# What’s Up App

## Summary

For our project, we were selected to do the “What’s Up” app of the Fall 2014 semester. Essentially, this app was thought up as a method of getting communities to be more involved with one another, allowing members of the community to post events and goings-on near their location for other people to attend and collaborate with them on. As part of the requirements, this application had to be customizable by kids after the fact, so there was a hard requirement that at least the UI’s appearance be customizable, with any added functionality coming at their request for modification.

## Mockups and Design Process

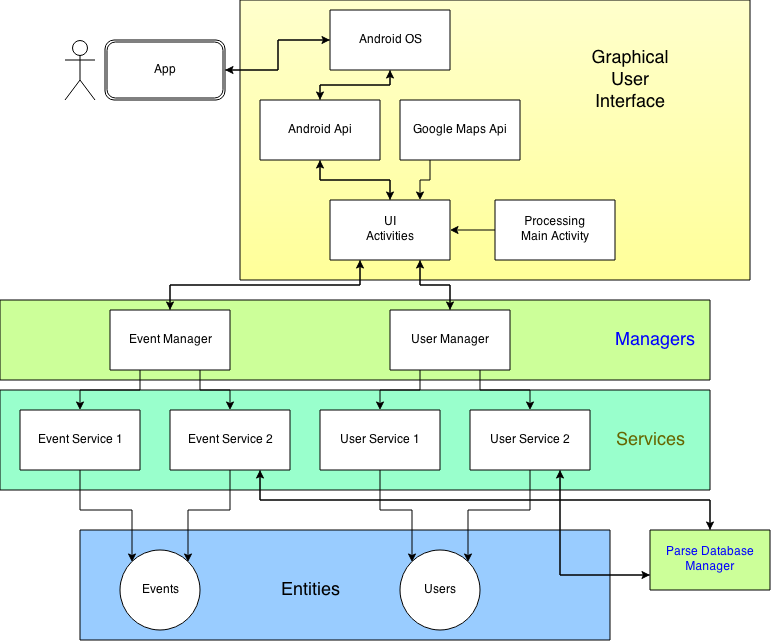
We started off with a notion of an application that you would have to log into, where a user would be presented with a login screen and sign in using their email address along with a password they could specify at sign up time. Once logged in, the user would be able to navigate through events, select a section and view events that were occurring nearby that applied to that section. At this point, the user would be able to view details specific to that section by clicking a little popup that would appear above the map marker, and a modal window would pop up, allowing them to view details about the event (title, description, how many people were attending, etc) and allow them to click “attend” as well, to increment the counter and push that notification to all the other phones using the app.

After the first round of mockups and speaking to the students, we surfaced a ton of other feature requests that unfortunately we were not able to fulfill entirely, but we implemented as many of the requests as we could. One of which was the ability for a user to set up an event right from their phone by long clicking on the map. The use case for this was someone who didn’t know an address directly, but did know approximately where the location they wanted to host an activity was. This also applied to being able to host a location where you’re currently at. This presented an interesting problem in that if a user’s GPS is not currently enabled, we don’t necessarily have that data available right away to provide to our query engine to pull events or addresses nearby. These technical problems and more will be covered in the structure portion of the application below.

We came up with a loose idea of how to go about the aesthetics of the design, allowing one of the customizations provided to the students to be the primary color of the app. Furthering this, we additionally provided functionality to determine a color compliment. The notion behind this was that the student using Processing to customize the app would be able to set a primary color quickly and change the entire theme of the application. This was an interesting implementation, but proved to be a lot more work in the design and most certainly the development process.

## Architecture

We soon decided that we needed to come up with a well-stratified architecture to follow for the application development process. We decided on the following for an architecture for our application to follow. **Application** uses **Activities** which use **Managers** to expose functionality of **Services** which manipulate **Entities**. The full relationship can be seen in the figure below, along with the context of the android subsystem and APIs provided.



*Figure 1 - Architecture of WhatsUp*

The notion here is one of adhering to the SOLID principles of software development. Each manager is concerned with a single task. This task’s functionality is provided through a service contract (Interface), and implemented by a service. The service performs its operation and its operation only. The advantage to this model is providing an easy way to pull existing functionality out, replace with new functionality while maintaining the integrity of the application.

A small but useful example of this can be seen in our ToastManager class. The ToastManager class operates as a singleton, providing a common, shared set of functionality to our Activities. An example of the code can be seen in the code block below.

public void SendMessage(String message, boolean longDuration){

\_service.SendMessage(message, longDuration);

}

This function stores no global state, does not try to do anything outside of the scope of the job of the method - to send a message to the screen without disrupting the current flow of the application. The \_service variable in question is defined as an interface, *IToastService*. This interface is implemented on the class *ToastService*, which provides a simple layer of functionality but very valuable abstraction nonetheless, as seen in the class declaration below.

public class ToastService extends BaseService implements IToastService {

public ToastService(Context c){

super(c);

}

@Override

public void SendMessage(String message, boolean longDuration) {

Toast t = Toast.makeText(\_context, message,

longDuration ? Toast.LENGTH\_LONG : Toast.LENGTH\_SHORT

);

t.show();

}

}

With this functionality we implemented our methods for obtaining GPS information, saving user information, with threading being provided by the manager. Implementing Parse as a storage medium using this method required a bit more work, but provides you with a longer lasting application, more durable to changes and supporting an architecture allowing multiple developers to work in tandem on the project.

## 

## Integrating Google Maps

The app revolves around events and their location. The addition of Google maps allows us to use an incredibly useful set of features. A Google map contains information for geographical features, streets, navigation, addresses, coordinates, and a plethora of other useful features. We were easily able to add our own graphical markers to indicate different types of events.

Adding a Google map is a fairly simple process. First, you must register with Google to receive a unique API key. This key allows Google to identify your project and keep track of how you are using their service. Next, you have to add a fragment one of your layouts with the map identifier. The final step is to just set the layout with the fragment as the content view. The map will be displayed based on how you layout the rest of the XML in the layout file.

To interact with the on screen map through java, you simply create a map object and pull its ID from the XML fragment. The map object has methods which allow you to manipulate the location, animation, and location markers in the fragment.

One of the drawbacks for using Google maps was the immense processing it requires from the device. In order for the map to drag across the screen and interact with the user seamlessly, Google uses the OpenGL graphics library. Loading all the graphical textures into the video memory takes some time. This is all done on the main UI thread and ties up the entire phone while the map loads. Our original design accessed the event database after the map was set as the content view. This caused intermittent crashes based on how well the parse query performed. If the query was too long, the OS decided our app was holding onto the main thread too long and put the app down to rest. Eventually the query to the database was put into another thread to avoid tying up the main UI thread.

## Threading

One of the problems we faced came up when we were emulating on the original android emulator. As an aside, we have found a much, much better emulator in GenyMotion and we cannot suggest it enough to developers. It can be cumbersome to debug on a device, and having a solid, fast emulator to run unit tests on and the like can be a huge productivity booster. While debugging on this platform, it provided us the ability to simulate poor network connection conditions to see how our app performed. In certain circumstances such as initial login, we were unable to speed this up much as Parse handles most of that on the backend. However, for certain save calls (adding yourself as an attendee comes to mind), the user most likely would not want to be held up, waiting for a web call to complete before continuning with their usage of the app.

For this case, we kept with the notion of our existing architecture, but provided a new method for each threading case as it came up in our managers, naming the majority <ActionToTake>InThread. The method declaration of just such an example from the UserManager class can be seen below.

public void SaveInThread(final User u){

Thread t = new Thread(new Runnable() {

@Override

public void run() {

Log.Info("Saving user's updated location in a background thread.");

\_userService.Save(u);

}

});

t.setName("User Save Background Thread");

t.run();

}

This method had no callback, as it was not written with the intention of caring when it was completed. Mostly this method was created as a fire-and-forget method, something that we could fire off on regular intervals whenever something happened on the user object that we had to persist and keep, but was not vital to the application’s functionality at any given moment. The most obvious example of this came as a late-blooming requirement, the necessity to store the user’s last known location to the data store. We came across this requirement when first testing in the building there on campus and realizing that we sometimes could not get a solid GPS signal. This necessitated keeping the user’s last known location on file, so that the next time they logged in with the app, if we were unable to get a fix on their current location we would at least have a point on the map to center on and query against.

## Conclusions

### Phillip Halpin

Overall, I enjoyed working on this project. It’s the first time that I’ve ever spent any large amount of time working on an Android application, and was a valuable lesson in how the android development lifecycle works. I’m more well versed in the world of iOS and Windows Phone development, to say it was an eye opener is not saying anywhere near enough. I found a lot of pros in the android world relative to some of the other frameworks, but also some cons.

On the plus side, I found the development stack on android to be pretty solid. I noticed that Google has been working pretty hard on unifying a lot of aspects of the development process, mostly noticed when we were working within Android Studio. Gradle is a great building system, somewhat reminiscient to me of Grunt or something of that ilk. Android Studio is an outstanding IDE, I am a huge fan of JetBrains and the tools they bring to the table in things like IntelliJ (the base of Android Studio) and PHPStorm. If they keep going at this rate it will be an even more amazing development platform here in the near future.

On the minus side, I found the methodology Google employs on the UI level to be a little bit on the messy side of things. Granted, most UI frameworks don’t do the job much better, but the notion behind the Activities, while admirable, a little too close to older frameworks at this point - windows forms comes to mind. A better example in my opinion can be seen in some of the binding frameworks of today, a la AngularJS (another google product!) or on the application side, MVVM supported by XAML. Being able to bind a value against a context and have it automatically update is an amazing thing. This seems as though it would be doable in Android, but take quite a bit more code to accomplish.

**Keith Scheuerman**

Prior to this project I had only dabbled in Android development. I too enjoyed developing this app. Having only brief experience with Java, I felt this project had a steep learning curve to it. Phil was a great partner, and taught me a lot about architecting code. After taking this course I feel much more confident in the mobile environment. I'm not sure how it relates to software hardware codesign, but I do not feel like I wasted my time or money in taking it.

During my first attempt using the Android platform I found the activities and UI very confusing. I'm glad I was able to put most of my efforts into the activity and XML side of the app.