**Instagram Visualizer**

**Overview**

The concept of our web application is to create a social media platform which uses a map view as a news feed. The user would be able to view their friend’s posts and other users for a particular area in the city. The application allows the user to access our app by logging-in with the social media authentication. Once logged-in the user is brought to our application home page. From there the user can create a filter to set the viewing preferences for location, distance relative to the location, number of posts displayed, and tag.

**Django**

We chose Django because it is an easy to use framework that is lightweight yet comprehensive enough for us to do what needed to make our web application. One of our group members had some experience with it, but the others had to learn as we created the app.

**Database**

We used SQLite as our database. We didn’t need to do any heavy database querying, and it was already pre-built into django, so we decided it was our best option. There are two tables in our database: json\_cache and user\_data. The ERD for these tables can be found in the “Documentation” folder under “Final Project”. “User\_data” contains the most recent filter a user has made. If a user logs out and logs back in, their last query is loaded from the database and into the filter page. It is connected to a user by the “userid” column. This is an integer field that holds Instagram’s user id. The “json\_cache” table holds the json data that is returned to us by the Instagram API. Because we are limited to 30 media requests per hour (because we are a sandbox app) it didn’t make sense to request data from the API every time someone changed a filter. Therefore, we save the json data in the database, and if a user makes a request after 15 minutes or more have passed since the json data was saved the json data is refreshed.

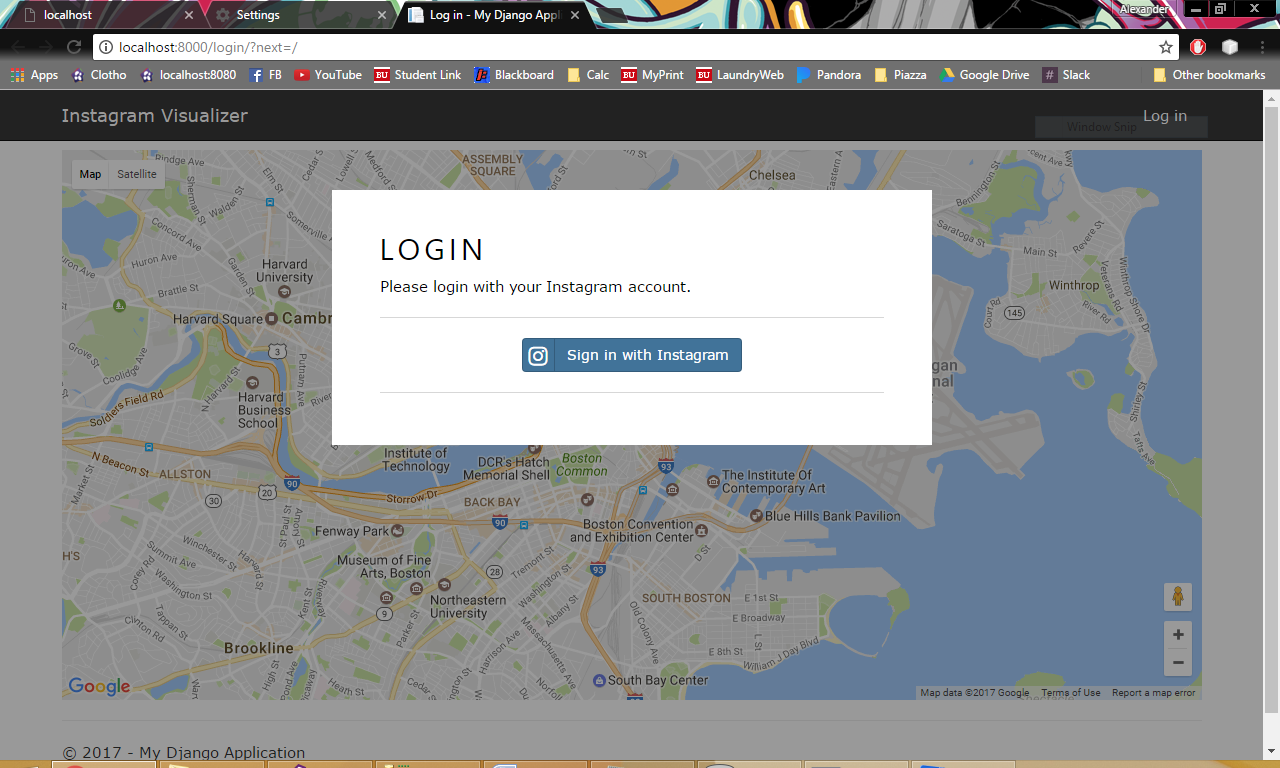
**Two Publicly Available APIs/Third party authentication**

We actually use three APIs in our project: Google Maps, Google Places, and the Instagram API. The Google Maps and Google Places APIs both use the same API Key. This authenticates us as developers, and allows use to get data from them. Google Maps is used to display the map and to display the markers that contain the images. We use google places in our filter to search for a location. If the location is found, Google Places returns the latitude and longitude, which we use to center our map and calculate the closest images to this point.

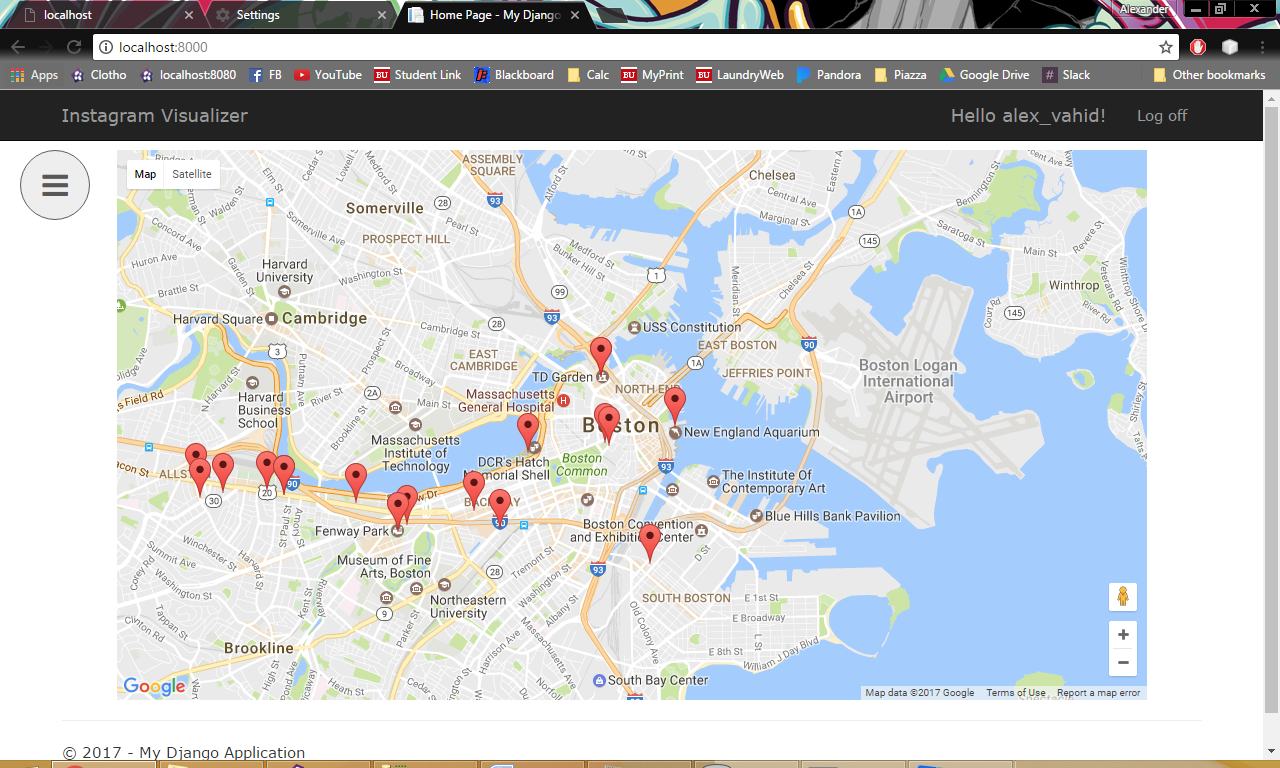
The Instagram API is used for authentication and for getting pictures. We downloaded a package called Django social login, which added middleware to aid in logging in with third party applications. When a user clicks “Login with Instagram” the user is redirected to Instagram, where they enter their credentials, and are routed back to our site with an OAuth token. This token has “public\_content” scope, meaning we can access that user’s public information, and make requests to the API for other user’s public content.

**Four Distinct States**

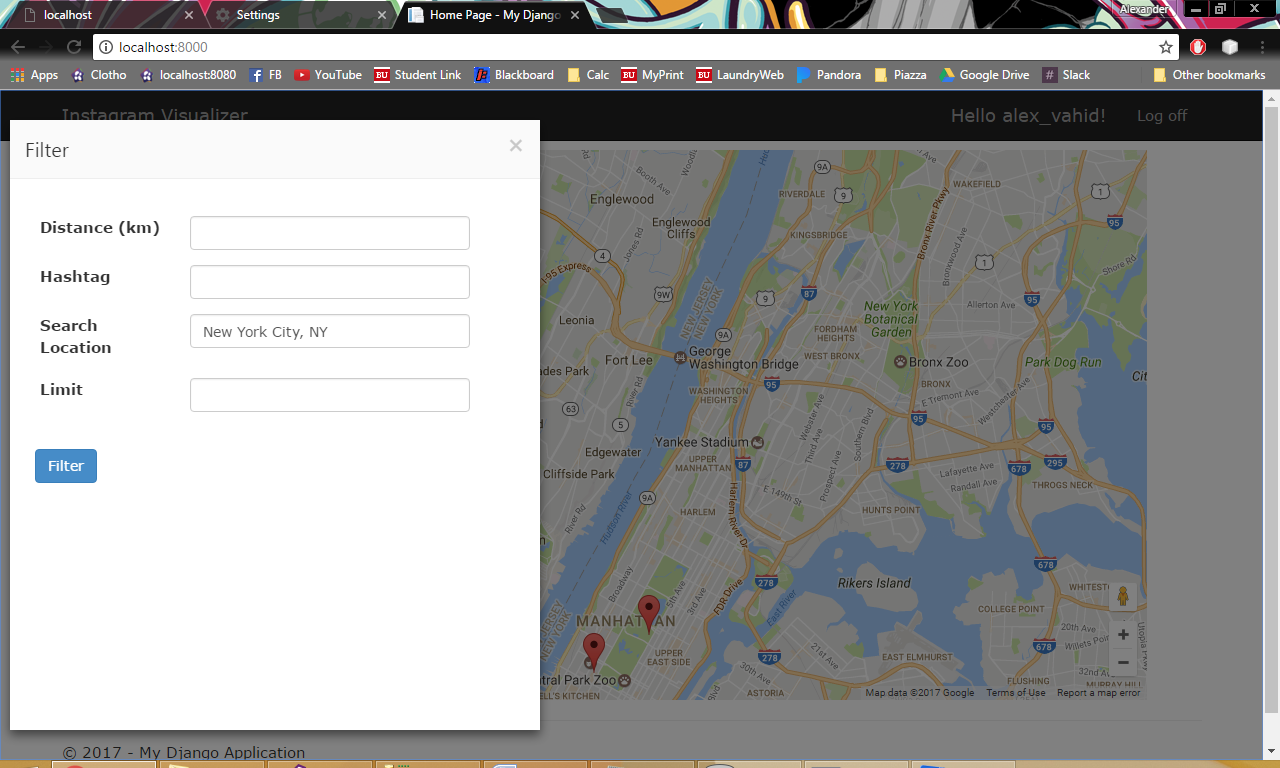
**State 1 : Third Party Authentication**



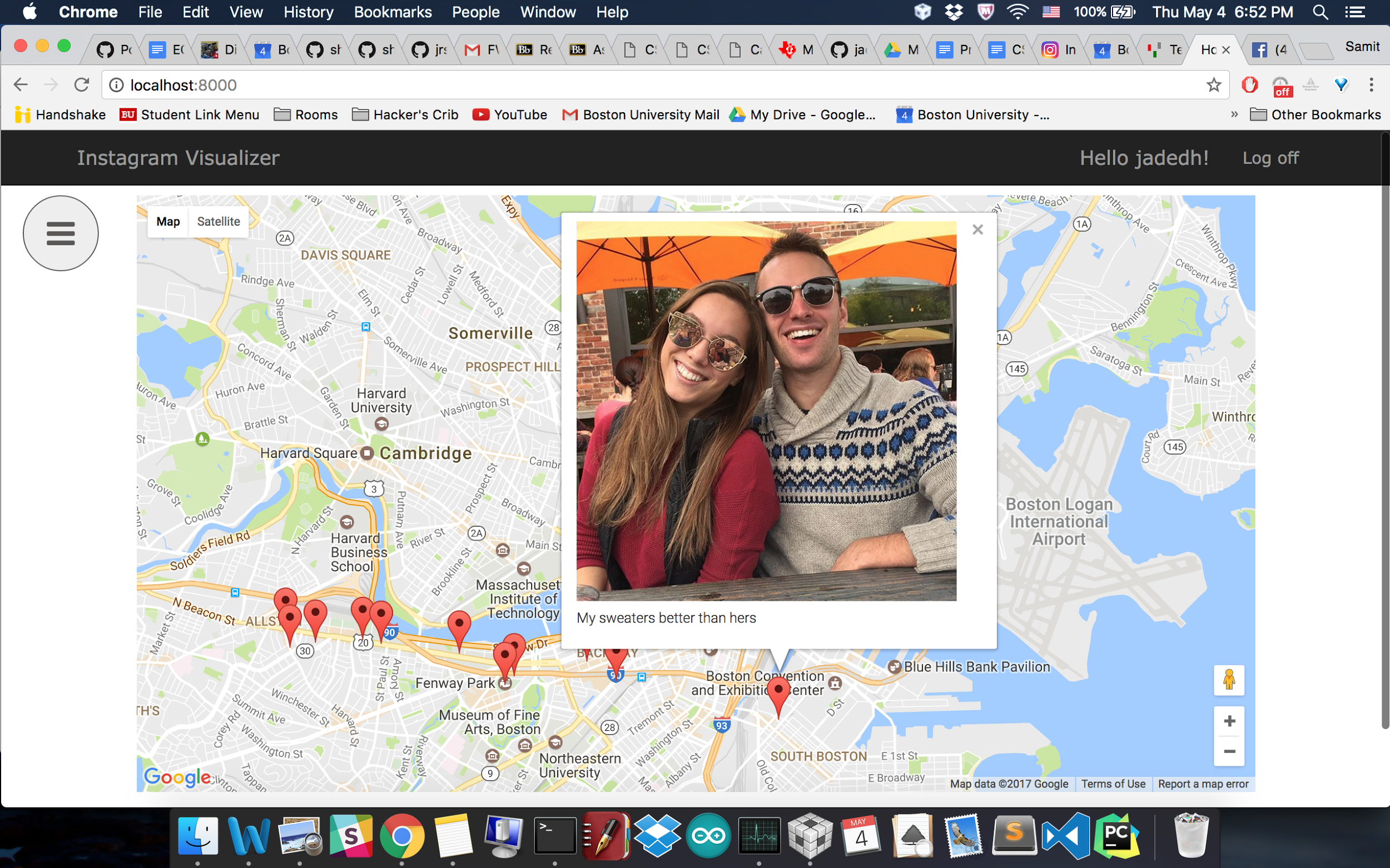
**State 2 : Home page**



**State 3 : Search Filter**



**State 4:**



**Transition from Facebook to Instagram API**

The application consumes three APIs: Google maps, Google Places, and Instagram. Initially we intended to use the Facebook API as our main social media source, however, we later found out that the Facebook API no longer supports the API calls to get the information we need from the user’s news feed. We decided to switch to the Instagram API as Instagram offers GET requests which return JSON objects with friend’s post information. For instance, we have access to uploaded pictures, captions, and locations(Latitude and Longitude) which we use to display on our app.

**User Case Changes**

In the early design phase we aimed to implement basic Facebook functions such as like, comment and share. As we switched from using the Facebook API to the Instagram API we encountered some technical difficulties. Instagram allows developer to consume their API in sandbox mode, and in order to go live mode we need the company’s approval to officially launch our application. Instagram has several restrictions that prevented us from getting endpoint responses from media that does not belong to the user of the access token. In addition, when we are accessing those resources, sandbox mode only allows us to use information from up to ten registered sandbox mode users, and 20 most recent media from each of those users.

We were able to display posts with location and caption using the google maps API; however, posts without location were not able to be displayed by our application.

With the new Instagram GET call we were able to implement tag search, which allows the user to choose to display the posts with the specified tag added by the instagram user. This could not have been done with Facebook API as it does not support hashtag.

**Team Management**

During the first week when our team was assigned, we chose to use slack as our main mode of communication. We shared our class schedules on google calendar and agreed on a weekly meeting (Tuesday and or Thursday 10:30AM - 1:00PM). We decided to make it a policy that everyone needs to discuss and constantly update any new ideas and changes to be made to the project. We mostly did the research individually during the week for ideas and solutions, and came to finalized each deliverable together at the meeting each week. We figured that it is most efficient to all work together on each milestones as we were able to bounce off ideas with each other and error check each other along the way. Several of our team members have less experience on the technology we chose for the project, so it really encouraged us to learn as the project was developing.

**Potential Next Steps**

For our potential next steps, we discussed about how we could do the user interface design, even though not required for the project deliverable. Most of the functions we want to implement is limited by the sandbox user mode but we could potentially apply media functions such as like and comment on the following’s posts.

Also, in addition to taking advantage of Instagram APIs call requests we discussed about the idea that we could potentially move the project off of using any social media API. In this case we would add new functions to our application to allow the user to posts pictures/videos with captions, etc, through our own application which would be stored. In other words develop our application into an independent social media platform that would allow users to know what other users are doing based on the location they want to explore. With this we would be able to implement our own unique functions.