Assignment 9 – Cloud Storage  
COS318 – Web Programming

You’re at tribal council, not sure if you can trust everyone who is sitting around you. Perhaps some of them really are on your side, or perhaps they are about to blindside you. Whatever the outcome, it is the people who learn to outwit, outplay, and outlast who will be at the final tribal council. In this ninth assignment, you will be using azure cloud storage to save table entities and images. Here’s to hoping that Jeff does not snuff out your torch.

1. **(40 Points) React**
   1. **imagesContainer:** The front end should display a list of images that have been uploaded to your Azure blob container. These image elements should be created dynamically from the JSON response of image entities received from the server.
      1. Each image should be enforced to have a reasonable max height and width and must use the name of the image for the alt attribute.
      2. Hint: the “img” element causes the browser to make a GET request to whatever URL is in the src tag. This can be used to call the server endpoint that returns a single image (which will then redirect to the image itself in the Azure blob container).
   2. **uploadComplete:** The image upload is a three-step process. First, a POST request is sent to create the image, then the image is uploaded, and finally, an upload complete is sent to the server to save that the image has finished uploading. The first two of these steps are already done in App.js.
      1. After the POST request that created the image has been completed, send another fetch to save that the image upload is complete. Then use the response from the upload complete to display the new image on the page.
2. **(40 Points)** **ImagesController**
   1. **GetAsync(id):** Return a URL to an image that is stored in an Azure blob container as a temporary redirect.
      1. The redirect should be cached for seven hours.
   2. **PostAsync:** Accept an ImageEntity JSON that allows the client to specific the image name to be created. The response JSON should have a URL that can be used to upload an image to an Azure blob container.
   3. **UploadCompleteAsync:** Accept a string id that is an id of an image previously created. Update that image in the database to save that the upload has completed (and therefore can be viewed by the client).
3. **(20 Points)** Code style, formatting, completeness, and quality.

Stretch Levels

If you already have a lot of experience with azure storage accounts, or if you just won an immunity challenge, try to complete these stretch levels for a reputation bonus. If you try for the stretch levels, make sure to type it in the comments on Moodle so I don’t miss it.

**Cirie Level**

Add some CSS to your page to make it look nicer. Background colors, font colors, or anything that looks good.

**Rupert Level**

Add a progress bar to your page that updates as the image is being uploaded.

**Boston Rob Level**

Add a slider to the page that controls image size.

The Rules

1. No inline styles or inline javascript.
2. Error messages must be “in-page” i.e. no pop-ups or alerts.
3. Any resources not created by you (images, javascript libraries, etc.) must be referenced using a CDN or URL, not directly included in your assignment submission.
4. All requests that submit a body to your server must have their entities validated with appropriate annotations, such as MinLength, Range, or Required.
5. The root path of your server must display the main page of your application.
6. Service/data/model classes must not have any http, request, or response references.
7. Controller entity classes must not be used directly to store data on the server; translate them into a model (data storage) class before saving the data. Conversely, controllers must not send any model classes to the user; translate them into controller entity classes before sending the response.
8. All service class instances must be obtained using dependency injection.
9. You may not use any synchronous methods in your C# code wherever there is an async option.