Waypoint: The Hike Tracker

Iñigo Beitia, Chloe Eghtebas, Brendan Ritter

May 13, 2013

1 Inspiration

Imagine you are on the side of a very lush green mountain sitting on a stone. You are watching the sun set over rocky hills listening to the sound of a gentle creek nearby. You may think to yourself, "This is so incredible, I'd like to share this peace and tranquility with the whole world." Or you may even find yourself so tired from your journey that you cant exactly find your way back. In either case, our product, Waypoint can help.

2 The Product

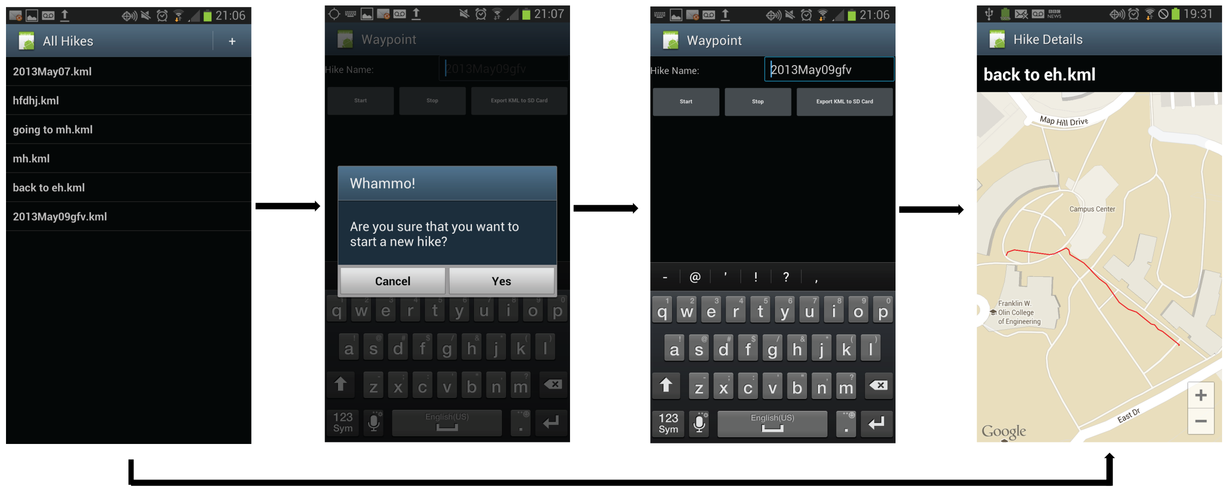
Waypoint is an Android mobile application that tracks your hikes through the device's GPS sensor. It allows you to create and will eventually allow to share hikes with your friends or search for public hikes previously recorded by other users. The application provides navigation controls for exploring previusly recorded hikes. In future versions, an "Explorer's Mode" will be developed that allows the user to navigate a path and receive vibration or audio alerts when their location is getting far from the recorded path.

3 Implementation and Results

3.1 Starting our Project

The first step of starting to code an Android application was to set up the developing enviornment. We used the Eclipse IDE with Andriod SDK. The use of Pivotal Tracker enabled us to use the agile development methodology for managing out software project. Our group split up into sections and started accomplishing individual tasks. Some of these early stage tasks included understanding and devloping sample applications making use of file I/O, accessing the GPS sensor, and implementing UI components.

3.2 User Interface



Are you sure that you want to start a new hike?

Figure 1: This figure illustrates Waypoint's UI experience.

The first screen that the user meets is a list of available hikes. These hikes are saved persistently on the device by making use of the SD card storage capabilities. From here, the user has the option to record a new hike using their device's GPS or view. If the user indicates that she wants to follow a pre-existing hike, she is taken to a detailed view of the hike, with a map of where the hiking path is.

If the user wants to be adventurous they can tap on the + in the top right hand corner of the screen.

A dialog box appears and confirms their desired action.

Next, the GPS logging can be started, paused, and stopped. When the user is finished with their hike, she/he simply has to hit 'done' and the application automaticaly loads all the collected data into a KML file. A flow chart of the processes are illustrated in Figure 1. At anypoint can the user go back, there is button on the Andriod hardwired to do so.

3.3 Program Architecture

Our program makes use of four main Java classes. Half of these classes deal with GPS data. One handles the collection of data and the other provides the user interface for the collection. Of the rest of classes, one provides the list view seen initially when starting the app and the other provides the detailed view of the hike when selected from the list view.

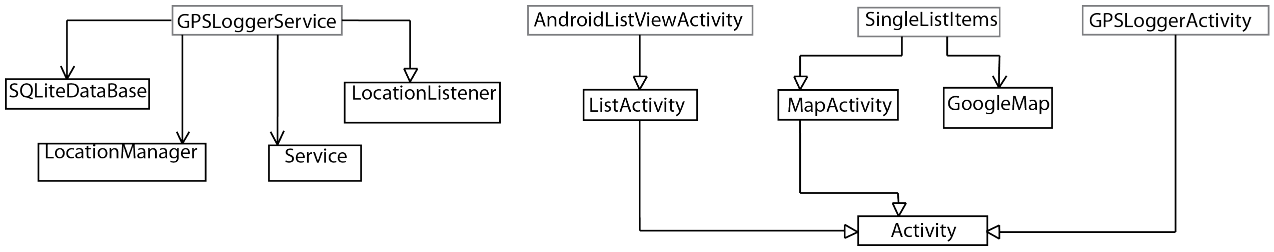


Figure 2: This figure shows the UML diagram of our product's code.

3.4 Location Manager

When researching location solutions for Android, we realized that most of the existing implementations were battery-intensive and most of the times connected to an online service. We wanted to create a custom location manager that fulfilled the following goals:

1. No need of network access, accessible in forests and other non-connected places.
2. Smart and customizable location updates for lower battery usage.
3. Easy to start and stop logging.

The result is a lightweight set of methods that can be easily instantiated as an Andoid Activity.

3.5 Persistent Storing

Because of the different rates at which locations are updated, we needed a way of storing a set of locations when the user is recording a hike. We considered using file writers that wrote single lines of a text to a file whenever the location changed. This proved to be unreliable because the user could remove the SD card or modify the file before completing a hike.

We decided instead to use a SQLite database with an archive for each single hike with each row representing a coordinate. This database is used for temporal data storage and is used to generate a KML file with all the coordinates. Using a database makes it fast and easy to add new individual updates and to retrieve all the entries.

3.6 Path Represenation

The previously generated KML file consists of hike metadata such as name, start and end times and distance as well as each coordinate in the path. To represent the data, we used an XML parser that allowed us to extract all the requiered information. We then created an overlay layer on our map view placing every point in the map and connecting the points to each other to form a visual representation of the path that the users can check when exploring the hikes.

4 Process Reflection

Reflecting on our process of completing this project, there was definately some strategies that we intended to follow but instead strayed away from. The follwoing subsections touch upon such examples of abandoning strategies and taking up the one that seemed most natural at the time[1].

4.1 SVN repo discipline

We had planned on applying our repo discipline skills and in the begining, we applyied housekeeping required. However, shortly after the start of the project we realized that SVN was going to be a nightmare to use with android developement. Android has runtime specific code that auto-generates and then saves but is required to run the project. This means that it is difficult to avoid a merge conflict. To make it even worse, android spreads its required files across many folders, also making it more likely that merge conflicts will occur, especially in the numerous xml files required to build the app. Very quickly it got so that each person had their own local copy and that merging became a tedious process which was definatley not a desired outcome.

4.2 Android Development

One of our goals for this project was to attain experience and learn about android application development. However, numerous software kits are needed in order to even begin development. The android API has extensive documentation, however, this hindered us more than helped because we had to spend our time going through all of the relevant documents. We feel that this decreased our momentum which is not in the spirit of agile development.

[1]For better or for worse.

On a different note, a process that worked well was the use of a physical android device (A Samsung Galaxy S3) over the emulator which came with the android SDK. Although it required an initial set up to get file transfer working, the physical device started the app faster. This was benificial because it communicated errors faster thus facilatiting quick debugging especially when working with a partner.

In the end we were capable of understanding basic file I/O and app structure, activity transition, communication with the phone's built in GPS, and utilization of the google maps API. We belive our goal, understanding the basics of the android system, was obtained. Despite this belief, practice makes perfect and further exploration could benifit our android application development skills.