account to register for the free trial subscription*, and **wait until you are ready to deploy to subscribe**. Once the 30-day period is over, you will still be able to use many Azure services for free for up to a year. You should use the Shared subscription tier to minimize your charges, it should just be a few dollars a month, but with judicious timing you should be able to just use the free subscription for these assignments[2].

[1] App Service replaces Azure Websites, integrating Azure support for Web apps with support for mobile apps.
[2] The Free Tier does not support SSL, custom domain names for your apps, or use of Virtual Machines.

- If you allow public access to images uploaded to blob storage, make sure that none of the images you use for testing violate copyright!

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 `ImageSharingCloudStorage`, with projects `ImageSharingWithCloudStorage` and `ImageSharingWithCloudStorage.Tests` (and DLLs in the folder `packages`).

In addition, record mpeg or Quicktime videos demonstrating your deployment working. See the rubric for further details of what is expected. 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.