LOKKAL
FINAL PROJECT DESIGN

Version 2.0
05/08/2018

VERSION HISTORY

The project will be controlled using GitHub™. In our design plans for the app, we broke our development into sub-groups based on key functionality aspects. For each sub-group, we split out the group into even more specific groups and tasks. Once a task or group had been completed and testing of the feature had been conducted, the code will be committed to the repository. Each of us will have our own development version of the application so that any changes, testing, etc. do not end up breaking the version that we can guarantee is working correctly. Major version changes are reported below as follows:

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	Josh Sloan	10/14/17	Josh Sloan Kolby Fenster	10/14/17	Initial Design Definition draft
1.5	Josh Sloan Kolby Fenster	4/22/17	Josh Sloan Kolby Fenster	4/22/17	First Official Working Version
1.9	Josh Sloan Kolby Fenster	4/27/17	Josh Sloan Kolby Fenster	4/27/17	Pre-Release Version

TABLE OF CONTENTS

Introduction

Purpose Of The Final Project Design Document

The Final Project Design document documents and contains the necessary information required to effectively understand our architecture and system design in order to give the development team guidance on the architecture of the system and how it was developed. The Final Project Design document has been finalized after the completion of the Development Phase of the project. Its intended audience is the project manager, project team, and development team. Some portions of this document such as the user interface (UI) may on occasion be shared with the client/user, and other stakeholders whose input/approval into the UI is needed.

General Overview and Design Guidelines

Assumptions / Constraints / Standards

There were several assumptions going into the design process of the application. One of our biggest assumptions is that a user has basic smartphone literacy. The users will need to know how to open apps, recognize standard symbols for settings, calling, messaging, etc. Additionally, for testing this application, a lack of phones prevented other users from testing the application outside of the developers. Lastly, our general architecture was structured off past and current experience in terms of designing applications and creating the database.

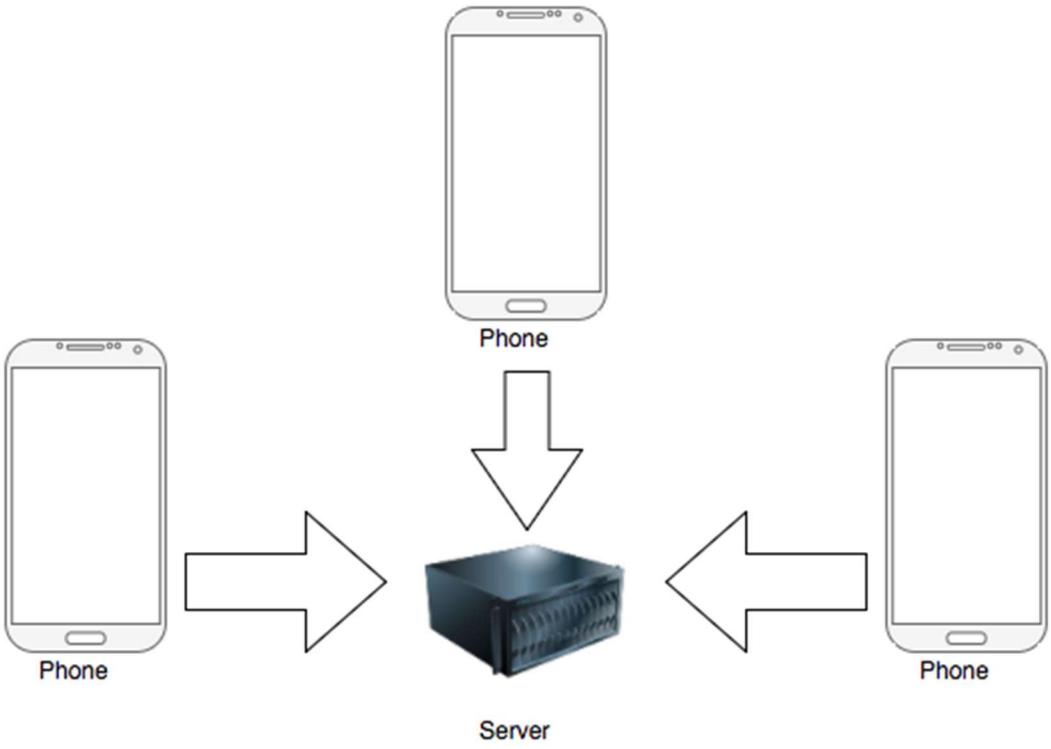
In regards to constraints, the mobile app is designed to only run on the Android operating system. Neither developer had an iPhone and due to lack of time, the the iOS application was unable to be developed. Time was a huge factor in regards to developing and building the functionality within the app. Along with this, testing of this application was semi-difficult due to the fact of hosting our server on campus and being required to be connected to the university's network. This meant testing our location outside of buildings was difficult due to the lack of connections. This application was not available on Google Play or the App Store so for testing each of our testers had to download the latest version of the application from our computers. When any changes were made, this process became quite tedious so we planned our "updates" accordingly.

For standards, we followed a standard design procedure in terms of using consistent colors, fonts, etc. We also wanted to use standard security practices for storing an individual's personal information in our database, but were unable to reach that far in development phase of the project.

Architecture Design

This section will define the various components and their interactions in context of the whole system.

Logical View



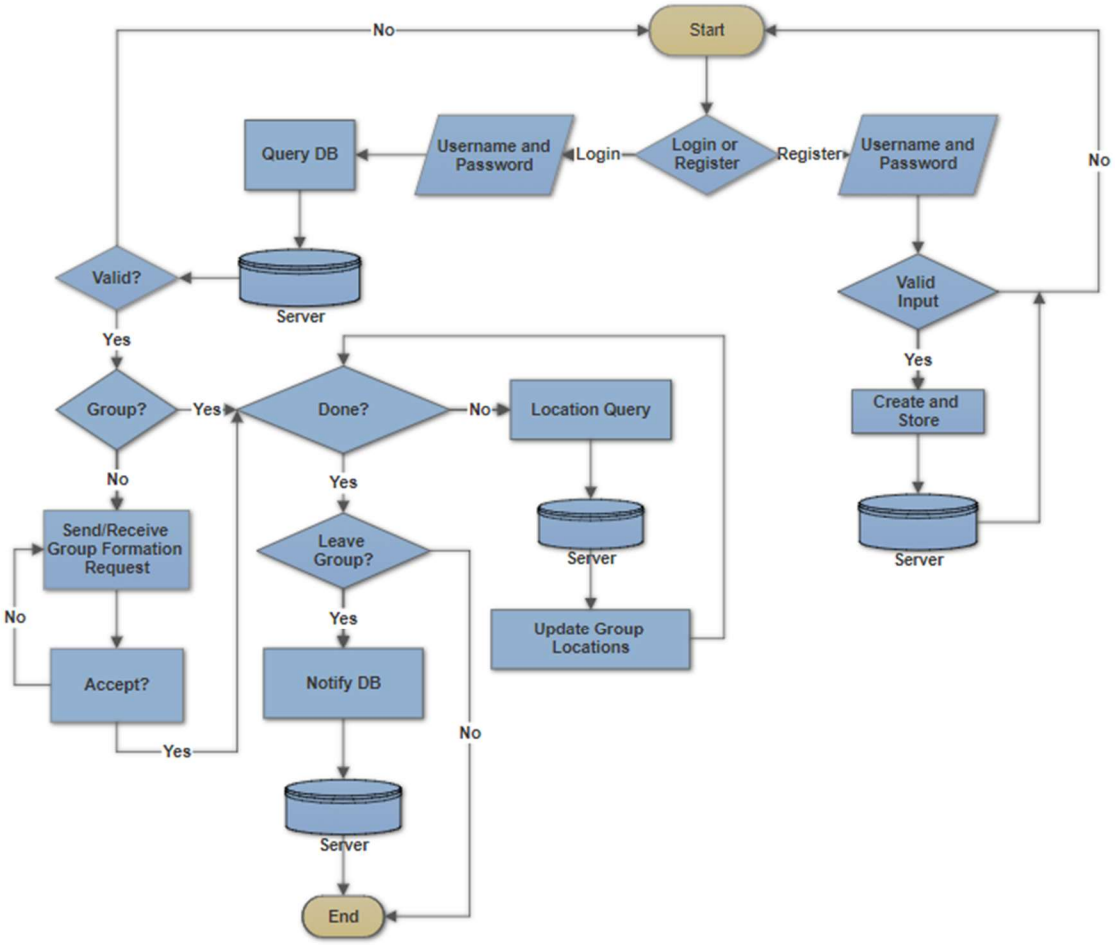
Hardware Architecture

This project will not provide any hardware to the client.

Software Architecture

Our app utilizes two activities, the LoginActivity and MapsActivity. We wanted to avoid using too many activities, as our app would become very computationally expensive to use if we were using activities for simple UI screen changes. For our purposes, we felt as though these were the only two areas which needed their own specific activity. For individual screen functionality, we implemented a variety of fragments which are responsible for a corresponding screen. Many of the fragments have an associated adapter which is used to populate a RecyclerView, which many of the fragment screens contain (corresponding with a specific database query).

The following flowchart on the next page defines the basic location tracking logic. It does not include the logical steps for special requests such as messages, nor does it show checking database information to notify the user of alerts (i.e. low battery) after receiving the location information from the server:



Security Architecture

Users must submit a valid user login which has previously been registered before gaining access to any content beyond that which is provided in the initial login screen. When registering this information with the database, measures were taken to ensure SQL injection could not occur. The database was queried to ensure that both the username and password were correct. From there the users were able to connect to each other via groups, which was handled by the back-end server. Location information was granted only to users who are identified as a group member of that specific user's group.

Communication Architecture

Location communication was handled through a back-end MySQL database, which clients queried on an interval set by the client. Each client uses sensor data to update their personal information and then notify the database of any changes, which all other users in a given group

can access through the shared MySQL database. The MySQL database only provides information regarding group member identification.

Performance

Performance was measured by the speed in which a user would send their information to the database, query and return the relevant group information, and update the map with this information. Through testing we noticed around a 10-15 second delay between the time a user would update their location and the update of the map would occur.

System Design

Use-Cases

To test our application, we came up with a wide number of use-cases to test the variety of features we implemented. Below are the some of the main use-cases that were essential to ensuring our app is working correctly.

- Case 1 - Opening the App and Signing In
    - 1a. Open up app on phone, enter a correct username and password to log in
        - User is taken to the next screen on the app where they can see the main menu of the app
    - 1b. Open up app on phone, enter an incorrect username and password
        - Error message should appear stating that the username/password is incorrect
- Case 2 - Sending, Accepting, Denying Friend Requests
    - 2a. Using the add friend feature, send a friend request to an individual
        - The individual the friend request was sent to should receive a notification and a message asking to accept the friend request
    - 2b. After receiving a friend request, accept the friend request
        - Both sender and receiver of the friend request will now appear on each other's friends list
    - 2c. After receiving a friend request, deny the friend request
        - The sender and receiver will not appear on each other's friends list
- Case 3 - Creating, Inviting, and Joining a Group
    - 3a. The user will access the create group screen and create a group
        - The user user will now be able to invite other people to the group
    - 3b. The user will create a group and attempt to invite another user.
        - The other user will receive a request to join a group
    - 3c. The user will join a created group
        - The user will now be able to see the locations of other members in the group

Database Design

These database tables are a general outline of the tables that we created for LokkaL.

Table Name:	PersonModule_Person					
Column Name:	PersonID	FirstName	LastName	DateOfBirth	Email	Password
Type:	int	varchar(50)	varchar(50)	date	varchar(50)	varchar(50)
Null:						
Relationship:	Primary Key					

This table stores information pertaining to an individual that creates an account with LokkaL. The PersonID column is a unique ID tied an individual. The FirstName column stores the first name of the individual and the LastName column stores the last name of the individual. The DateOfBirth column keeps track of an individual's age. Email and Password are tracked in this table as well. The email and password are stored in plain text as we were unable to implement any security features to encrypt this information.

Table Name:	PersonModule_Friendship					
Column Name:	FriendshipID	LeftPersonID	RightPersonID	StateDate	EndDate	ResponseTypeID
Type:	int	int	int	DateTime	DateTime	int
Allows Nulls:					X	
Relationship:	Primary Key	Foreign Key	Foreign Key			Foreign Key

This table tracks an individual's friends list. The FriendshipID column is a primary key that tracks each friendship and friend request. The LeftPersonID has a foreign key relationship with the PersonModule_Person table. This is the person who sent the friend request. The RightPersonID also has a foreign key relationship with the PersonModule_Person table. This is the person who received the friend request. The StartDate and EndDate columns are used to track when a friendship starts and ends and when a friend request was received. The ResponseTypeID column has a foreign key relationship with the ResponseType table and tracks whether a friend request is pending or was accepted/declined.

Table Name:	GroupModule_GroupSession				
Column Name:	GroupID	Group Name	GroupCreatorID	StartTime	EndTime
Type:	int	varchar(15)	int	DateTime	DateTime
Allows Nulls:					X

Relationship:	Primary Key		Foreign Key		

This table tracks created groups. The GroupID column is a primary key that tracks each group. The GroupName column is the name of the group that the user sets and will be limited to 15 characters. The GroupCreatorID has a foreign key relationship with the PersonModule_Person table and is the individual who created the group. The StartTime and EndTime columns track when a group starts and ends.

Table Name:	GroupModule_GroupMembers					
Column Name:	GroupMemberID	GroupID	PersonID	StartTime	EndTime	ReponseTypeID
Type:	int	int	int	DateTime	DateTime	int
Allows Nulls:					X	
Relationship:	Primary Key	Foreign Key	Foreign Key			Foreign Key

This table tracks the individuals who belong to a group. The GroupMemberID column is a primary key that tracks each group member. The GroupID column has a foreign key relationship with the GroupModule_GroupSession table. The PersonID column has a foreign key relationship with the PersonModule_Person table. The StartTime and EndTime columns track when an individual joins a group and leaves a group. The ResponseTypeID column has a foreign key relationship with the ResponseType table and tracks whether a group request is pending or was accepted/declined.

Table Name:	GroupModule_GroupMemberLocation					
Column Name:	GroupMemberLocationID	GroupMemberID	LocationTime	Latitude	Longitude	BatteryLife
Type:	int	int	DateTime	double	double	int
Allows Nulls:						
Relationship:	Primary Key	Foreign Key				

This table tracks a group member's location. The GroupMemberLocationID column is a primary key that tracks each location update for each group member. The GroupMemberID column has a foreign key relationship with the GroupModule_GroupMember table. The LocationTime column is the time the location update was received. The Latitude and Longitude columns store the

latitude and longitude of the individual when an update was received. The BatteryLife column represents a user's battery life at the time of the request.

Data Conversions

There will be no data conversions for this project.

Application Program Interfaces

As it currently stands, our application only relies on the following API (in conjunction with the Standard Android Development Kit):

Google Maps API (https://developers.google.com/maps/documentation/android-api/)

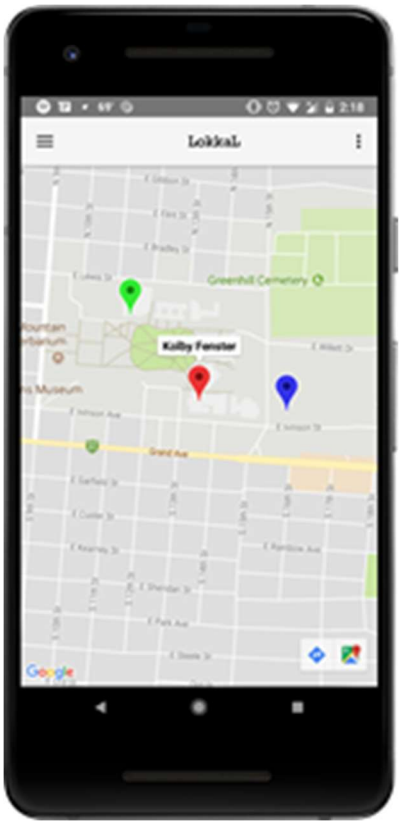
User Interface Design

These are the final user interface elements we designed for our application.

Login Activity Screen:

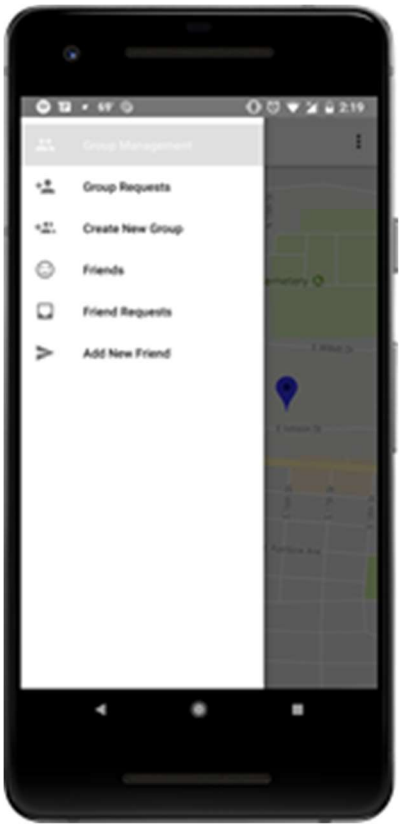
The login screen features two edit-text fields where users are required to enter in a valid user email address (which has been registered prior) and the associated password. They can then use the "Sign In" button to proceed. If the user does not have an account, they can use the "Register" button to open up an account registration dialog which will allow them to register a user account to be stored in our database.

Maps Activity Screen:



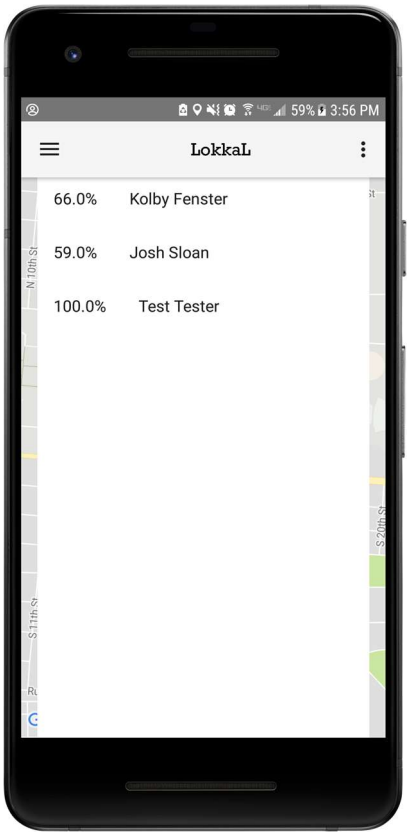
The Maps Activity screen shows group member location changes in real time (on an interval set by the individual user). Users are identified by their group-specific color, as well as the information which can be obtained by clicking on an individual marker.

Navigation Drawer Screen:



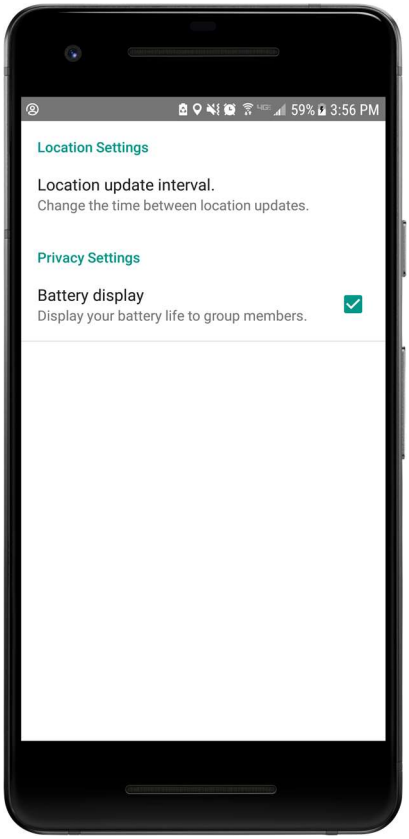
The Navigation Drawer screen allows users to have an easy way to have access to all the controls needed for complete group location sharing functionality. This entails the following: the Add New Friend screen, Friend Requests screen, Friends List screen, Create New Group screen, Group Requests screen and finally the Group Management screen. Most of these screens simply contain a RecyclerView which is populated with the associated list tied to the account, or edit-text fields to input values for Add New Friend or Create New Group functionality (see Group Management screen below).

Group Management Screen:



The Group Management screen enables users to easily review their group members, as well as their associated battery life (depending on their respective permissions). In the future, we intend to build off of this by adding more information, such as a user's personal profile picture, as well as a button to send a direct message to a specific group member, or broadcast a message to the entire group.

User Settings Screen:



The settings screen allows the user to customize their LokkaL experience by specifying advanced permission sets not default for the location tracking functionality, as well as to change any personal preferences the user might have when using LokkaL. These settings are able to be accessed at any time after the user has logged in through the toolbar dropdown in the top right corner of the Maps Activity screen.

Section 508 Compliance

LokkaL agrees to follow and comply with the recommended guidelines and standards proposed by Section 508 of the Rehabilitation Act of 1973.

Final Project Design Approval

The undersigned acknowledge they have reviewed the LokkaL Final Project Design document and agree with the approach it presents. Any changes to this Requirements Definition will be coordinated with and approved by the undersigned or their designated representatives.

Signature: Josh Sloan _____ Date: _____
Josh Sloan
Lead Developer I of LokkaL

Signature: Kolby Fenster _____ Date: _____
Kolby Fenster
Lead Developer II of LokkaL