Wau: Location-Based Photo-Sharing and Friend-Finding Android Application

Senior Project

Tam P. Nguyen

thien93@gmail.com

June 10, 2015

1. Introduction

Wau (Where are u?) is a location-based photo-sharing and friend-finding Android application. The goal of this project was to broaden my knowledge in software programming, while attempting to create something useful for the public. Since mobile applications have become so prevalent in our everyday technology, I figured I would be able to hit a large consumer base by trying to make one myself. Given the option of working in either of two languages I had no experience with (iOS or Android), I decided to pick Android because I was most familiar with Java, which Android primarily uses. The basic concept of my app is to help people avoid the hassle of having to call/text their friends in order to know where they are. I imagine this app to be largely used at big events where people get split up easily, or for when driving to pick up a friend at an unknown location. The ability to simply tap on a name and visually see their location in reference to your own would be a huge convenience. Also, I hoped to include the ability to have people send a picture of their location to further increase the convenience of having to look around once in the vicinity.

As this idea developed, I wanted to include more features into my app. Thus, relating to coordinates and taking pictures, I decided to incorporate public photo-sharing. The neat thing about this feature is it allows users to take and upload pictures wherever they are, and the only way for anyone to see the picture is to be within the same vicinity that it was taken. I imagine people using this functionality for scavenger hunts, geo-caching, or even for looking back at old photos in a specific area. The possibilities are endless!

The plan for implementing this application included the use of Android Studio (IDE), Parse (BaaS), and Google Maps (location-based services). All of which I have never worked with before, so the goal is to expand my programming knowledge in these areas.

Related Works/Apps

WhereUAt:
"With WhereUAt, you can send a location request with just a single tap. When you get the location request from your friends, you can choose to share your location or dismiss it silently with a single tap. Responses you send are just like what you would type in a chat message. Along with the message, your friend who received your location response will be able to see the location in map."

Piximity:
"... an app that lets users capture and browse pictures nearby them – anonymously. The social media aspect is completely taken. The anonymous factor plays a big roll in what's

uploaded – there's a lot of off-the-cuff photography with little editing. Using location to help sort pictures allows users to discover pictures in their vicinity as well as have a live photo stream at a certain location."

2. User Guide

In order to run this application, a device running on Android 4.0 (Ice Cream Sandwich) or higher must be used. The device must have an active internet connection, as well as a camera. GPS is not required to run this application, but is highly recommended for certain features to work correctly.

Startup

When the application is initially launched, a splash screen (Figure 1) will appear temporarily. Its duration will depend on the application's loading time. If there is no user session active (i.e., no user currently logged in to Wau using this device), the user will be directed to the 'Login' screen. If there is already an active user session (i.e., a user already logged in to Wau on this device and hasn't logged out), the user will be directed to the 'Main' screen.

Figure 1: Wau's splash screen at startup

Login

At this screen, the user is able to enter in their login credentials. Selecting the <<back>> key at this point will exit the application. Selecting the 'Log In' button with valid

credentials will direct the user to the 'Main' screen. An error message will pop up on the screen if the user is unable to log in for one of the following reasons:

- Invalid credentials (Figure 3)
- Failed internet connection
- Server error

If the user wishes to acquire login credentials, a new Wau account by selecting the 'Sign Up' button, which will direct them to the 'Sign Up' screen.

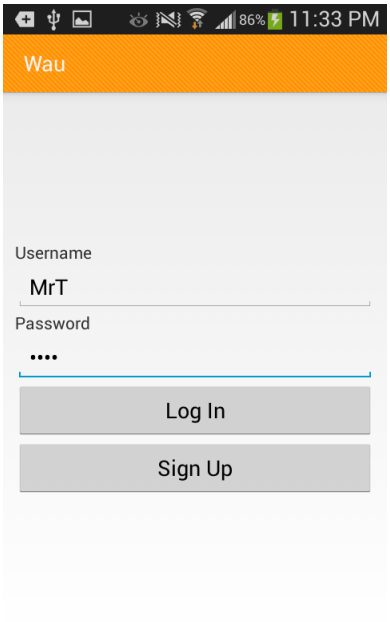

Figure 2: 'Login' screen

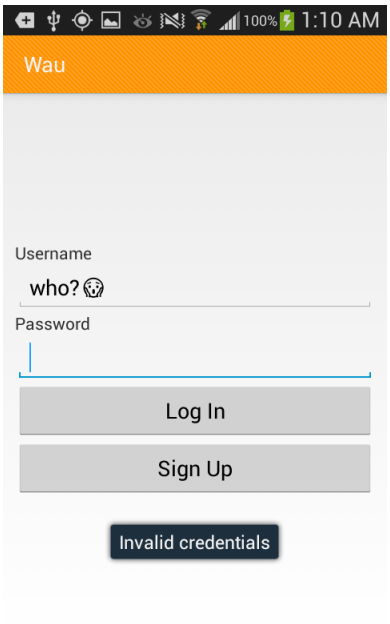

Figure 3: Error message for 'Invalid credentials'

Sign Up

At this screen, the user can create a new Wau account and acquire login credentials. Selecting the <<back>> key at this point will return the user to the 'Login' screen. Selecting the 'Sign Up' button will send the user's information to the server for processing. An error message will pop up on the screen if the user is unable to sign up for one of the following reasons:

- One or more fields are empty (Figure 5)
- Username already taken
- Failed internet connection
- Server error

Once the user successfully signs up, their credentials and information will be saved to the Parse database. The user will then be automatically directed to the 'Main' screen.

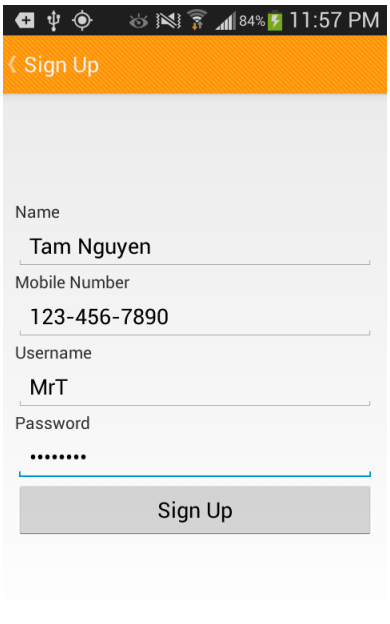

Figure 4: 'Sign Up' screen with all fields filled out

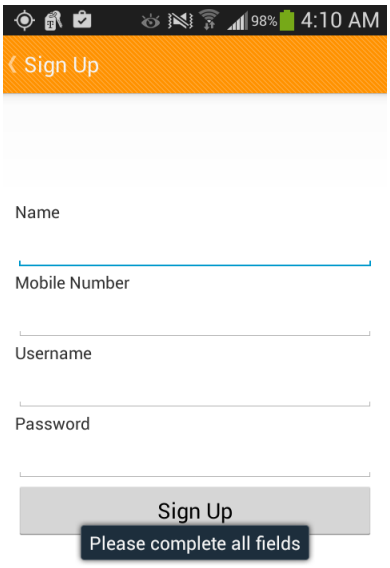

Figure 5: Error message for empty fields

Main

At this screen, the user is able to access and use the main functionalities of the application. Selecting the <<back>> key at this point will exit the application. This screen contains three tabs, each described in further detail below.

Map:

The middle tab, signified by the map icon. This tab is currently the default active tab when the user is directed to the 'Main' screen. In this tab, the user is presented with a map, provided by Google, as well as the user's current location (Figure 6). The accuracy of this location depends on the accuracy of the device's location service. Every 30 seconds, the user's coordinates are updated to the Parse database (if the user exits the application and leaves it running in the background, then this time increases to 3 minutes).

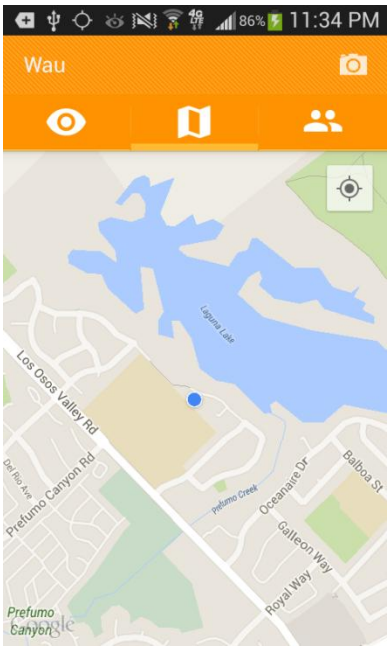

Figure 6: Map view containing user's location

This map can also contain markers (Figure 7), representing the coordinates of friends who have agreed to share their location. The location of these markers are updated at the same rate described earlier. Selecting a marker will display the name of the friend at that current location, and will also bring up options to use Google Navigation and Google Maps. These markers currently have a default time to live (TTL) of 5 minutes, after which the marker will be removed from the map.

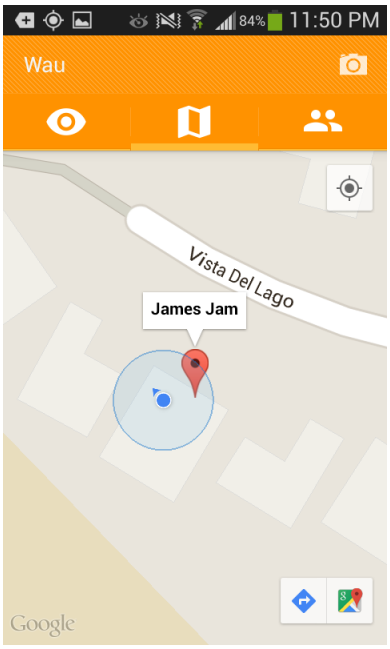

Figure 7: Map view with a friend marker selected

Initially opening this tab will zoom the map view to encompass the user's location, as well as all other markers, if present.

Friends:
The rightmost tab, signified by the people icon. This tab displays a list of all current friends. For a new user, this list will initially be empty (Figure 8). Friends can be added or removed in the 'Edit Friends' screen via the 'Menu'. Selecting a friend's name can do one of several things:

- Send a pending location request to the friend (e.g. 'Billy The Kid' in Figure 9).
    - The friend will be notified appropriately via a message under the sender's name in the 'Friends' tab.
- Accept a location request from the friend (e.g. 'Jody' in Figure 9).
    - Both users will begin sharing each other's locations for the duration of the default TTL (e.g. 'James Jam' in Figure 9).
    - This action is instantly reflected in the 'Map' tab.
- Cancel a pending location request or an existing agreement.
    - Both users will stop pulling the coordinates of one another if they are already sharing locations.
    - This action is instantly reflected in the 'Map' tab.

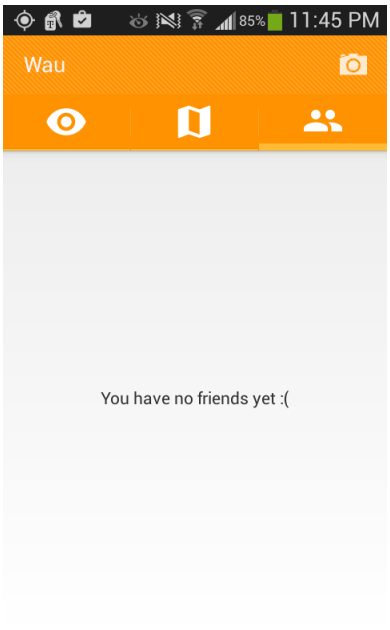

Figure 8: Empty friend list

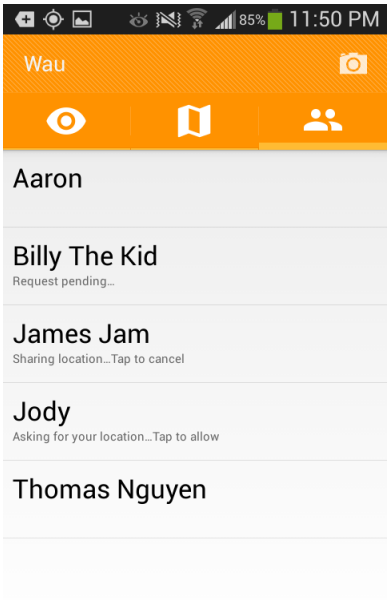

Figure 9: Friend list with active location requests

Nearby Photos:

The leftmost tab, signified by the eye icon. This tab displays a grid view (Figure 10) of all public photos whose coordinates, at the time of upload, are within 0.005 miles (26.4 feet) to the user's current location. Photos can be uploaded via Wau's 'Camera' feature.

Selecting a thumbnail will bring up a full screen view of the photo (Figure 11). Photos are sorted by descending upload date.

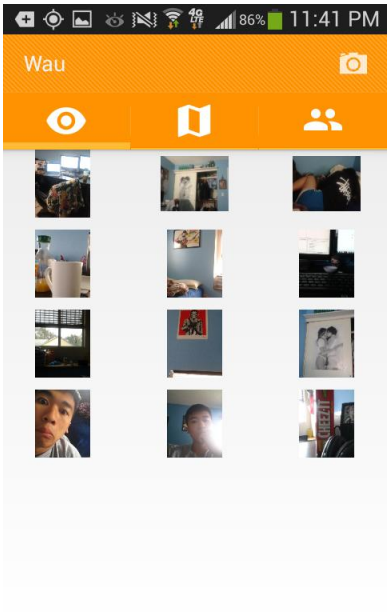

Figure 10: Grid view of nearby photos

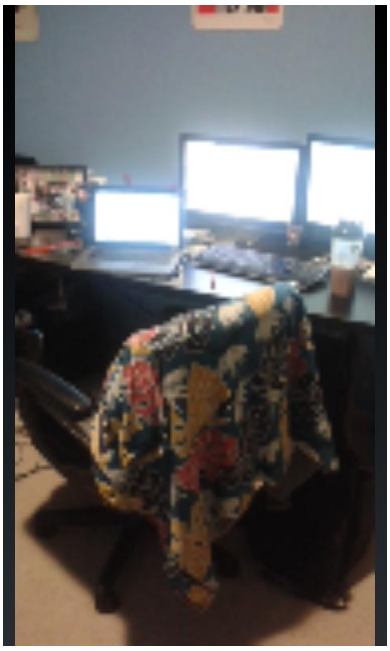

Figure 11: Full screen view of a photo

Camera

This feature can be accessed via the camera icon located at the top, right-hand corner of the 'Main' screen. Upon selecting the icon, the user is prompted to select either 'Picture' or 'Video'* (Figure 12). Selecting either will bring up the device's default camera application and set it to the appropriate mode (Figure 13). After taking the photo/video, the user will be prompted to either discard or save it (Figure 14). Discarding will bring the user back to camera application to take another photo/video. Saving will upload the photo/video to the Parse database, and store the compressed file along with the user's current coordinates, upload date, and file type (photo/video).

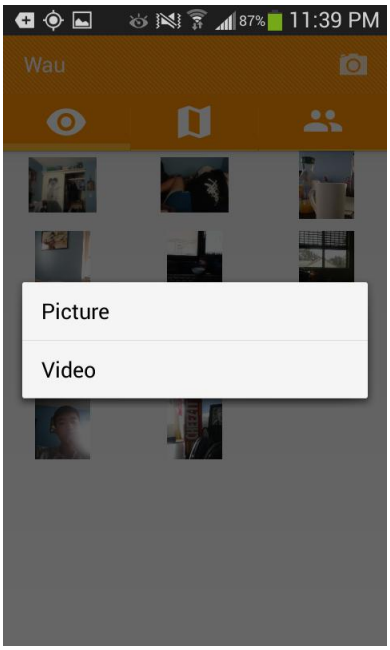

Figure 12: Options after selecting the camera icon

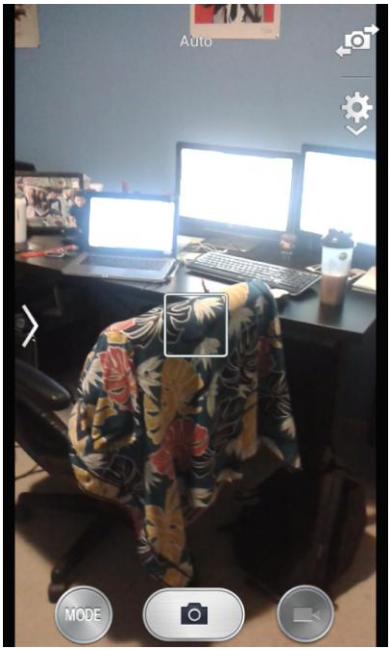

Figure 13: Device's default camera application

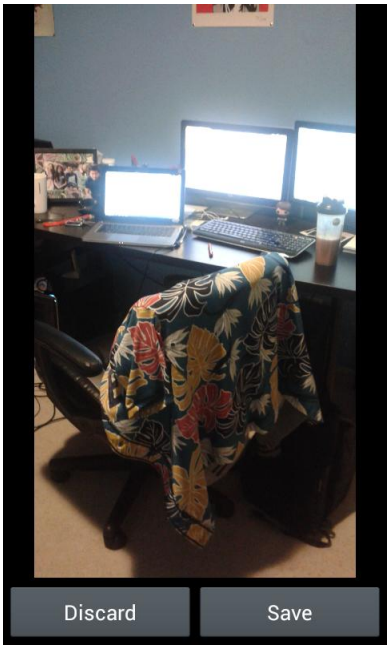

Figure 14: Options to discard or save the photo/video

*video functionalities have not yet been fully implemented in Wau.

Menu

This feature can be accessed at any time in the 'Main' screen by selecting the <<menu>> key/button on the device or in the application (depending on device's version of Android). A menu will appear (Figure 15), allowing the user to select one of two options, each described in further detail below.

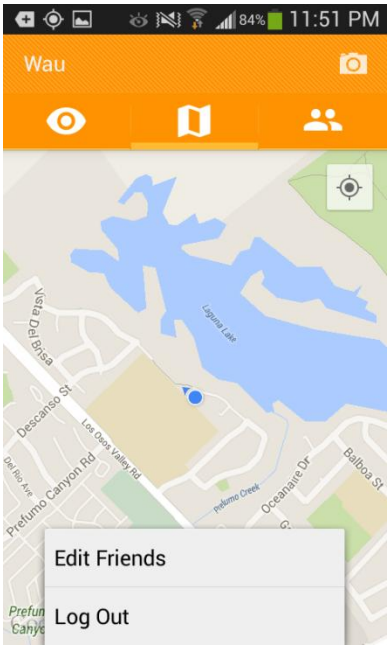

Figure 15: Menu items

Edit Friends:

Selecting this option will bring up an 'Edit Friends' screen (Figure 16), where the user can choose to add/remove friends. This screen contains a list of names of all Wau users. Selecting a user's name can do one of two things:

- Add to 'Friends' (e.g. 'Aaron' in Figure 16).
    - This is signified by a green checkmark next to the user's name.
    - The added friend will be listed under the 'Friends' tab.
- Remove from 'Friends' (e.g. 'Al Tur' in Figure 16).
    - This is signified by an empty checkmark next to the user's name.
    - The user will be removed from the 'Friends' tab list.

Selecting the <<back>> key at this point will return the user to the 'Main' screen.

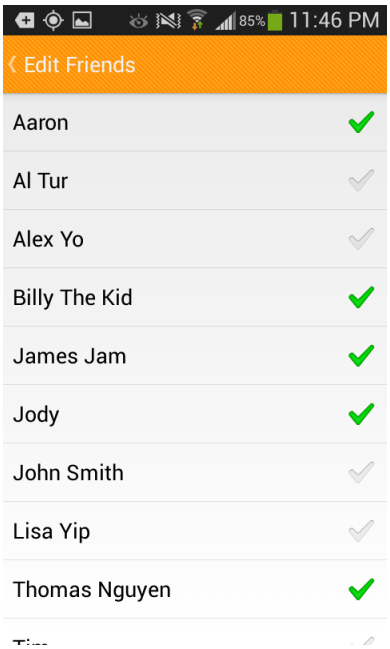

Figure 16: 'Edit Friends' screen

Logout:
Selecting this option will end the user's active session and direct the user to the 'Login' screen.

3. Design and Implementation

The initial design consisted of only a friend-finding feature. This quickly evolved with the incorporation of the location-based photo-sharing feature. With multiple features in one app, it made sense to create multiple tabs. Not only would it keep everything neat and on one 'page', but it would create a smooth and easy-to-use user experience. Originally, labeled with text, I decided to have each tab display a unique icon, which I got from one of Google's web pages. The decision to use icons over text was an easy one, since icons are more universal and easier to interpret, whereas text would limit the user base to a specific language.

While designing the UI of the application, the choice of color was a big question for me. I wanted my app to look unique, yet appealing. I originally wanted the theme to be a shade of blue, but that was the most commonly used color in apps. Next was a light green, but Yik Yak, popular among college students, had already defined itself with pastel green. Red was my next choice, but that color seemed for fitting for a food app. After doing a little bit of research, I settled with orange, since it wasn't being used as much, and it had a friendly tone to it. Once I decided on the color, I used an 'Android

Action Bar Style Generator' to help create the UI of my app. I played around with the style choices it offered in order to get what I have now.

The 'Login' and 'Sign Up' screen were my initial obstacles when writing the code. I knew I wanted to use Parse as my backend service, since it offered a very easy-to-use API. Parse's database was also quite reliable, and more than sufficient for my prototyping. Given this, I tried looking online for Android guides on how to make a login and sign up screen that incorporated Parse. I soon found one, and customized it to my own needs.

The next obstacle was designing the databases, from which my app would store and use information. I have a 'User' database/class (Figure 17), which is used to store most of my user's information, such as login credentials, personal information, and coordinates. I made this class public read-only, and gave read/write access only to the user. This prevents other users from tampering with other users' information without permission. Another primary purpose for this class is to store the friend relationships of each user. This is something that Parse handles for you very nicely. I also have a 'Multimedia' class (Figure 18), which is used to store all the uploaded media (e.g. photos and videos), as well as their upload coordinates and creation dates. I also made this class public read-only in order to prevent users from deleting or modifying other users' uploads. The last table I have is the 'LocationRequest' class (Figure 19), which is used by my app to handle location requests between users. I had to make sure to give read/write access to the public, otherwise users would be blocked from changing requests made by other users (i.e., a user who wants to accept or deny a location request). This class consists of a column for the TTL and another for the time accepted. Both are used once two users have agreed to share locations with each other. This is my way of only temporarily allowing users to share locations.

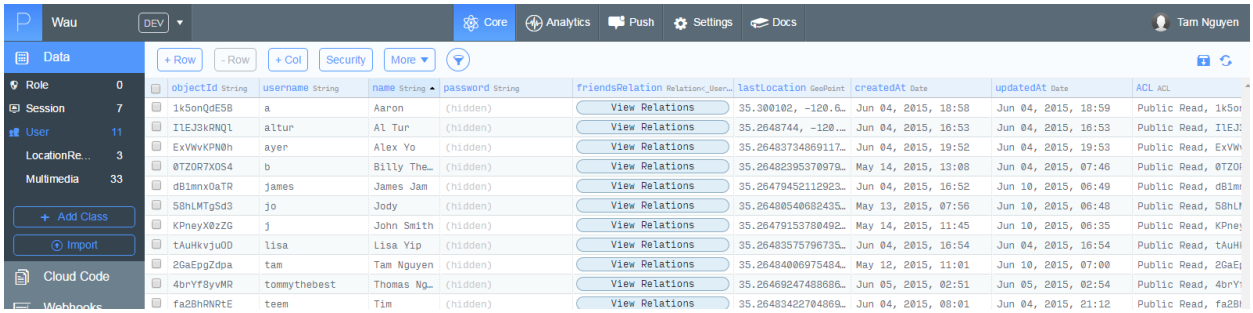

Figure 17: Parse User class

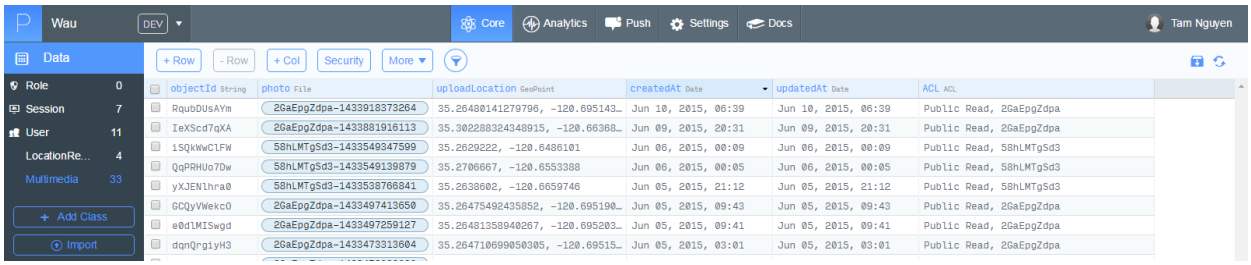

Figure 18: Parse Multimedia class

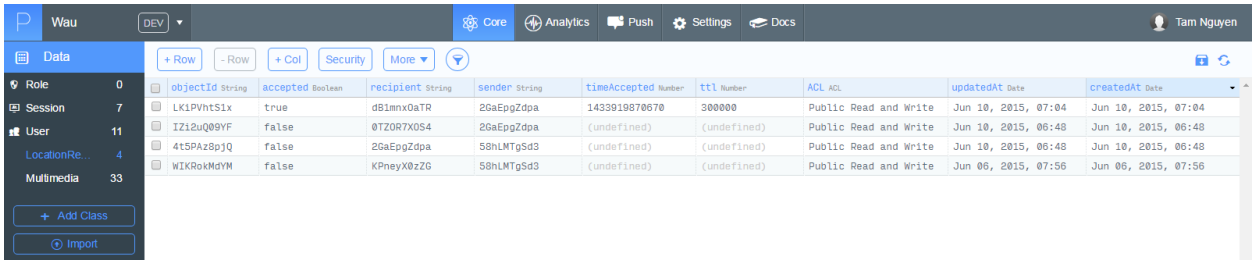

Figure 19: Parse LocationRequest class

Another design choice was having a grid view for the nearby photos. This was the most logical option, since I thought it would feel more natural to interact with adjacent thumbnails, rather than a list of a pictures. A difficulty when implementing this was figuring out how to decode the file data stored on Parse and convert it into an image in my app. Compressing the image and storing it is a lot less complicated than the other way around. Luckily I was able to find a tutorial online that helped me achieve this. The only problem however, is that I have yet to figure out how to maintain the quality of the photos after decoding it from byte data. Thus, every time a photo is expanded to full screen, it looks blurrier than when it was originally taken.

When planning out the friend-finding feature, I knew I wanted to use the Google Maps API. Given that I was working in Android, it made sense to go this route since it would make everything a lot easier to implement, and help would be readily available via numerous guides. The available APIs made it simple to display a map and work with coordinates. However, I noticed I had to convert between multiple coordinates types many (e.g., ParseGeoPoint, Location, LatLng), which I think is very unnecessary and time consuming.

Overall, throughout the design and implementation of this app, I aspired for clean, efficient code and UX. Many times I would go back and refactor a class or variable name just so my code stayed consistent and organized. I made sure to understand the differences between an 'Activity' and a 'Fragment', as well as their similar lifecycles. I researched many Android coding and naming conventions, just in case someone else would wanted to look at my code in the future. Ultimately, I wanted my app to be scalable, and something I can build upon in the future.

4. Future Work

Due to time constraints, I wasn't able to fully implement all the features I wanted to have in this app, as well as work out some of the bugs. Thus, I've listed out the major features I wish to work on in the future and include in my app:

- Add a password reset option.
- Implement a standard mobile number format, and verify numbers via text.
- Make use of the registered mobile numbers for the 'Edit Friends' screen, such that it only lists Wau users who are already in the user's phone contacts.
- Implement pending/accepting/denying friend requests functionality.
- Implement push notifications for friend/location requests.
- Allow users to send a temporary photo of their surroundings to the user requesting their location.
- Include a 'Settings' option as one of the Menu items. This will open a new Activity where the user can edit their user information and privacy settings.
- Implement video loading and viewing, as well as fix the decoding quality of photos.
- Implement a way to refresh (i.e., pull down to refresh)
- Fix random crash errors when initially opening the application.

5. Conclusion

Ultimately, I learned a lot throughout this whole process. I firmly believe that I managed to complete the majority of my project goals. The basic functionalities of the application were completed in time, such as the friends list, location requests, and location-based photo-sharing. Although not all of my desired features made into the final cut, I still plan on working and improving on my application in the future. I had a lot of fun working in Android. Learning to integrate Parse and Google Maps into my application was also a valuable learning experience. One that I'm sure to encounter again in the future. I had to apply many of the time management, organization, and research skills I learned in my classes and push them to the limit in order to complete this project.

6. References

Action Bar Style Generator:
http://jgilfelt.github.io/android-actionbarstylegenerator/

Android API:
http://developer.android.com/reference/packages.html

Android friend list and photo capturing tutorial:
http://teamtreehouse.com/library/build-a-selfdestructing-message-android-app

Android image loading tutorial:
http://www.technotalkative.com/android-load-images-from-web-and-caching/

Android login/signup tutorial:
http://www.androidbegin.com/tutorial/android-parse-com-simple-login-and-signup-tutorial/

Google icons:
http://www.google.com/design/icons/index.html

Google Maps API:
http://developers.google.com/maps/documentation/android/

Parse API:
http://parse.com/docs/android/api/

Piximity:
http://tech.co/piximity-young-developer-location-based-photo-sharing-app-2014-11

WhereUAt:
https://itunes.apple.com/us/app/whereuat-share-your-location/id918439814?mt=8&ign-mpt=uo%3D4