Link

Get connected

Link App Progress Report

Justin Richard

Version 1.0

November 10th, 2016

Table of contents

Project Overview

The Link App is a mobile application designed to make it easy and convenient for people and groups to find each other. When the users are ready, they can enter sessions where they share their real time location with each other to make meeting up a breeze. A detailed description of how the app was designed can be found in the design document. This document is intended to give an outline on how development has went so far, and what features have been completed and what they look like.

Current Status

Currently the app is in a near finished state. There are a couple different screens completed which can be seen below in the work completed section, but they have yet to be polished to be the most visually appealing, and not all UI bugs and quirks have been ironed out. The main component being worked on right now is the ability to enter a created link session, and have the users locations overlaid on top of the map.

Work Completed

In the sections below I will outline the various features and components of the application that have been completed. All in all, lots of the work has taken more time than expected to complete, largely due to the learning curve required for Android development. Being new made me have to learn the core concepts and design patterns on the fly, which sometimes meant I would build something in a weird or incorrect way which would later need to be rebuilt in a different manner. The work so far reminds me very much so of an agile work style as opposed to the waterfall methodology. I think that the agile-style of work is more effective and reasonable considering the circumstances. Planning everything in advance would be near impossible given my lack of knowledge of Android development.

Authentication

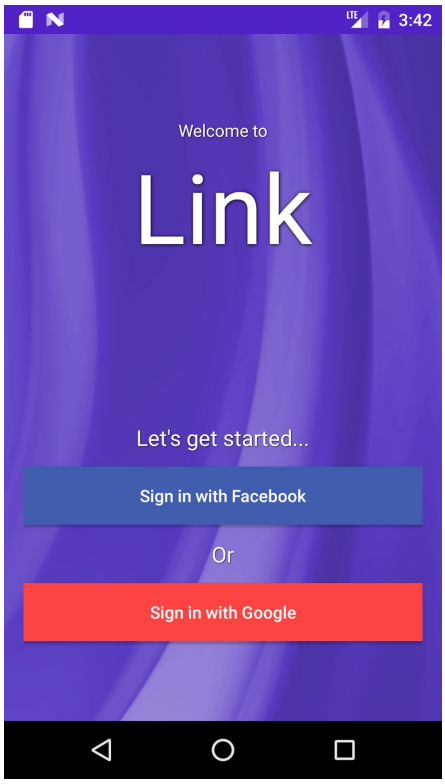

Figure 1.0 Login interface

The authentication section of the app arguably took the most time compared to other components. I created a user login interface just like the examples in the System Interfaces section of the design document. A screenshot of the interface can be seen to the right in figure 1.0. As in the design document, I continued ahead with only using Facebook and Google and login providers. All Android users should have a Google account already as a part of their device setup and Google Play account, and adding Facebook allows users to have a choice and connect with the majority of their friends since Facebook has a larger social media presence over Google Plus. I skipped creating a custom authentication service since I did not believe it would add much to the application and the learning curve, plus this App is meant to be social anyways so it only makes sense to be connected to social media.

Configuring the social media login proved to be tricky when testing the applications log in functions since lots of the errors provided were very generic with regards to incorrect client/api keys. I had to register the application in the Facebook Developers portal as well as enable the Google+ API under my account in order to be able to use the login methods. It became a bit tricky to set this up for the Google account because there are various different type of keys you can create for your account. The Google API provides API keys, OAuth 2.0 client ID's and service account keys. Furthermore, within the sections you had to clarify if it were a web-based key, or if it is a key for an Android client which would require you to specify a SHA-1 signing-certificate fingerprint restricting calls to only come from your app not a third party.

Once the keys were figured out, the authentication worked smoothly. Users are now able to sign in using Facebook or Google. The information stored in the database can be seen in figure 1.1.

userId	firstName	imageUrl	lastName	username
us-east-1:b2ed44ec-0b8d-	Justin	https://lh4.go...	Richard	jgoogle
us-east-1:a6ec0d84-d336-	Justin	https://sconte...	Richard	Justin
us-east-1:f6184c49-f89c-4	Shannon	https://sconte...	Donovan	Shannon

Figure 1.1 Users Table in Database

Home

After the authentication interface was complete, logically the next step is to create the landing area. This will be the home screen of the app where the most interactions will occur. This area is extremely important because a poor user interface could easily result in frustrated users. After looking into the Android material design documents which outline how things should look I gained a feel for some of the common design patterns. I also investigated some of the popular apps that people use that would need a similar interface and took note of how their application was laid out. In the end I opted for a tabbed layout, each containing a list of items. This style very closely replicates the style of Facebook Messenger, which most users are likely to be familiar with. Even if users are not already familiar with it, the style is straightforward, and proven to be efficient by the Facebook researchers. It's logically best to not stray too far from what the user interface specialists have decided is a good idea, when I only have minimal user interface design knowledge from a previous course or two. You can see what the home screen looks like in the three sections below that outline each of the tabs.

Username selection

When users load the app, It is checked that the user has a username set. Usernames are set by the user and are how other users can add people as contacts. In the future you will be able to add people directly from your friends lists in Facebook and Google, but to get things running the username method was selected since it works for all cases. If a user does not have a username set, they are prompted to enter a username and may not continue until they select a valid name.

Usernames are unique amongst users, so if you enter a username that is taken then you are alerted that the name is in use and are prompted to choose a new one. The username is stored alongside the user's userId and personal information, as was seen in the authentication section above in figure 1.1. A sample of the username selection screen can be seen in figure 2.0.

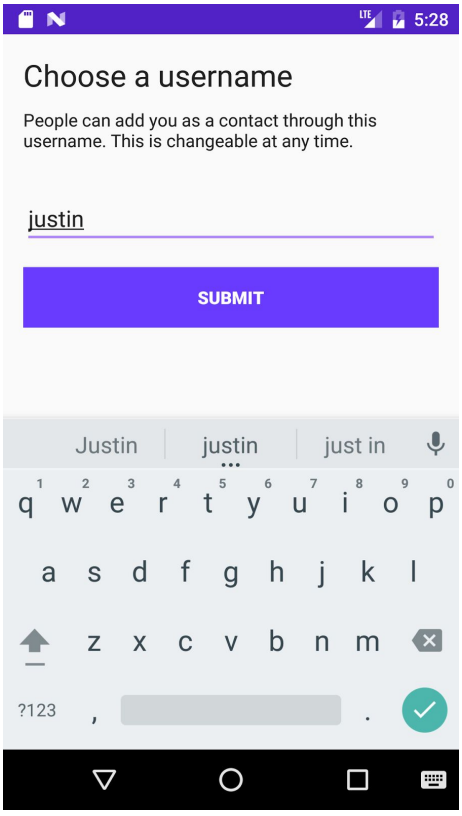

Figure 2.0 Username selection screen

Contacts

The contacts tab is used to display a list of users the user can create a session with. Similar to a contacts list on anything, it will display other users that also use this app, that you have deemed you have a connection with. As you can see below in figure 3.0, the contacts list simply shows the user's profile image, their preferred real name, and the username they chose. Selecting a contact opens up a drop down menu which lets you choose to either start a new Link session with that user, or to remove that contact.

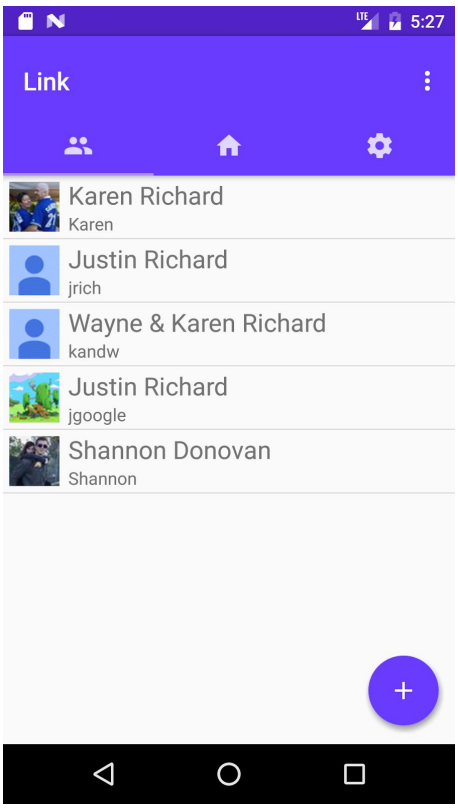

Figure 3.0 Contacts tab

Contacts are stored in the backend in a simple way in order to avoid having to deal with approving/denying contacts, and sending requests to the other users as this functionality is not of high priority, and can be added at a later point. Each contact entry in the database simply consists of the users userId, as well as the userId of the contact, as can be seen in figure 3.1. This is a simple, non-connected relationship meaning that I can add someone as a contact, without them having me as a contact. This resembles an address book, or a phone's contact list opposed to something like a Facebook friends list. This means a contact list can be constructed fairly easily by scanning the database table and retrieving all entries where the userId matches the user. From the other contacts userId, we can then lookup information about them from that Id.

userId	contactUserId
us-east-1:b2ed44ec-0b8d-43d0-8fef-a97182b4c4ca	us-east-1:a6ec0d84-d336-47f7-a373-eba...
us-east-1:b2ed44ec-0b8d-43d0-8fef-a97182b4c4ca	us-east-1:f6184c49-f89c-47f0-b7f9-4e238...
us-east-1:a6ec0d84-d336-47f7-a373-eba99bc405f2	us-east-1:b2ed44ec-0b8d-43d0-8fef-a971...

Figure 3.1 Contacts table in the database

Links

The heart of the application lays in the Links tab. This tab lists all of the Link sessions that you are a part of. The groups display name is set to either the other person's name if it is a two person session, or a comma separated list of people if the group is larger than three. You can see from figure 4.0 that I am in a 3 person group with people named Justin and Shannon, and in two one-on-one two person sessions with Karen, and Justin. Below each group display name is a textual description of when the last update to the Link session occurred. Link sessions are stored in order of most recent, with newly made sessions sticking to the top. If a group session is active without me participating, it will still jump to the top of my list when I finally open the app. When a session has recently been active, the name and update time will both become bold, and the icon on the right half side will change to indicate connectivity. Clicking on the list item will open the Link session once complete.

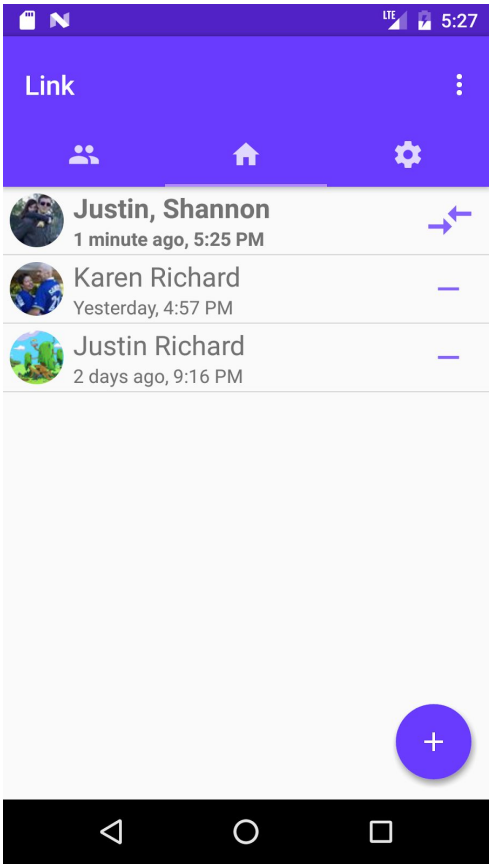

Figure 4.0 Links Tab

The database stores minimal information for each participant in the Link. The schema can be seen below in figure 4.1. This means that we are only storing the last known location of each person, so that means in the unfortunate case of a data breach that extensive knowledge of people's past locations and tendencies cannot be found out. Users will also be happy to know that only minimal amounts of their data is stored. The userId column links to the users userId, and the linkId is a unique ID generated for each new link session. The last update column is a unix timestamp of the last update, and the altitude, latitude, and longitude columns are for the users location of course.

userId	linkId	altitude	lastUpdate	lat	long
us-east-1:a6ec	154172fc-5d...	0	1481091939...	48.4647348	-123.4998546
us-east-1:a6ec	51e78e6e-06...	69	1480554453...	48.4663737	-123.5029832

Figure 4.1 Link Participants Database Table

Settings

Fairly self explanatory, the settings tab is where the user can customize their in-app experience to their preferences. There currently is not a super wide range of options as you can see from figure 5.0, but it also acts as a sort of placeholder for when more features come along that enable customization. Currently there is a section for personal preferences which only includes the ability to select a new username after yours has been set, as well as a section to control your data usage. The username preference also doubles as a way for users to see what their own username is incase they forget. The two data usage options will control how often information is sent back and forth between the server when the phone screen is on or off. I felt that users would enjoy being able to customize their data consumption and be less likely to uninstall the app blaming battery usage or something similar.

The users preferences are not stored in the database as they only pertain to that one user. The data is stored inside local application data. Selecting the username preference opens the username selection activity as seen in the previous username selection section.

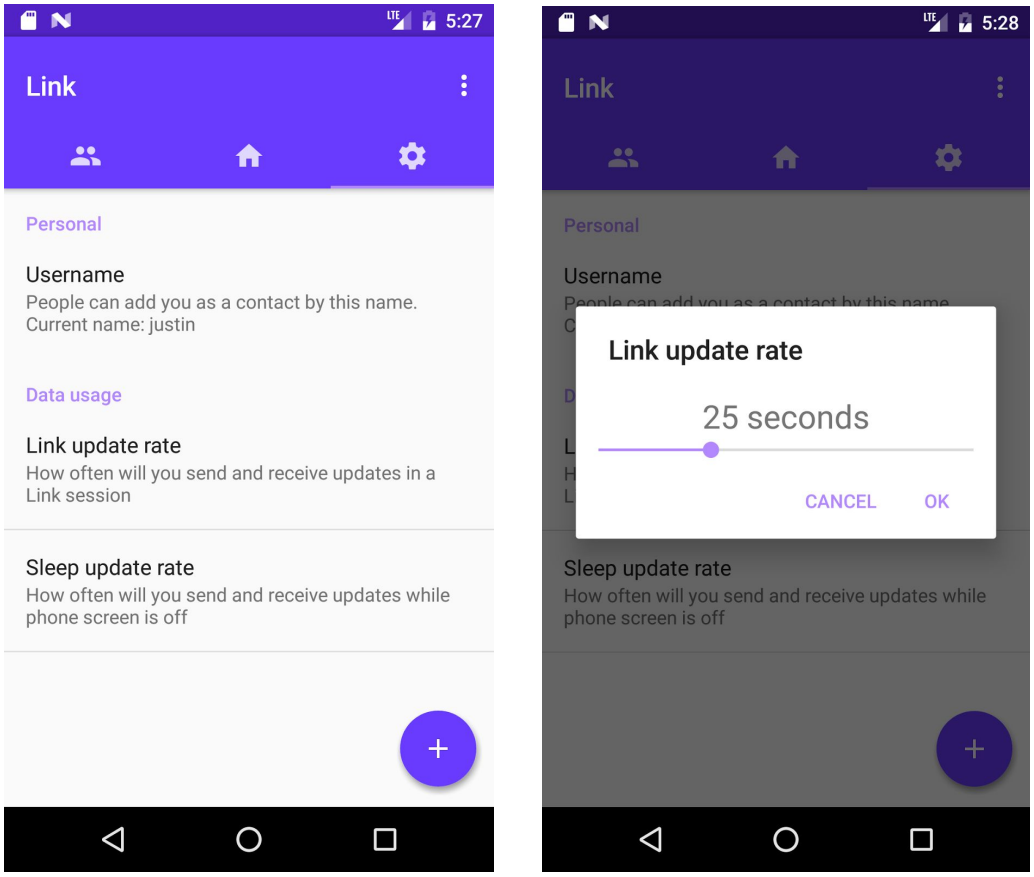

Figure 5.0 Settings Tab and Preference Change

Future Work

While a large portion of the work is complete, they are still key components that need to be completed. The following sections outline the major tasks remaining.

Active Link Sessions

Most of the completed work so far can be considered scaffolding when compared to this element. The active link sessions will be the heart of the application, where you can see real time updates of each other's locations. For each Link session you create you should be able to see each person's location overlaid on top of a Google Maps interface, with information on the distances between you and the other people. The groundwork is done to get this section going, such as adding contacts and signing in and initializing the Link, but it still needs to be made.

Notifications

The app is not very useful if you don’t know when to use it. Users need a way to be notified when the other person/people in the Link session are active and want to Link up. Notifications are not entirely simple, they require a lot of surrounding work in order to be efficient. You cannot simply poll the server waiting for updates as that would consume a ton of data, battery and CPU cycles for absolutely no reason. Google supplies ways to deliver push notifications immediately and efficiently, but it requires a fair bit of setup and an app server to forward requests. Currently we only have the app (clients) and a database which they interact with, no intermediate server sided processing layer which is needed.

Optimization

Optimization is a hard, but important task. Users will not want to use an app that is heavy on data, battery or processing power and want quick immediate responses to their actions. Ideally all appropriate data should be cached, and any new data requests should be completed with as little overhead as possible and be bundled together. Currently the application lacks caching of the contacts and Links lists. It was decided to avoid caching it for the moment since they are changed often, but ideally it should be implemented.

The user interface also needs its own form of optimization, as there are elements which are clunky and/or cause bugs. A smooth UI is important to any app. The app will also need to be tested on a wide range of devices on various firmware levels to ensure it responds similarly on each.

Work Plan

A rough estimate of the work plan can be seen below in figure 6.0. This is only a rough outline since work can easily be set back by a variety of reasons. Learning all of these news technologies and features can often add a lot of time, as well hard to solve bugs can eat up a lot of time when trying to move on. Work also tends decrease in productivity near when other courses have exams (See figure 6.1, can you guess when midterms were?).

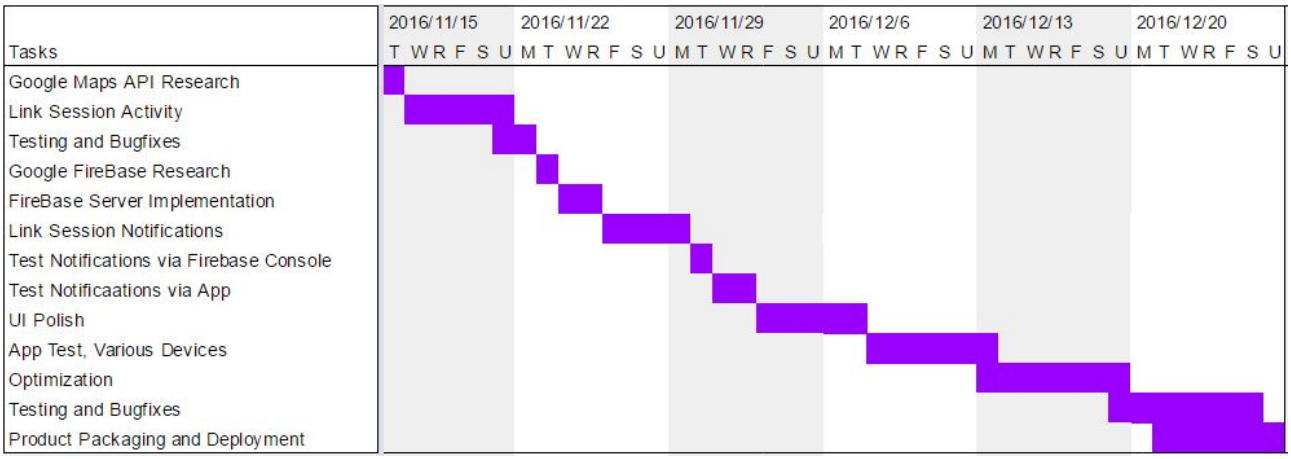

Figure 6.0 Work Plan Gantt Chart

Figure 6.1 Git Contribution Chart