Snag Tagged
CSC 491/492
March 15th, 2016
Pate Cobb
Cal Poly Computer Science

Abstract

Currently, Facebook is the most popular social networking site in the world. In the fourth quarter of 2015 they surpassed 1.5 billion monthly active users [1]. Together, these users have uploaded over 250 billion photos to Facebook, an average of 217 each [2]. Users, however, do not have the ability to back these pictures up. To solve this problem, we created Snag Tagged. Snag Tagged is a website that allows users to download all of their pictures off of Facebook. The website utilizes Facebook's Graph API to authenticate users and query Facebook for their pictures. The website is hosted through Amazon Web Services and utilizes open source software to run the web server. In this report, we document the website as well as the process of building it.

Table of Contents

1. Introduction

Snag Tagged is a website that allows users to download their pictures from Facebook. Snag Tagged utilizes Facebook's Graph API in order to access users' photos. It uses the Javascript SDK on the client side and Python SDK on the server side.

Facebook does not offer an easy solution for users to download their pictures. Users are only able to download a single picture at a time which would prove very time consuming to obtain a copy of each individual picture. When users are tagged in a significant number of photos, they are forced to to rely on Facebook to continue host and allow access to their pictures. Many people do not want to rely on Facebook as the sole means to access hundreds or even thousands of pictures of them.

There is a need for an application that can provide a tool for users to obtain a copy of all of their Facebook photos. The solution I created to solve this problem is a website called Snag Tagged. This website utilizes Facebook's Graph API to authenticate users and download their photos. All a user must do is to login using their Facebook credentials, giving Snag Tagged permission to access their photos, and then click download. The server downloads approximately 25 pictures per second, and upon completion the user will be prompted to download a zip file containing each picture. The speed in which pictures are downloaded is limitied because it is running on a single, small AWS instance. Alternatively, the user may also opt to upload their pictures to Google Drive as another means of storage.

There is conflicting information regarding if an application has access to photos that a user is tagged in. For example, if user A is using Snag Tagged and is tagged in a photo uploaded by user B, this photo may or may not be returned. The documentation states, "in some cases the photo's owner's privacy settings may not allow your application to access it." Unfortunately, there is no information regarding what privacy settings prevent the photo from being accessed by Snag Tagged. Furthermore, there are some cases where a user is tagged in two photos with identical privacy settings, but only one of these photos is returned. A bug was reported for this example that was assigned to the Specialized Product team and is still awaiting resolution [8]. Another user reported a bug regarding the API not returning tagged photos, and was given the response, "It seems like this behaviour is intentional but the documentation hasn't been updated." No response has been received after asking for clarification on what exactly the intentional behavior is [7].

This market for this application is people who wish to use their Facebook photos for other purposes. More and more users are signing up every day, and there are already over a billion active users. Any one of them who wishes to download a copy of their pictures is an ideal user of Snag Tagged.

One could argue that since this application is relying on storage and accessibility from another company, it may be unethical to collect and aggregate this data. However, the data being accessed is individual user's photos. They are the owners of this data, not Facebook, and thus should have a means of downloading a personal copy.

2. Application

Snag Tagged works by allowing users to login through Facebook and grant permissions for Snag Tagged to access their photos. Once it has permission, the application is able to query Facebook API to download the pictures and return them to the user. Similarly, users can login to Google Drive and grant permission for Snag Tagged to upload pictures to their account. The diagram below shows how the application connects with the different parts.

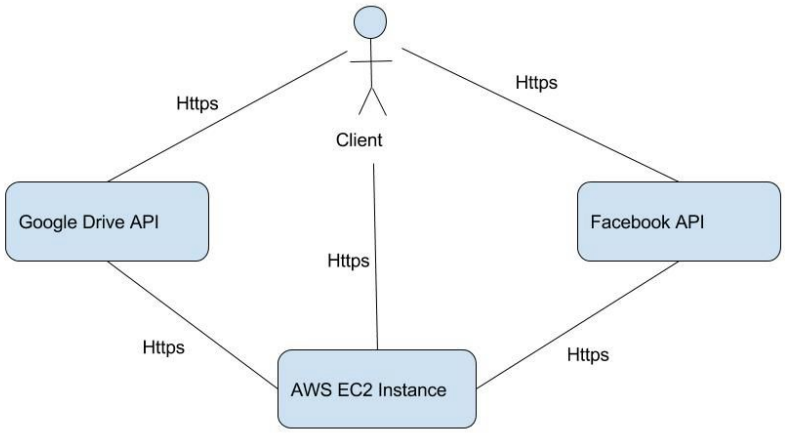

Figure 1: Diagram of Application

2.1. Facebook Authentication

User authentication is handled through a request to Facebook's Graph API. When a user navigates to the site, they are prompted to login using their Facebook credentials shown in Figure 2, and the login dialog is shown in Figure 3. Once a user is authenticated, Snag Tagged is now able to obtain a user token which is used to query their data. This access token is used when making calls to the API to prove both that the user has logged into Snag Tagged and has granted the appropriate permissions.

When a user is logging in, Snag Tagged requests the following permissions: `email, profile, and user_photos. Email` and `profile` are default permissions, but `user_photos` requires approval by Facebook pending a review of the application [5].

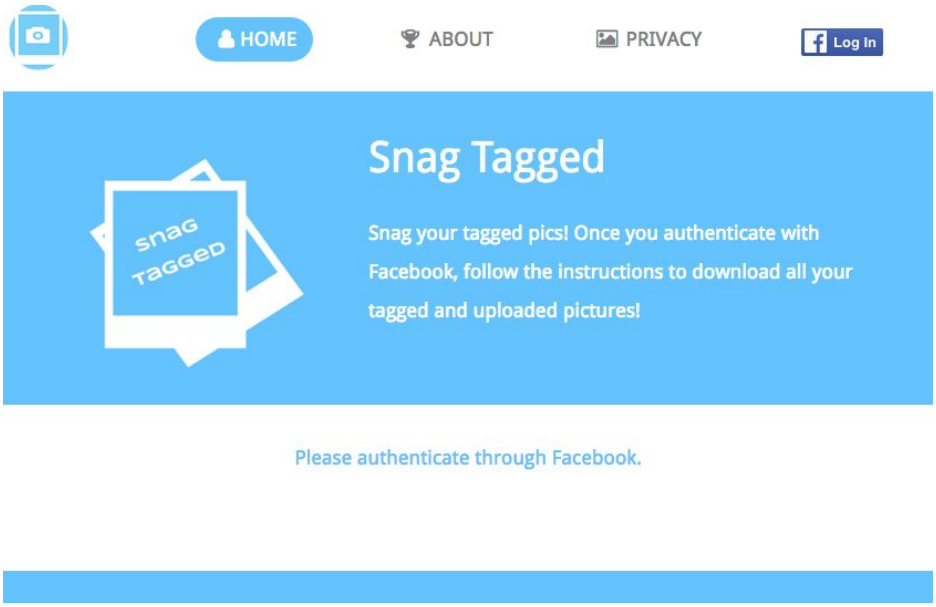

Figure 2 shows the landing page when user navigates to Snag Tagged.

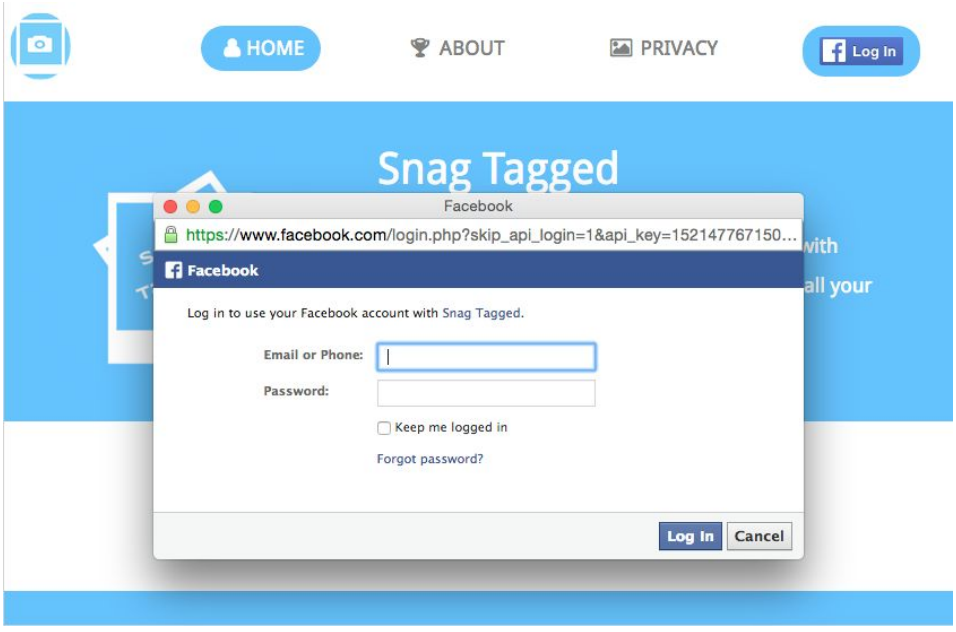

Figure 3 shows the login dialog that pops up when user clicks the Facebook login button.

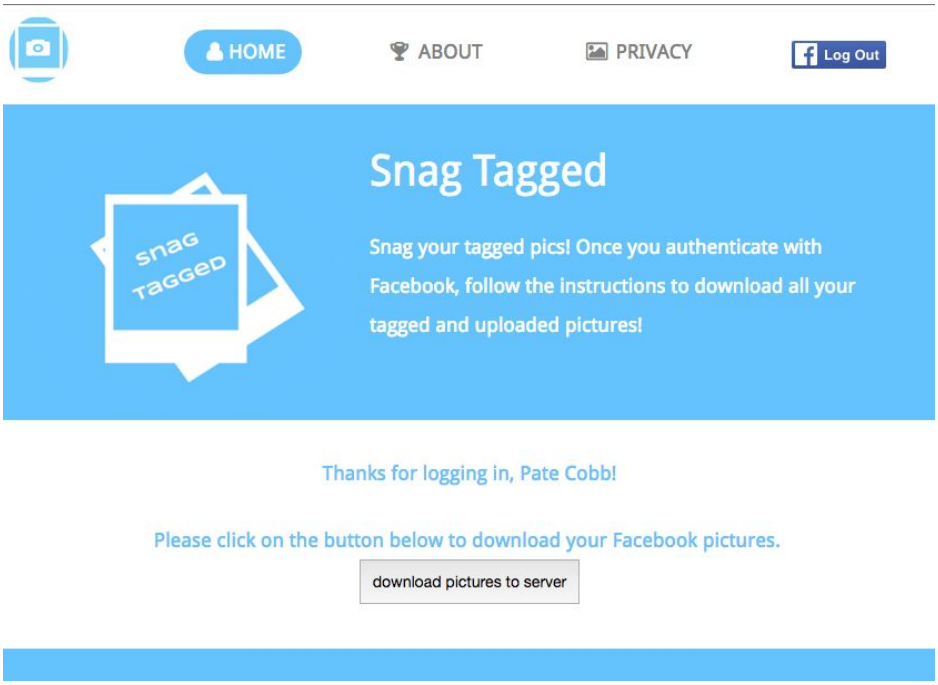

Figure 4 shows the view after a user successfully logs in.

2.2. Downloading Pictures to Server

When the user clicks the download button as seen in Figure 4, a request is sent to the server to begin downloading the pictures using the Graph API and the user is prompted to wait as showin in Figure 5. First, the application searches a user's profile to obtain list of all of their uploaded and tagged photos it has access to. Once the list of all photos to be downloaded has been compiled, it then starts sending requests to download each picture individually. Snag Tagged utilizes the python multiprocessing package in order to download pictures concurrently, significantly increasing the speed in which all the photos are able to be downloaded. After successfully downloading the pictures, a zip file is created and returned back to the user containing all their photos.

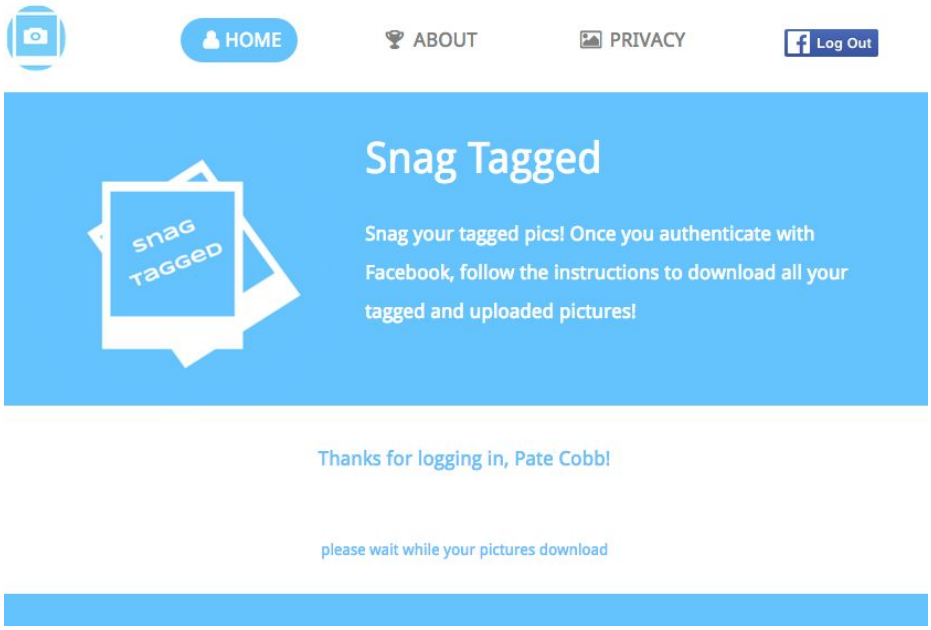

Figure 5 shows the view while user waits for pictures to be returned.

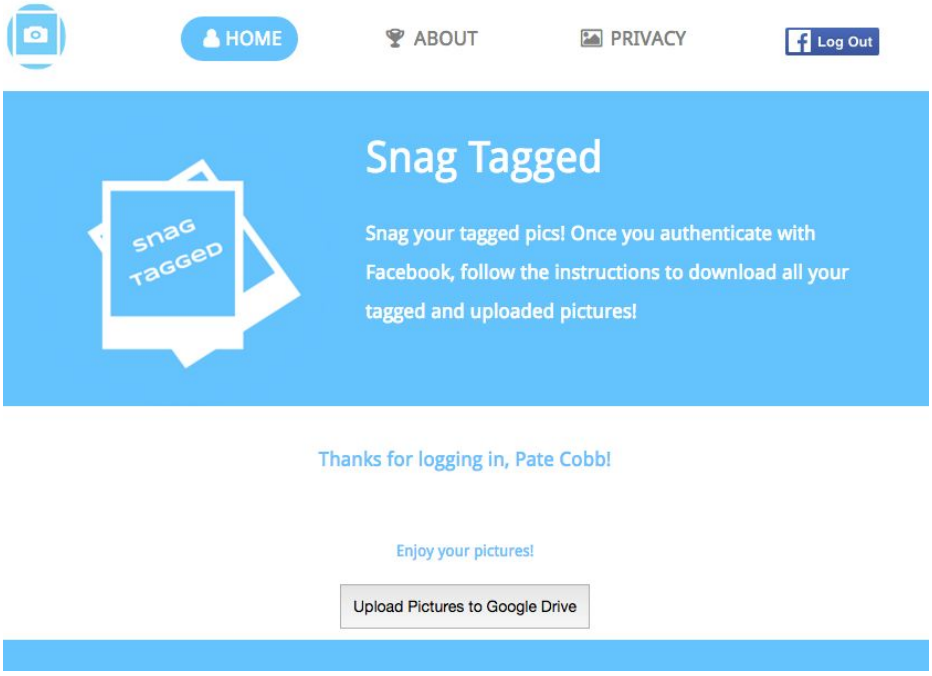

Figure 6 shows the view after user receives pictures.

2.3. Upload to Google Drive

The process for uploading pictures to Google Drive is very similar to downloading them from Facebook. First, the client side checks if the user is logged into Google Drive. If not, a pop up prompts them to login and authorize Snag Tagged as seen in Figure 7. Once they are logged

in, a user access token is retrieved and sent to the server in a request to upload their pictures. The server then uploads the zip file to the user's Google Drive account.

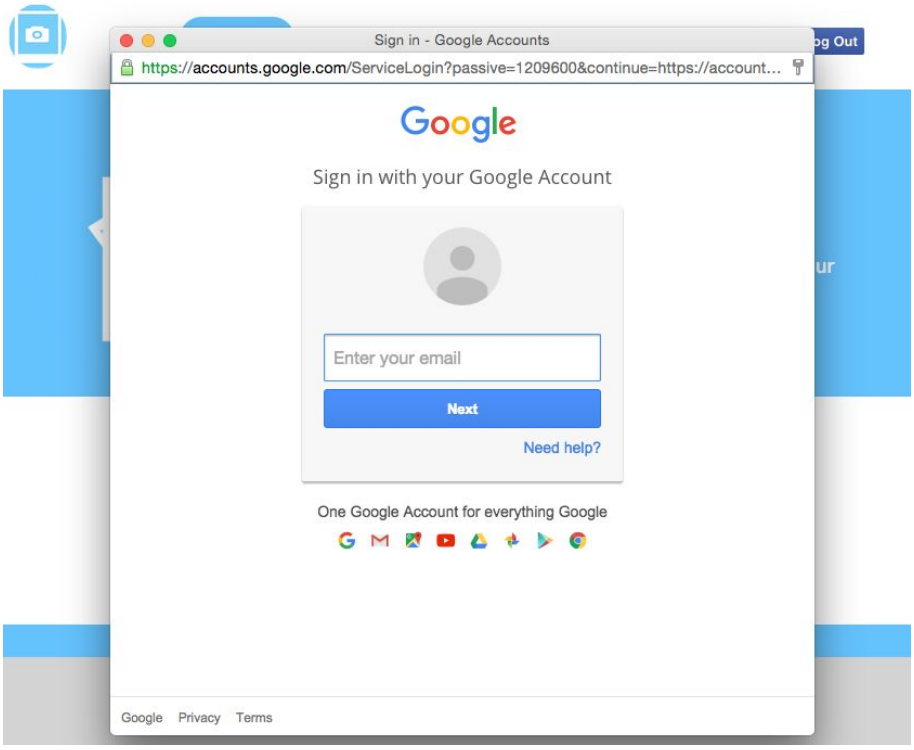

Figure 7 shows the dialog that appears when user clicks Upload Pictures to Google Drive

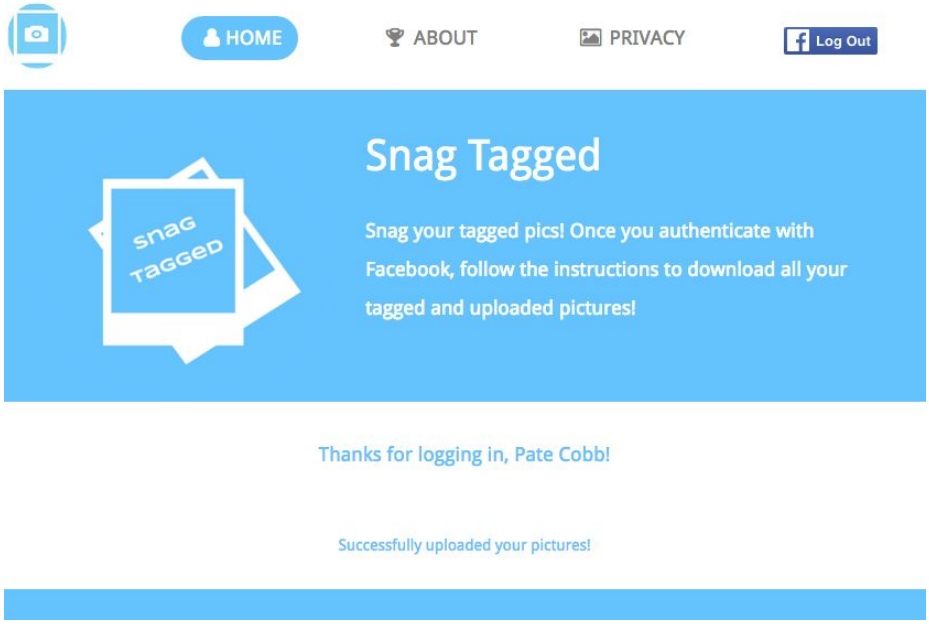

Figure 8 shows the view after pictures have been uploaded to Google Drive

3. Documentation

3.1. Facebook API

Photos are accessed through the `/photos` edge of the API. Since you can either query for tagged or uploaded photos, it is impossible to access all of a user's photos in one request. When the request for `/photos?type=tagged` is sent, this should return information for photos the user is tagged in that were uploaded either by the user or one of their friends. A request using `type=uploaded` will return information for all photos that have been uploaded by the user [4].

The information returned in these queries contains details of when and who uploaded it, likes, comments, and other metadata. The information used by the application is a unique identifier for the photo and a list of links to several different sizes of the image. These images are all different compressions of the original image that was uploaded. Since Facebook immediately compresses photos when they are uploaded, it is not possible to get a photo of the original size back after it has been uploaded.

There are a few cases where Facebook's compression does not affect photos. First, Facebook will not compress photos under 99kb. Second, because PNG files use lossless compression, they will preserve much more information than a JPEG file after undergoing Facebook's compression algorithm [3].

3.2. Google Drive API

Using the Google Drive API proved to be a faster process than with the Facebook API. The Javascript SDK provides a function `gapi.auth.authorize()` which handles all user authorization. This function takes as parameters a `client_id` which is used to prove authorization is being given to Snag Tagged for the current user. This function returns an object containing an access token which is then used by the server.

The server side utilizes the PyDrive package in Python. This package is essentially a wrapper around the Google Drive Python Client API. Once a request with an access token is received, it creates a new GoogleDrive object using the token for authorization. While looping through the list of pictures, this object is used to create a new file, set the contents to the current picture, and then upload.

3.3.Web Server

Snag Tagged utilizes NGINX as the web server and uWSGI as the application server. NGINX is configured to redirect users using http to https connections. Snag Tagged has a class 1 certificate from StartCom Certificate Authority for domain name validation. The application is running on an Amazon Web Services (AWS) server. Go Daddy was used to buy the domain name www.snagtagged.com, and Go Daddy also performs DNS routing for the website.

3.4.Web UI

The web UI design started with a free template [9] and was modified to fit the website's needs. The main aspects that were utilized were the menu bar at the top, as well as layout. I modified the style sheets to have the colors, fonts, and sizing that I wanted. I also designed the Snag Tagged logos that are used on the site.

4.Lessons Learned

The documentation for Google Drive's API was much more thorough and informative than the documentation for Facebook's API. This lead to a much simpler process of implementation within the application. Also, it appears there are bugs within Facebook's API which is the reason only a fraction of a user's tagged photos are returned.

4.1.Using API's

Facebook

Facebook's current running version of their API does not match the current documentation. Their documentation for the `user_photos` permission says it "Provides access to the photos a person has uploaded or been tagged in [5]." The permission requirements for an application to access a photo are "For any photos uploaded by someone, and any photos in which they have been tagged - A user access token for that person with `user_photos` permission" [6].
In reality, it seems the `user_photos` permission now only grants an application access to photos uploaded by the user who granted it. This means that the permission does not in fact give access to photos a user is tagged in unless they are uploaded by the same user. This behavior was introduced in order to allow the owner of the photo to control the privacy settings rather those who are tagged.

There are anomalies where this behavior is not followed. There are several cases when this application is used to download pictures, there are many photos returned which have been uploaded by other users who have not granted approval to the application. It is unclear whether

this behavior is a bug with Facebook mistakenly returning some pictures or under certain circumstances the photos are returned.

I submitted a bug with Facebook Developer's support who confirmed that this behavior is indeed a bug. I have not heard back about any fix, and the bug is still waiting for resolution by Facebook staff.

Google Drive

Overall, using the Google Drive API was a very smooth process. The documentation was much more thorough and everything behaved as expected. The PyDrive package also proved to be very useful and made API calls to Google Drive much more simple.

4.2.Facebook App Approval

Snag Tagged only required App Approval by Facebook because it required the `user_photos` permission. Most other permissions can be granted without having to submit an application for review. This process required a detailed description to be written about the application and its intended uses. This included screen shots from every step of the way documenting where the permission would come into play. Next, they required a screencast of a user navigating through the application, start to finish. Once submitted for review, it was approved in 3 days.

5. Ethics

One main ethical consideration that must be taken into account is that this application relies on data that another company is collecting. Facebook expends a lot of resources to host the billions of pictures on their site, which costs the company a sizable amount of money. Because of this, these pictures could be viewed as being Facebook's data. However, I would argue that since the users are the ones who are taking the photos, they are the ones who own them. Therefore, they should have the right to download the pictures at will.

5.Conclusion

Potential improvements that could be make to Snag Tagged would be hosting it on a larger AWS EC2 instance. This would increase data throughput for both downloading and uploading pictures, as well as allowing for increased multiprocessing. Increasing the instance size would be a necessity if this application is to grow to any significant scale. Another improvement would be to have a better UI design that is also compatible on mobile devices.

If I had more time, I would consider integrating Snag Tagged with other websites. This could include allowing users to get pictures from Instagram, Twitter, or their Google Plus accounts. Another idea that I explored was allowing users to transfer their photos directly to Shutterfly, so

they could have hard copies printed and shipped to their house. This could also generate profit if the users start buying their pictures on Shutterfly. Unfortunately, Shutterfly never responded when trying to register my application to use their API.

Resources

1. http://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/
2. http://www.businessinsider.com/facebook-350-million-photos-each-day-2013-9
3. http://havecamerawilltravel.com/photographer/sharp-text-facebook-image
4. https://developers.facebook.com/docs/graph-api/reference/photo/
5. https://developers.facebook.com/docs/facebook-login/permissions
6. https://developers.facebook.com/docs/graph-api/reference/user/photos
7. https://developers.facebook.com/bugs/1654180898168006/
8. https://developers.facebook.com/bugs/1676110236003717/
9. http://www.free-css.com/free-css-templates/page181/mini-portfolio#shout