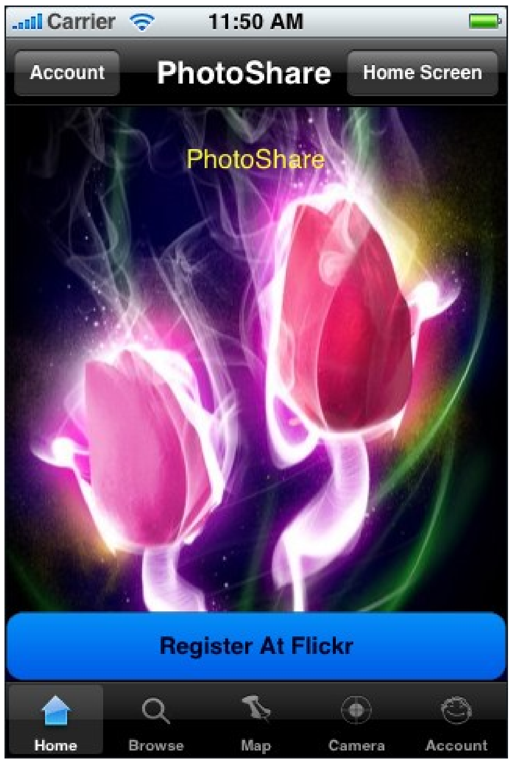
Final Report
PhotoShare (iPhotoShare)

COMS E6998
MOBILE COMPUTING WITH IPHONE AND ANDROID

Authors
Waseem Ilahi (wki2001)
McClain Braswell (mkb2014)



http://www.youtube.com/watch?v=Ud2AaENSpyc

Introduction

You are visiting a new city or just going to a place you like. You want to take a picture of what you see. It can be a monument or a masterpiece in a museum or anything that catches your eye. You launch PhotoShare, take your picture, and it locates you using the phone's built-in GPS and automatically tags and uploads the picture to the web.

Now you choose to see pictures that other people have taken from the very spot where you are standing. You can see the Statue of Liberty from every angle, by day or night, under snow or sunshine, by only going there once! You can even see back in time to watch the progress of a famous new building as it is built from the ground up.

In the future, this application could even work with Microsoft's PhotoSynth technology to create large-scale 3D collages to which every user can make a contribution.

The key to such an application is having a large user base. Therefore, we decided to develop using Apple's iPhone rather than Google's Android, since the iPhone has already become a commonplace device. We utilize the iPhone's built-in GPS and data connection to tag every photo with geographic coordinates and upload them to the web. We also use the Google's map API to provide a map of the user's location with markers to show where other users have taken photos. To host and query photos, we use Flickr, which provides free hosting and a free API with support not only for tagging photos with geographic and timestamp information, but also for searching within a certain radius of a given location.

This is not, however, a Flickr application. We require the use of a Flickr account for hosting and querying, but we do not provide a full featured Flickr interface. Users are not be able to organize their photos, browse other user's albums, or add or remove friends. However they can do that from the web interface, accessed by tapping register button that essentially takes you to flickr.com. While Flickr has not yet released an official full-featured iPhone application, there are third party projects in the works, and we do not feel the need to create another.

Instead, the purpose of our application is to change the way our user interacts with his environment. Rather than spending his time searching through a friend's latest updates, we want our user to spend a few moments to check out what other people thought were interesting about the very place he is standing at any particular moment. We want our user to be able to view the same landmark through the eyes of other visitors who have come before and be able to upload the picture of what the user thinks is interesting.

Related Work

Flickr, PhotoBucket, WebShots, Picasa, and numerous other online photo sharing websites have become extremely popular on the desktop computer over the better part of this decade. Most of these services have released APIs and websites designed for mobile access. Even social networking sites, such as Facebook, MySpace, and Twitter now support "mobile uploads" of photos. The limitation most of these mobile services is that they require the use of a web browser. While the iPhone does implement perhaps the most robust and capable mobile web browser on a device of its class, there is nevertheless an inherent inefficiency associated with using Safari to interface with these services.

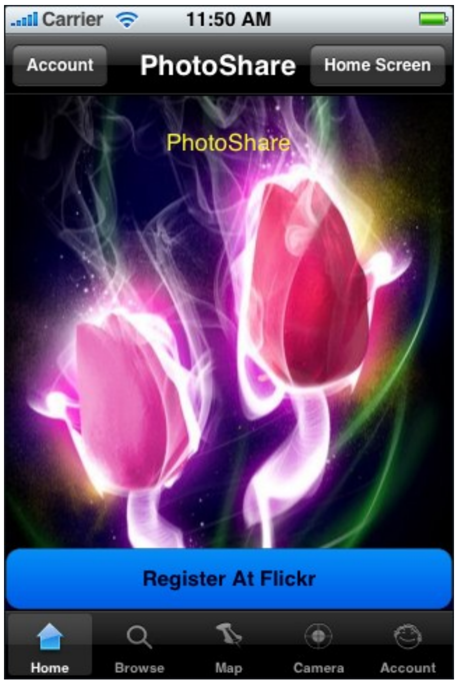
At the time of this writing, Flickr has not released an official iPhone Application, and when we began this project, there were no Flickr applications available on the AppStore. Since then, we have discovered two similar projects via web search: Darkslide[1] and Mobile Flickr[2]. Both projects aim to be full featured Flickr interfaces. Here is where our goals differ. We do not intend to be a full featured Flickr application. Instead, our goal is to provide a simple way to browse photos taken within a small radius of the user's current location and to quickly snap a photo and upload it complete with location tags. And save the photos that the user likes. We do not intend to provide services to upload full albums or to browse a certain user's albums. Our application is strictly location based. We do not want our users to be sucked into long browsing sessions ("flickr stalking") -- instead, we want our users to be able to share photos with others

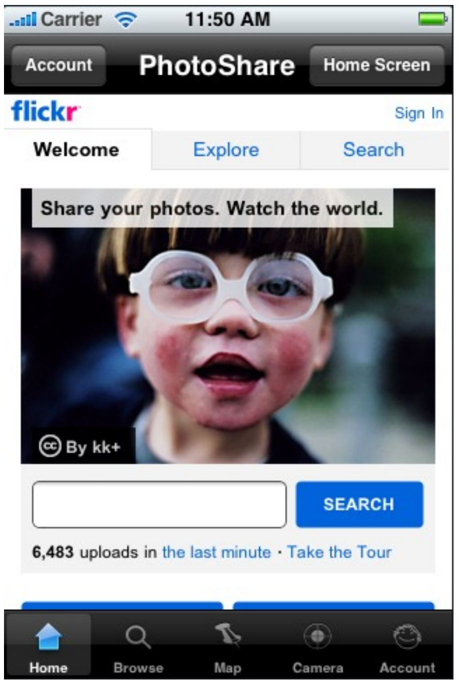
1 Darkslide Official Website, <http://connectedflow.com/darkslide/>, accessed March 12, 2009.
2 Apple iPhone School, <http://www.appleiphoneschool.com/2008/04/15/mobile-flickr/>, accessed March 12, 2009.

who have traveled the same path. We want our users to be able to travel back in time with a few flicks of their fingers to see what their surroundings looked like last week or last year, to see their surroundings from someone else's perspective. Our application is about interacting with our surroundings and sharing that experience with other users. One other advantage our application has on the others is that, our application lets the user view the photos as the markers on the map and the user can tap the marker to view the photos. Other applications don't provide this service.

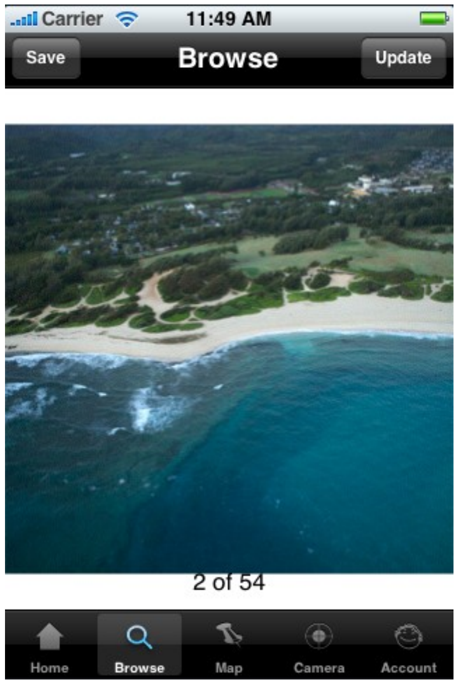
Usage Model

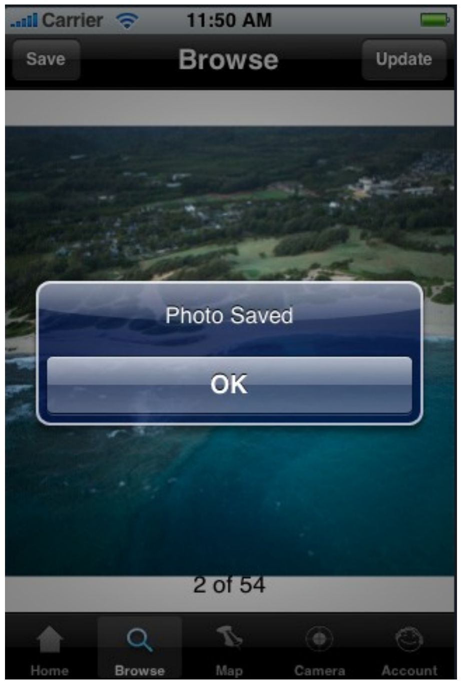
The application launches with a splash screen with a nice background and the application's name as the heading. From here, the user may access 4 other screens via tabs at the bottom of the screen. In addition to the Home screen, the user may choose Camera, Browse, Map, and Account. The account page houses all settings related to the user's Flickr account (required to use the application). Here, the user may sign in with his name and password. There is a register button at the bottom; tapping on it opens up Flickr.com from inside the application. The first time users can register to flickr from this site. After the user is done, he/she can simply tab the "Main Screen" button on the navigation bar to get back to the main home screen. There is another button on the other side of the navigation bar that takes you to the account tab, so you can sign in to upload the pictures. You can browse without being signed-in.



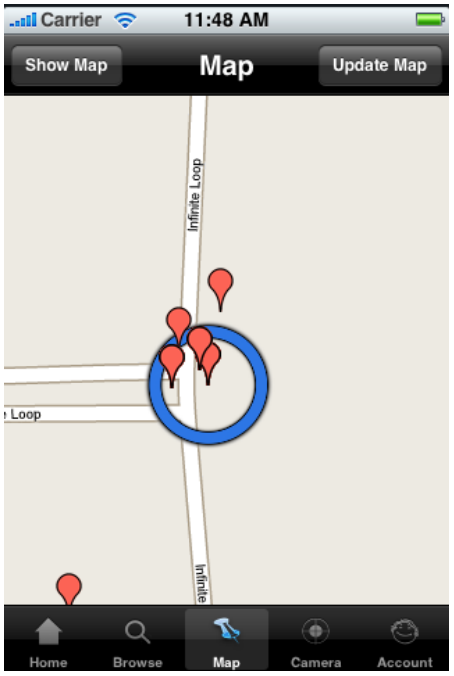


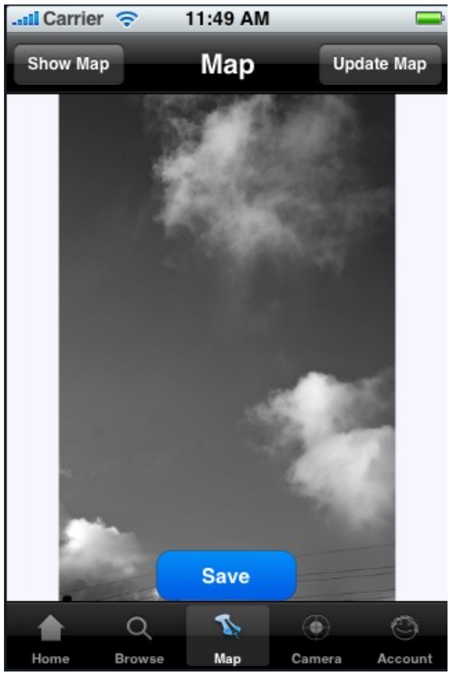
The Browse screen is where the excitement begins. From here, the user can scroll through the photos taken by other users standing in the same spot in a radius of 1 mile, only at different times. The user may also choose to save the image to his library by taping on the save button on the left hand side on the navigation bar. The confirmation comes up in the form of an alert popup telling the user that photo was saved successfully. On the top right side of the navigation bar is the button "Update". By tapping this button the user can get a new list of the pictures, based on the location he/she is at now. This feature is especially useful, if the user is in motion and wants to view pictures for different locations.



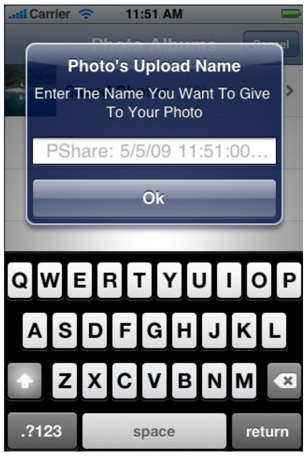


The Map screen is another exciting view. In this view, the user is presented with a full screen map centered at his current location, with a crosshairs showing the user's current location. The map is marked with markers representing photos taken nearby. The user may explore the map using more gestures, or tap on a marker to view full screen photos from the location. Again, from the full screen view, the user may choose to save the image or go back to the map view. User can update the map to center his/her location at any time by tapping the update button. This action also gets a new list of photos for the user, again as is browse new, based on the user's current location.

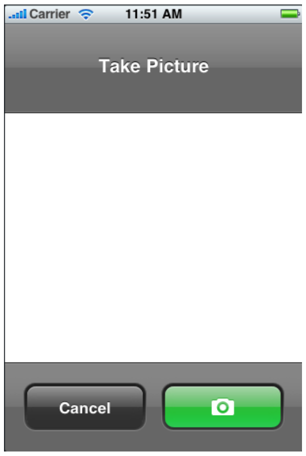


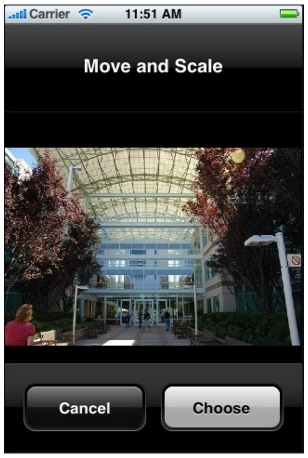


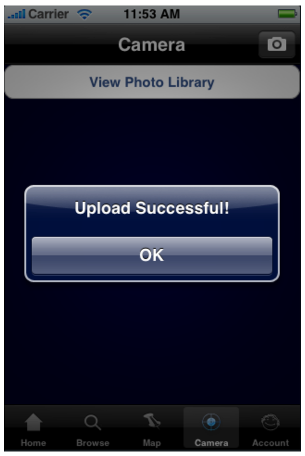
The Camera Tab is perhaps the first thing the user will want to do, and is very simple and straight-forward. There is a camera button on the navigation bar, when the user taps it, he popup becomes appears, asking the user to input the name for the photo. This name will be used as the title for the photo in the user's flickr account. If the user doesn't input anything, then the photo gets a name of the format "PShare: timestamp", where timestamp is the current time of the system. After that the built-in camera becomes active, and the user can easily aim and shoot to take a photo. If the user is satisfied with the shot, he simply taps use photo and the photo is on its way to Flickr, complete with location tags and a timestamp. If the user is unhappy with his photo, he can choose to discard it and take a new one. The user may also choose "cancel" to give up and return to the Home screen. There is also a 'view photo library' button; tapping on it does the same thing as tapping the camera button, except, this lets the user upload the photo from the photo library.



Photo's Upload Name
Enter The Name You Want To Give To Your Photo
PShare: 5/5/09 11:51:00...

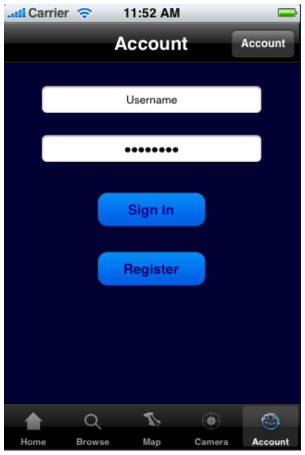


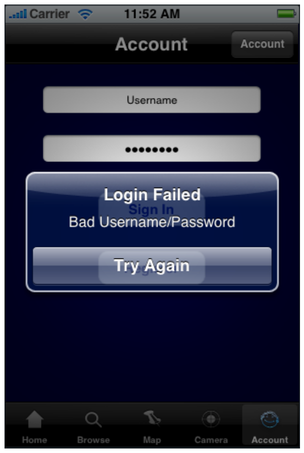




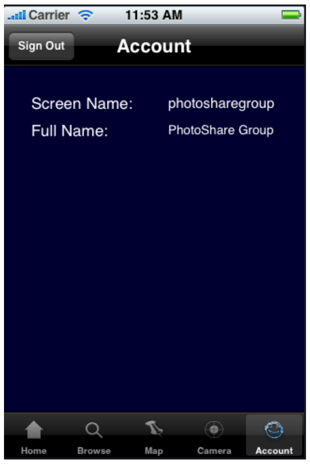
Upload Successful!

Finally, the fifth tab is the Account tab. On this tab, the user can sign in to the flickr.com, or register if they want don't have an account or they wants a new account. The register works the same way as it works in the main screen. If the login is successful, a new view pops in giving the user's username and the full name to let the user know that they have successfully logged in. If there is an error with the login process, an alert pops up with the appropriate error message, so user can try again right away or may be later.

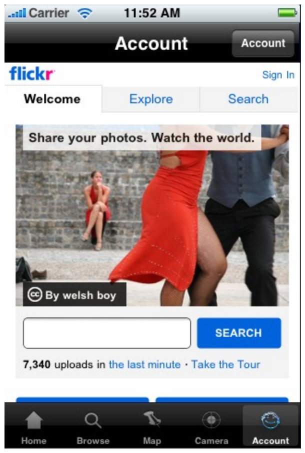




Login Failed
Bad Username/Password



Screen Name: photosharegroup
Full Name: PhotoShare Group


Carrier
11:52 AM
Account
Account
flickr
Sign In
Welcome
Explore
Search
Share your photos. Watch the world.
By welsh boy
SEARCH
7,340 uploads in the last minute · Take the Tour

The user may easily jump from one screen to another by simply clicking on the appropriate tab at the bottom of the screen. These tabs are always present for easy navigation, except when the user is taking a photo. On-screen dialogs will keep the user informed of every action, for example, if a file fails to upload, he may choose to try again or discard the image.

Architecture Design

To use our application, we assume the user meets the following requirements: to be able to upload t photos, the user must have a Flickr account (our application provides a link to register online for free at flickr.com); a reliable data connection, the user must be able to find an accurate reading (or semi-accurate estimation) of the current location via the iPhone's GPS, and in order to upload photos, the user must also have a fully functioning camera (or the user can upload the pictures from photo library. We use the Flickr API for storage and query functionality, in particular, for its ability to store latitude and longitude coordinates, and to search for photos within a certain radius of a certain point. We also use Google's map API to provide a visual representation of nearby photos.

Once a user has signed into his Flickr account, he may choose one of two basic modes: Camera and Browse. In Camera mode, the user simply points his camera at something interesting, snaps the photo, and after reviewing the photo, uploads it to Flickr, complete with title and location tags. The application automatically tags each photo with a latitude and longitude straight from the iPhone's GPS, as well as a timestamp, prior to uploading.

If the user prefers to see what other users have deemed interesting, he may instead choose Browse mode. In browse mode, the user may scroll through photos taken near where he stands, with the closest and most recent photos showing up first. The user can scroll through the photos by sliding his/her finger across the screen. If the user happens upon a photo he is particularly fond of, he may choose to download it. Additionally, the user may browse via a Map view. Here, the user will see a map of his current location with pinpoints showing where other users have uploaded photos. The user may then choose to view these images full size. The idea here is to let the images of other users inform the user's decision on where to go next.

For example, suppose our user is exploring a large area, such as Yellowstone National Park, but he doesn't know which direction to go, or he is looking for a particular location that he will only know when he sees it. Rather than exhaust himself by wandering through the whole park, our user may simply browse through the Map view to find a photo of a landmark he remembers from the last time he visited the park, or he may find an area with a particularly large number of photos and decide there must be a reason so many have shared photos of this area. Once the user has found an interesting photo, he can simply orient himself on the map and find his way there.

The key to our application is the Flickr API. We use the Flickr API to tag and upload the user's images and to query other users' images. Specifically, we will use the following API methods:

flickr.photos.search
Query nearby photos, passing current GPS coordinates as latitude and longitude parameters of the query.

http://api.flickr.com/services/upload/ with multiple parameters is used to upload a picture, the tags to this pictures are then added using the function descriped below.

flickr.photos.geo.getLocation
This function is used to get the location of a particular photo, given photo's pid. This function is used in our application to get the location coordinates for the pictures that we are going to put on the map.

flickr.photos.geo.setLocation
This is used for setting the coordinates for the picture that we are in the process of uploading.

Other helper functions used for user authentication:
flickr.auth.getFrob
flickr.auth.getToken
flickr.auth.checkToken

We also use the Google Map API. The calls to the API get the map, with the marker functionality. Utilizing that, we are able to put the markers on the map, where the picture is supposed to be.

Implementation

PhotoShare uses CLLocationManager Delegate to retain the latitude and longitude for user's current position. This service is initialized at the application startup, and remains active until the application termination. All the calls to APIs' use these coordinates to determine the user's location. Other than the location service, "flickr object" is also declared globally. This class is used by all the tabs in the application to access the functions that communicate with flickr. PhotoShare is a tabbed application. As you may know, it has five tabs used for separate functionalities. We also use QuartzCore framework to do the tab animations to produce the effect of slide transition, while the application moves from certain abs to the others, or maybe in the same tab, like when the image is swiped over and the new image comes in.

PhotoShare has its own implementation of functionality for communicating with flickr API and the Google maps API. The code used for getting the Map was borrowed from IGMC[3]. Some significant changes were made to the code to increase the speed and put the images in the markers. We also moved the Google API file directly to the resources folder of the project, rather than keeping it on a remote site and calling that site to return the map, which by the way didn't utilize the latitude and longitude of the device. To overcome that, we made some changes (allowed by the owner, as stated in the comments section of the source) to use the API for our purposes.

The entire interface with the flickr API was implemented by us. All the methods were implemented as part of the flickr class. The URLs for pictures downloaded, were parsed in the XMLtoObject.m file, written by us. We also utilized the image picker delegate to get the Camera Tab's functioning going.

The Interface builder was used extensively as well. The objects created by IB were then linked to the IBOutlet objects inside the objects in the controllers in the xcode. So, both the resources were utilized to get the maximum output for the application's graphics.

[3] IPhone-Google-maps-component , < http://code.google.com/p/iphone-google-maps-component/>

Evaluation

PhotoShare is basically a network based application. The main factor for how usable our application will be is the speed of the data connection. If a user cannot upload a photo or browse thumbnails of other photos in a timely manner, then this application will be of no interest to him. Unfortunately, we are at the mercy of the service providers on this point, but it is our experience that the data connection is certainly sufficient at least in heavily populated areas where our application will be most intriguing, and where there are more likely to be large numbers of photos to browse.

Our application also requires an accurate GPS reading, which again, is something that is out of our hands. It has been our experience that the GPS is accurate every time we have used the application.

The Upload and sign in is pretty fast and doesn't make the user wait long. Browsing is fast enough as well. For a reasonable number of pictures, it doesn't take very long. Map functionality however is not that fast. It works with 100% accuracy and puts all the markers of the map and after tapping the marker we can get the image instantly. The slight problem is with a rather slow population of map with the markers. Since the location for each photo is obtained from flickr separately, doing it for all the photos using only one thread sometimes takes fifteen seconds for all the markers to load. We partially took care of the problem by creating a separate thread for each marker and then killing the thread after putting the marker on the map asynchronously. There are some issues with this implementation that haven't been resolved yet, therefore this implementation is not in our stable 'release' yet.

Other than this, the rest of the application is pretty good and using the xcode built in memory leaks checker, we determined that our application worked pretty well.

We used the application on both the simulator and the iPhone. The stable version (presented at the demos in class), works fine on both and all the functionality can be used on either platforms.

Limitations

As it has been mentioned above before, PhotoShare depends upon a network connection. If there is no network then there is no point in using it. Also, the users need to let PhotoShare use the current location, in order to upload and download the photos. If the user doesn't do that that again PhotoShare is pretty much useless in that case. There is one more thing; PhotoShare currently doesn't let the user do advanced queries for photos. We originally planned to give the user options to queries photos from a certain time period.

Member Contributions

Waseem Ilahi is a Master's Student in the second semester of Software Systems track in COMS. Waseem Designed the UI Structure and implemented the controllers for each tab. Waseem also wrote functions to communicate with the flickr API, written by his partner. He worked with the Google Map API. Putting the markers on the maps at the location where the photos are suppose to be and assigning the photos to the markers so they can be seen by tapping the markers, was done by him. The work in IB was also done by him. The calls to appropriate flickr API functions at appropriate places, and the function that handles the response that is returned from flickr on the try to sign in was implemented by him. He also worked with the flickr API, to add some more parameters to the function calls and to get the user information parsed out after successful login and put in on the Account Tab for the user. Also implemented the logout functionality for flickr API.

McClain Braswell is a Master's student in his final semester of the Software Systems track. His interests include Computer Vision, Graphics, and interface design. He earned his undergraduate degree in English and Math at Columbia, and worked designing web applications prior to returning to school. McClain wrote the flickr API functionality from scratch. The function calls to the API for getting the user authentication, photos, and location for the photo and to upload the photo was written by him.

Lessons Learned

Most important of all, have a contingency plan prepared right away. A member left right after we decided what we wanted to do for the application. Secondly, do a little research on the resources that you think exist for you to use them. We assumed there was a very good implementation for us from flickr to be used for iPhone application development, but a it turned out, there was none.

Since it was our first application in objective c, there was a learning component involved, so the second time around and for future applications, we don't have to worry about that, since we have the basics down very well and know how the language, most frameworks and the SDK works and how to integrate the IB with the code and vice versa.

Conclusions and future work

Even with its limitations, PhotoShare is a pretty complete application, and we are very confident about it. There were ups and downs; however we managed to get a pretty decent implementation by the end. We managed to keep the functionality simple as promised originally.

Even though the application is complete, we may still want to add certain functionalities to the browsing. We can let the user choose the timeline for getting the photos from flickr. May be even let the user change the radius of search a little bit. We would also want to get that asynchronous map population working properly.

Acknowledgements

Flickr API Documentation:
We used Flickr's online API documentation and how-to pages to learn how to interface with Flickr's services via HTTP requests.

ObjectiveFlickr Project:

The ObjectiveFlickr project provided some demo applications with source code to help us understand how to use Objective-C to use HTTP requests to interface with the Flickr API.

We used the source code from iphone-google-maps-component as basis for the Map functionality for the application.

References

ObjectiveFlickr, <http://lukhnos.org/objectiveflickr/blog/>

Flickr API, <http://www.flickr.com/services/api/>

Iphone-Google-Maps-Component,
< http://code.google.com/p/iphone-google-maps-component/>

IPhone SDK Development by Bill Dudney and Chris Adamson

Apple Documentation, <http://developer.apple.com/documentation/>

Google and all that there is on it ☺

Code Appendix

Download the sources from the Google code site and build the application to try for yourself.
http://code.google.com/p/iphotoshare/

The three things declared globally are initialized here:

```
- (void)applicationDidFinishLaunching:(UIApplication *)application {
        [tabBarController setDelegate:self ];
   [window addSubview:tabBarController.view]; count =0;
        flickr = [[flickrapi alloc] init];
        self.locmanager = [[CLLocationManager alloc] init ];
        [self.locmanager setDistanceFilter:1.0f];
        [self.locmanager setDelegate:self];
        [self.locmanager setDesiredAccuracy:kCLLocationAccuracyBest];
        [self.locmanager startUpdatingLocation];     tab_loc_count = 0;
}
```

In the code part of the development, the files handle various functionalities as follows:

Main file: PhotoShareAppDelegate

Flickr API:
md5.h, md5.m, XMLtoObject.h, XMLtoObject.m, flickr.h, flickr.m

Map API:
CallBack.h , Callback.m , MapMarker.h , MapMarker.m , MapWebView.h , MapWebView.m , MapView.h , MapView.m

ViewControllers: (.h/.m)
HomeViewController, BrowseViewController, MapViewController,
CameraViewController, SettingsViewController

The End