The Report Committee for Salvatore Gregory Scaffidi III Certifies that this is the approved version of the following report:

The Smartphone as a Data Collection Device

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor:

Joydeep Ghosh

Christine Julien

The Smartphone as a Data Collection Device

by

Salvatore Gregory Scaffidi III, B.A.

Report

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

August 2013

Dedication

To my loving wife and parents, for their unwavering support and encouragement.

Acknowledgements

I would like to thank my supervisor, Joydeep Ghosh, and my reader, Christine Julien, for their guidance and advice. I would also like to thank the friends and family who provided valuable feedback that guided the development of the iSeeMe application.

Abstract

The Smartphone as a Data Collection Device

Salvatore Gregory Scaffidi III, M.S.E.

The University of Texas at Austin, 2013

Supervisor: Joydeep Ghosh

The introduction of mobile devices to the pockets and handbags of people living all over the world has made the practice of mobile computing nearly ubiquitous in modern society. iSeeMe is an Android application that empowers the user through the revelation of the vast amount of private data that mobile devices are capable of silently capturing in the background. iSeeMe strives to provide the user with a means to correlate this passively-collected information with data of personal importance to the user. This report looks into the development and implementation of the iSeeMe solution. It discusses design decisions, describes the iSeeMe architecture, and outlines the process of engineering the application. It also examines the role of personal data in modern society and explores the mobile application market to see where iSeeMe will fit in among similar applications. Finally, it analyzes the results of the development effort and identifies areas for future enhancement.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Background

The introduction of cell phones to the pockets and handbags of people living all over the world has made the practice of mobile computing nearly ubiquitous in modern society. In the United States of America, approximately 87% of adults own a cell phone and 45% of adults own a smartphone. Cell phone owners use their phones for much more than just placing phone calls. Some other common activities include: texting, social networking, taking photos and videos, browsing web pages, using email, banking, playing games, watching videos, listening to music, getting news and weather information, and using GPS navigation [1]. All of these activities and services are accessible through a single device that is available anywhere, at any time, providing a unique opportunity to collect information about a person's interests and habits.

Using a single device for such a variety of purposes is not a new phenomenon—a personal computer could facilitate many of the tasks mentioned above. However, two qualities of cell phones, which are not shared with PCs, make them especially interesting as sources of personal data. First, cell phones are mobile. They are not merely portable (designed to be easily taken to, and used from, different locations); they are meant to be carried all the time and used in almost any situation. Second, cell phones are always-on devices; they must remain powered if they are to receive phone calls. Consequently, cell phones are capable of capturing a nearly constant stream of data about their users.

1.2 Problem

1.2.1 Data Overload

Today, with the Internet playing such a large role in the daily lives of so many people, collecting, analyzing and monetizing the personal data of individuals has become

big business. Organizations have invested enormous amounts of time, money and expertise in developing systems to predict the likelihood of a person clicking on an advertisement based on previously collected data describing that person's activities online [2]. This thirst for data has led to the creation of many jobs with the sole purpose of sorting through and making sense of user data. The data scientist—a "high-ranking professional with the training and curiosity to make discoveries in the world of big data," has been named the, "sexiest job of the 21st Century [3]." Manipulating and analyzing this data are incredibly important tasks to firms that generate and track enormous amounts of data. Companies like Google and Facebook derive significant profits by mining this data to formulate strategies for displaying targeted advertisements to users [2]. While this kind of personal data is clearly important from a business/marketing perspective, its value to the individuals generating the data has been largely unexplored.

1.2.2 Data Privacy and Monetization

Data is used internally for marketing, advertising, and product personalization purposes. However, companies also recognize the monetary value of this data and exploit it as a commodity. Verizon has a new product called Precision Market Insights which, when a user navigates to a webpage on their cell phone, accumulates information about the website, and the user's location. This data is then sold to other businesses [4].

Facebook, the world's most popular social network, collects massive amounts of data about the user. Facebook not only collects data from its own site but can also latch on to other sites that its users visit to gather additional information about their habits and preferences [5]. The Facebook mobile app provides yet another avenue for Facebook to source data about its users. The latest incarnation of Facebook's mobile offering, "Facebook Home" replaces the default home screen and application launcher of an

Android device. As a home screen and application launcher, Facebook Home inherently knows which applications are used, when they are used, and where the device is located at any point in time [6].

In addition to Verizon and Facebook, many other applications collect user data and send it to advertisers for profit. "Armed with this information firms ...track the individuals' movements and sell personalized adverts for which they can make more money than regular ones [7]." Most of the behind-the-scenes data collected by the applications run on mobile devices is not shared with the user. Instead, the primary use of this data is to benefit corporations seeking to market their products and services

1.2.3 Data for the User

Among the many applications that deal with user data, some apps are geared towards using personal data to help benefit the person that is generating it. However, most of these apps focus on tracking a single kind of data. There are applications designed solely for the purpose of tracking and analyzing sleep patterns and REM cycles (Sleep Cycle[1]). Some applications track workouts and calories burned (Fitocracy[2]). Others track meals and calories consumed (Meal Snap[3]). Still others are designed to track your moods (Mood Scope[4]) or stress levels (Stress Check[5]). Some of these applications may incorporate one or more of these ideas, like tracking meals and workouts, or workouts and sleep [8]. Table 1, shown below, provides a sample set of Apps that help users track certain aspects of their daily lives.

[1] http://www.sleepcycle.com/
[2] https://www.fitocracy.com/
[3] http://mealsnap.com/
[4] https://www.moodscope.com/
[5] https://play.google.com/store/apps/details?id=com.azumio.android.stresscheck&hl=en

App Name	Item Tracked
Calorie & Nutrition Manager[6]	Calorie Consumption
Fitbit[7]	Sleep Patterns & Physical Activity
Foursquare[8]	Location
Gymrat[9]	Workout
iBP[10]	Blood Pressure
Meal Snap	Calorie Consumption
Mood Scope	Mood
Moodlytics[11]	Mood
My Fitness Calculator[12]	Calorie & Water Consumption
Runkeeper[13]	Workout
Sleep Cycle	Sleep Patterns
Stress Check	Stress Levels
Waterlogged[14]	Water Consumption

Table 1: List of Similar Applications

The problem with such specialized apps is that they do not consider how each of these individual areas may relate to one another. For example, a person may eat more, exercise less, and sleep less soundly when under more stress. This important correlation would be missed if an application tracked only exercise and meal data but not stress. Tracking one or a few aspects of a person's life independently may allow the user to visualize the measurements they have recorded over a period of time, but considering more factors over the same period of time could reveal more interesting (and personally meaningful) relationships.

6 https://play.google.com/store/apps/details?id=com.nutritionfoods.NutritionCalculator&hl=en
7 http://www.fitbit.com/
8 https://foursquare.com/
9 https://play.google.com/store/apps/details?id=com.gymrat&hl=en
10 http://leadingedgeapps.com/iBP.html
11 http://www.moodlytics.com/
12 https://play.google.com/store/apps/details?id=com.finessCalculator
13 http://runkeeper.com/
14 https://itunes.apple.com/us/app/waterlogged-drink-more-water/id352199775?mt=8

1.3 SOLUTION

The aforementioned problems provide a unique niche, not yet capitalized on in the marketplace. iSeeMe is a mobile application that empowers its users by granting access to the wealth of information that their mobile devices are capable of capturing. In addition to passively collecting data, the user can personalize the app by entering any number of things that other apps can track (stress level, mood, weight, etc.). This application not only records the user's manual inputs, but also records when other applications are used, how long they are used, and the location where they are used. iSeeMe performs two main functions:

1. Make this information available to the user to find meaningful relationships in the data
2. Preserve the data for future analysis.

The primary function of this application is to allow users to correlate regularly recorded measurements or ratings with the data that the phone passively collects. For example, users might wish to track mood or energy level to see what things they do on days where they feel happier or have more energy (or conversely, what they do on days where they are not as happy or have less energy). Maybe users learn that they feel more depressed on days that they spend a lot of time using social media applications on their phones or more energetic on days they play certain games. By preserving the data for future analysis, users are also able to identify trends over time for any of the items tracked.

iSeeMe gives the user access to much of the data that phone companies sell for profits, social networking sites use for advertising, and in general data that is used to benefit others. It allows the user to benefit from the advances in technology that allow a

phone to record virtually everything the user does. It gives users the tools to better understand their behavior and better their lives if they so choose.

Three central tenets drove the development of the iSeeMe application.

1. Flexibility – To provide the most benefit to the user, the app will not constrain the type of data the user wishes to track.
2. Extensibility – The application will be designed such that, as new methods of capturing data become available, they will be able to be incorporated with minimal effort and disruption to the user's existing data.
3. Transparency – All data about the user captured by this application will be made available to the user, and the application gives the user complete control over where and how the data is used and stored.

1.4 Overview

The remainder of this report describes the process of developing iSeeMe, a proof-of-concept Android application. It will first discuss the design process. It critiques and explains the many decisions that were made in the design of the database, user interface, Android service and web server components. The report will then look at the results; a number of graphs are developed to give users insight into the kinds of data that iSeeMe records. The next section of the report will discuss the challenges faced during development and how each was addressed. The report will then conclude with an evaluation of the overall project and potential for future improvements and enhancements.

Chapter 2: Design

The iSeeMe architecture specifies four primary components: a user interface component, a data management component, an Android service, and a web server. Together, these components allow the user to manually enter data, record data about the user's activities in the background, and provide the user with access to that data.

Figure 1, shown below, depicts the internal structure of each of the four conceptual components:

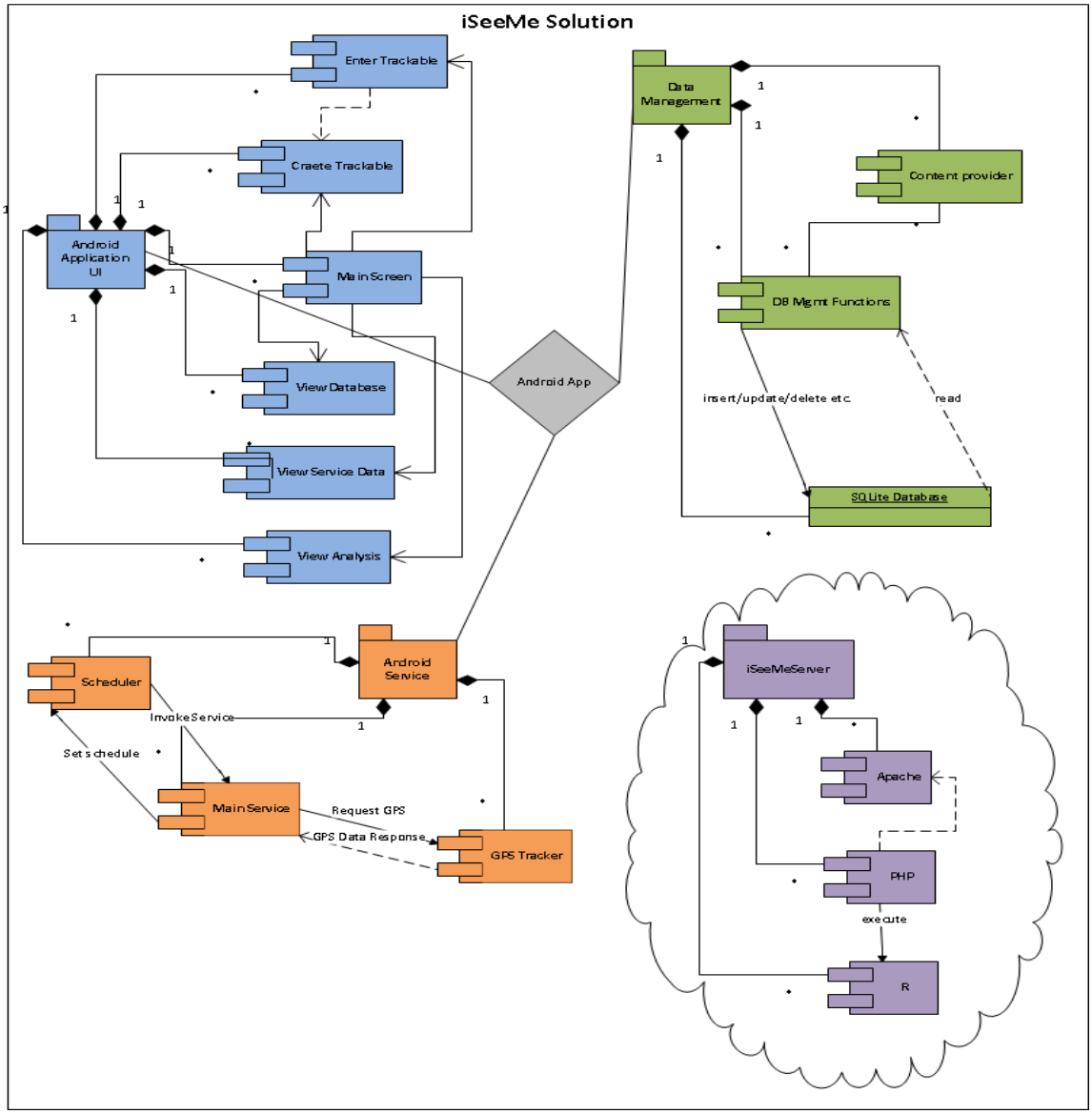

Figure 1: iSeeMe Architecture

The next figure highlights the inter-component communication:

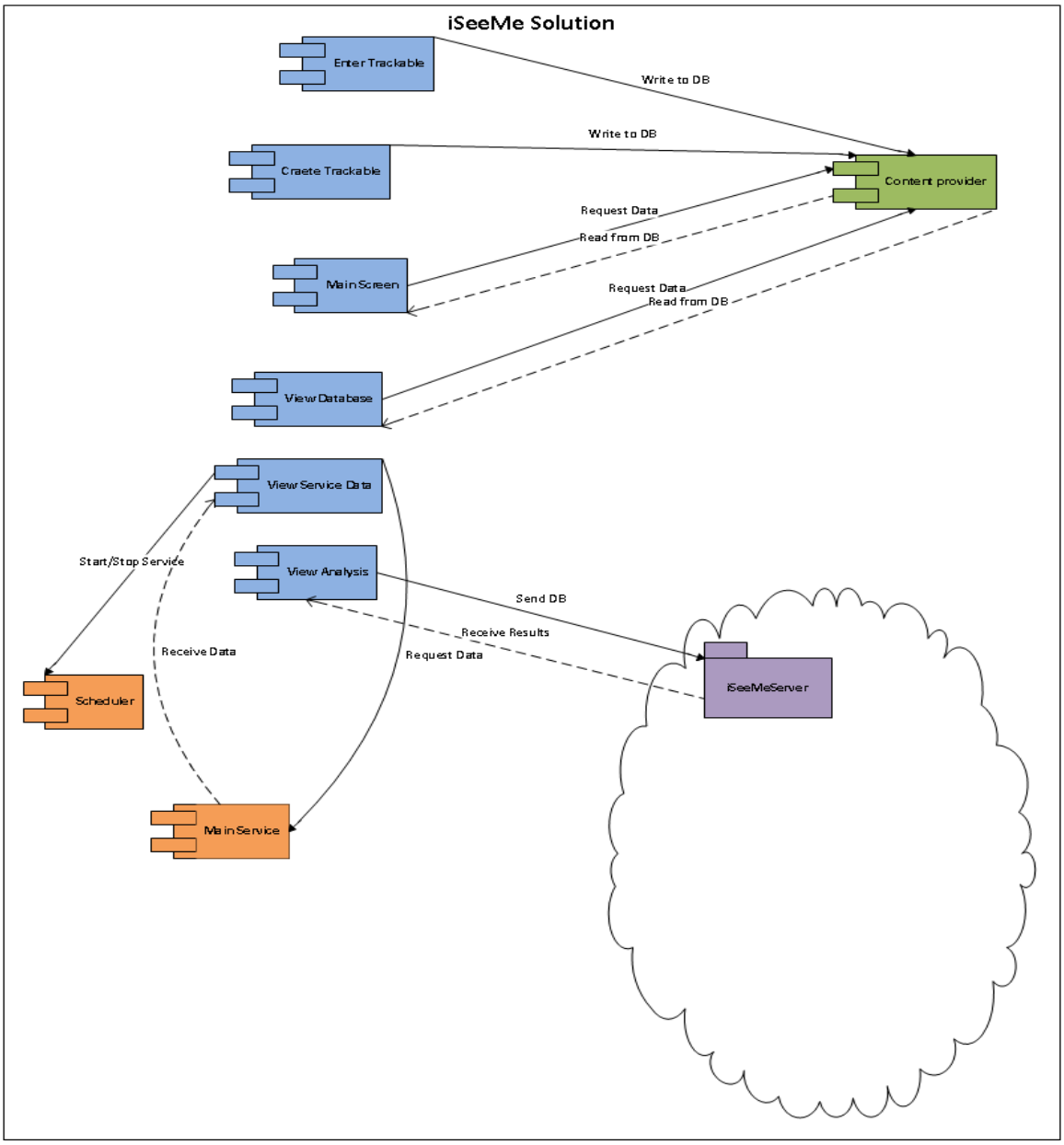

Figure 2: iSeeMe Inter-Component Connections

2.1 DATABASE AND DATA FORMAT

2.1.1 SQLite

The Android operating system supports the SQLite database out-of-the-box, and the iSeeMe application makes use of this support to store all of the user's data in a single SQLite database. To reduce complexity in the Android application, all the data is stored using just one table in the database. A SQLite table can be described as a set of columns and rows. Each row has an ID, which can be automatically incremented as data is inserted into the table. Each table must at least contain one column under which this ID number is recorded. Other columns need to be defined to record different kinds of information in the table. A table designed to track a person's weight over time might contain the following columns: _ID, Weight, Unit, Date, Time.

The first 2 columns capture individual measurements of one type of data, weight. Additional columns are added to record the date and time data. Now, suppose a user wishes to track something else in addition to weight. Tracking the additional feature would require either:

1. Adding another column to the table
2. Creating an additional table
3. Otherwise altering the format of the existing table.

Because a primary goal of this application is to allow the user flexibility in determining which specific things to track and the number of those things, the first option does not scale well. Adding a new column to an existing table in SQLite is possible, but removing or renaming a column is not [9]. Thus, if a user created a category for something to track and later decided that it was no longer important, the database would be left with an unused column. One alternative would be to make a new version of the table to replace the old version. In this case, preserving the contents of the original

database would require, at a minimum, reading all the original records into memory, creating the new database, and writing them back to the new database. Database operations are relatively expensive because they require access to the slower non-volatile memory; the application performance could suffer each time the user decided to track something new. Additionally, more complex code would be required to deal with an unknown number of columns. Another alternative would be to create a number of additional dummy columns that could be used to track additional items in the future. However, this solution would really just be working around a fundamental limitation of the database design. It imposes a limit on the number of features that could be tracked and demands additional resources to process empty columns and cells.

A database design that instead follows the second option, using additional tables for tracking additional features, avoids the problems associated with upgrading the database version. Such a design is common among SQL databases. Data from multiple tables can be combined using SQL queries. However, additional complexity is required to provide a means for linking the separate tables. In SQL, primary keys and foreign keys are used to perform this function. An application that used this solution would have to be capable of generating the necessary SQL statements to join a variable number of tables. Such statements could quickly increase in complexity with the number of additional tables. The application code to generate those statements and process their results would also add a non-trivial amount of complexity to the design. This solution also presents additional challenges when considering another goal for this application, presenting the data to the user in an accessible form.

The drawbacks associated with the first two options led to a more creative design for a single-table database with a fixed number of columns capable of tracking an undetermined number of features. The solution borrows some ideas from each of the

other database designs. While the table contains a fixed number of columns, not all of them are used. A few extra columns were added to allow specific features to be tracked in the future, when and if support is added in the application to gather the data. The design exploits the fact that while changing the columns of a SQLite table is relatively cumbersome, creating and deleting rows is extremely easy. The solution employed is to specify a column to track the generic feature 'name' instead of a specific feature (such as 'weight'), and to add two additional columns, one to record the data value and one to record the type of data being captured. With these changes, the table discussed above now contains the following columns: _ID, InputType, Name, Data, Unit, Date, Time. A table using this format can conceptually track an infinite number of features, recording a data point, date and time for each.

2.1.2 iSeeMe Database Solution

The iSeeMe database solution puts into practice the concept of 'semi-structured data'. In his paper titled, "Semistructured Data", Peter Buneman writes, "In semistructured data, the information that is normally associated with a schema is contained within the data, which is sometimes called 'self-describing' [10]." Buneman discusses three common reasons for using semi-structured data, two of which are reasons why iSeeMe uses this type of data format:

1. "…to have an extremely flexible format for data exchange…"
2. "…to be able to query data without full knowledge of the schema."

iSeeMe needs to send the data that it collects to a separate server for analysis. The flexibility of a semi-structured data format promotes a database design that appeals to the software on both the Android device, as well as the server. According to Buneman, the second reason is important when considering how the user will be able to browse the

contents of the database. Having all the data in one table allows iSeeMe to display all of the data that iSeeMe collects, exactly how it is being stored on the device. A simple example of a table that follows the same design principles used by iSeeMe is shown below:

_ID	inputtype	name	data	unit	Date	Time
1	number	Weight		lbs		
2		Weight	165		7/1/2013	8:00
3		Weight	166		7/2/2013	8:15
4		Weight	168		7/3/2013	19:00
5	number	pulse		bpm		
6		pulse	65		7/3/2013	19:02
7		iSeeMe			7/4/2013	12:00

Table 2: iSeeMe Database Example

If the above table was used by an application to allow a user to enter data, it would likely contain a form for the user to type the value of the 'name' in one field and enter the data in another. If such an application was to provide a more user-friendly interface that allowed the user to select the 'name' from a list of previously-entered names, it would need to query the database for the distinct name values and provide them as options to the user. This need is easily satisfied with 'SELECT DISTINCT' in SQL. This solution would work for an application that is designed for only tracking manual inputs. However, the iSeeMe application needs to track both manual inputs as well as input that is generated in the background without the intervention of the user. The iSeeMe database also records the name of the applications that the user uses. These application names are also stored in the 'name' column, and would show up in the list as one of the user-defined input types.

The ‘inputtype’ column is used to indicate whether the value in the ‘name’ column reflects a user-defined category label for manual input or a name automatically generated by the iSeeMe service. The only time that a value is ever written under this column is when a category is first being defined by the user. When categories are first defined, no value is recorded under the ‘data’ column, providing a simple method for distinguishing category labels from actual instances of the user recording measurements under those categories.

The actual table used by iSeeMe includes several other columns to capture different kinds of data. These columns are listed and described in the table below:

Column Name	Column Description
id	unique number identifying the row. automatically increments as new rows are added
inputtype	specifies the kind of data being recorded and how the value in the 'name' column should be interpreted
name	a category supplied by the user. a human readable name associated with the data being recorded or the name of the application used.
description	a user-supplied description of the data being tracked
data	a value entered by the user
unit	the unit of measurement for data entered under a certain category
datestart	the starting date for an activity the user wishes to track or the date the user started using an application on the phone.
datestop	the stop date for an activity the user wishes to track or the date the user stopped using an application on the phone.
timestart	the starting time for an activity the user wishes to track or the time the user started using an application on the phone.
timestop	the stop time for an activity the user wished to track or the time the user stopped using an application on the phone.
loclat	latitude of the device as recorded by the iSeeMe Service
loclong	longitude of the device as recorded by the iSeeMe Service
locname	address associated with a given latitude and longitude
locnetwork	SSID of the WiFi network the device is connected to
rssi	signal strength of the WiFi network that the device is connected to
temperature	for future use. intended to record current outdoor temperature
weather	for future use. intended to record current outdoor weather conditions (sunny, raining, cloudy, etc.)
sensortemperature	for future use. measurement from an onboard temperature sensor.
sensorpressure	for future use. measurement from an onboard barometric pressure sensor.
sensorhumid	for future use. measurement from an onboard humidity sensor.
locationdiff	difference between the location recorded in this row and the last known location

Table 3: Database Columns and Descriptions

The 'inputtype' column also specifies the type of data being recorded. Currently, iSeeMe only records numerical data, but additional values in the 'inputtype' column

could specify that the data for a category should be treated as a different kind of input. For example, the category label, "happiness" might be better understood as a rating input rather than just a numerical input. This distinction could be used to provide a different user interface for inputting data, more suitable to the kind of data being recorded. In fact, Android provides many different options for users to enter data. Some that are already used by iSeeMe include: the standard soft keyboard[15] combined with a basic text field[16], date and time pickers[17], a check box[18], a button[19], a context menu[20], an options menu[21], and a list containing clickable items[22].

The Android "spinner[23]" and "rating bar[24]" are two currently unused options that would potentially work very well for users to manually input data that may have contextual meaning beyond what can be effectively conveyed using raw numbers. The spinner works like a drop-down menu and contains a set of options for the user to choose from. The rating bar presents the user with a horizontal sequence of empty star images that the user can interact with to select a rating by "filling" the stars up to a certain point in the sequence with color. In the case of "happiness" the Android rating bar might be a good choice. A spinner with a list of values like "grumpy", "cheerful", "sad", "happy", "content", "confused" etc. might be appropriate for a user that wishes to track "mood". iSeeMe is not currently capable of accepting input in this manner— the current

[15] http://developer.android.com/training/keyboard-input/style.html
[16] http://developer.android.com/guide/topics/ui/controls/text.html
[17] http://developer.android.com/guide/topics/ui/controls/pickers.html
[18] http://developer.android.com/guide/topics/ui/controls/checkbox.html
[19] http://developer.android.com/guide/topics/ui/controls/button.html
[20] http://developer.android.com/guide/topics/ui/menus.html#context-menu
[21] http://developer.android.com/guide/topics/ui/menus.html#options-menu
[22] http://developer.android.com/guide/topics/ui/layout/listview.html
[23] http://developer.android.com/guide/topics/ui/controls/spinner.html
[24] http://developer.android.com/reference/android/widget/RatingBar.html

implementation should be viewed as a stepping stone to achieving this level of sophistication.

2.2 SERVICE

The iSeeMe service runs in the background and performs a set of operations at a specific repeating interval. To ensure that the service runs every time the phone boots, a customized BroadcastReceiver is registered with the "android.intent.action.BOOT_COMPLETED" intent filter to receive notification that the operating system has finished booting. This broadcast receiver sets up a repeating calendar event which occurs every few seconds. Another broadcast receiver receives this calendar event and has the sole function of triggering execution of the android service. In the course of developing this application different time periods were tested, from one to thirty seconds, with the goal of balancing the accuracy of the service in capturing all application usage against the effects that running a service at such a high frequency has on battery life. The results of this testing will be discussed in more detail in the challenges section of this report.

2.2.1 Application Monitoring

Each time the Android service is triggered, its onStart() method is called. From this method, a list of all recently running Android activities is requested from the operating system. Activities in Android are initiated by means of registering 'intents' with operating system. The iSeeMe service hashes the string representation of the base intent for each recent task in an array, currRecentAppHash[]. Before the service exits, a copy of that array is stored in another array, lastRecentAppHash[]. An example of a base intent expressed as a string is, "Intent{act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER]flg=0x10000000cmp=com.owlmonkeyapps.is

eeme/.ISeeMeOverviewActivity},” which represents the base intent for the iSeeMe application. The string representing the most recent activity’s base intent is stored in the string variable, “recentApp1”.

The recent tasks are provided by Android in an ordered list, with the most recent listed first and the least recent listed last. Since the list is ordered, currRecentAppHash[] and lastRecentAppHash[] are also ordered. These two arrays are compared each time the service runs. If a difference is found, the iSeeMe service determines that the user must have either started using or stopped using one of the applications corresponding to an intent in the list.

An important marker for determining whether or not the difference resulting from the comparison of the *RecentAppHash[] arrays indicates that the user stopped using an activity, as opposed to started using an activity, is the string corresponding to the base intent of the Android home screen. If the home screen activity was previously the most recent in the list, but is no longer the most recent, the user must have started using the most recent activity. Likewise, when the Android home screen itself appears as the most recent activity, the user must have stopped using whatever activity was previously the most recent in the list. Fortunately, the Android home screen is easily identifiable. It’s base intent appears as something like the following, “Intent{act=android.intent.action.MAINcat=[android.intent.category.HOME]flg=0x10600000cmp=com.android.launcher/com.android.launcher2.Launcher}.” The key to identifying whether an activity is a home screen is the substring, “cat=[android.intent.category.HOME],” which the iSeeMe service looks for when it parses the base intent string of the most recent activity. When the iSeeMe service determines that an activity has been started, that activity’s base intent string (recentApp1) is resolved to an application name, which is then written to the SQLite database. The

value stored in “recentApp1” is then copied to another static string variable, “recentApp2”. If a string is stored in recentApp2 when the home screen becomes the most recent activity in the list after a change is detected in the *RecentAppHash[] arrays, the name of the application that owns that activity is recorded in the database.

The algorithm described above works well when the user is actively using the phone and opening and closing applications from the home screen. A minor modification is required to support the case where the user switches applications without first going back to the home screen. An example illustrating this situation would be when the user is currently using some application and receives a notification that a new email or text message has arrived. The user can open the email application or text message composer by clicking on the notification. In this case, the iSeeMe service determines that if *RecentAppHash[] arrays are different, and recentApp1 and recentApp2 are different, and neither of them contain the substring, “cat=[android.intent.category.HOME]” that the user must have stopped using “recentApp2” and started using “recentApp1”.

As described above, the iSeeMe service’s application usage detection capabilities are almost complete, but the full solution requires the service to handle one additional case where the user may stop using an application without returning to the home screen or starting another application. In this case, a user stops using the application simply by turning off the phone’s screen (or leaves the device idle long enough for the screen to turn off on its own). To handle this case, the iSeeMe service itself must implement its own broadcast receivers to request notifications from the operating system when the user turns off the screen. When that notification arrives, the service determines that any activity represented by recentApp1 (that is not the home screen) is no longer being used. The database is updated with the change and the *RecentAppHash[] arrays and recentApp* strings are cleared.

Normally, when a user turns the phone's screen off while an application is still in the foreground view, that application will again become visible in the foreground the next time the screen is turned on and the device is unlocked. To completely handle the case where the user does not return to the home screen before the screen turns off, the following three intent filters are needed by the iSeeMe service and are registered with its broadcast receivers:

- android.intent.action.SCREEN_OFF
- android.intent.action.SCREEN_ON
- android.Intent.ACTION_USER_PRESENT

2.2.2 Location Monitoring

In addition to monitoring application usage, the iSeeMe service also attempts to record significant changes in the user's location. Every five minutes, the iSeeMe service requests the most recent location known by the Android operating system. This location can be represented as latitude and longitude coordinates. Similar to how the iSeeMe service retains the previous most recently used application in recentApp2, the last-known location is also recorded in a static variable, "lastLoc." Each time the iSeeMe service requests a location update from the operating system, it will compare the result to the previous last-known location. If the distance between the two locations is greater than fifteen meters, the value in "lastLoc" is updated with the newly acquired location and a new entry is made in the database reflecting the change in location. When location updates are made to the database, the iSeeMe service attempts to resolve the location to an address, which is also written to the database.

In addition to location information, the iSeeMe service also updates the database with the SSID of the currently-connected-to WiFi network (if any) and the WiFi network

signal strength. The SSID is included because it may be useful for identifying locations that are more important to the user. A user may be more likely to automatically connect to WiFi networks in places they frequently visit. The signal strength is recorded because it may be useful if future enhancements to the application attempt to measure how much an individual moves around while they are present at a particular location.

2.3 iSeeMe Server

The task of analyzing data is taken up by the iSeeMe server, which utilizes the Apache HTTP server and PHP to handle communications with the iSeeMe Android application. The purpose of the iSeeMe server is to transform the data captured by the iSeeMe application into forms that are meaningful to the user and simple to understand. Simplicity and meaning are easily expressed by using charts and graphs that visually present features in the data. R provides a means of preparing this data, modeling it, and generating a wide variety of charts and graphs. Unfortunately, R is not available as an easily distributable Android application, nor is there an obvious alternative to R (on Android) that provides the same set of features, flexibility and community support.

For iSeeMe to use R, the user's data (in the form of a .CSV file) is sent from the Android application to a PHP page hosted on a networked server. This .CSV file is then written to the file system. After the data has been successfully uploaded, the name of the .CSV file is passed as an argument to R, which runs a .r script residing on the server. The .r script uses the filename and assembles a customized .Rmd (R markdown[25]) script with R code to import the .CSV file and perform some analysis functions on it. After the .Rmd file has been created, the knit command from the 'knitr' package[26] is called to convert the .Rmd file to a .md file. This .md file is used with the markdownToHTML

[25] http://cran.r-project.org/web/packages/markdown/index.html
[26] http://cran.r-project.org/web/packages/knitr/index.html

command from the 'markdown' package to generate an HTML report containing the results of the various analysis performed, including graphical representations of those results. After the HTML file is created, the PHP page streams the file back to the device where it is displayed to the user.

2.4 Application User Interface

The user interacts with the iSeeMe application via a set of graphical user interface screens, each with a specific purpose. The main interface, which is displayed when the user first opens the application, provides direct links to all of the other individual user interface screens through several menu options. As an Android application, all the user interface components are implemented by Android 'activities'. "An activity is a single, focused thing that the user can do. Almost all activities interact with the user… [11]."

2.4.1 Main Screen and Data Input

The main activity extends the Android ListActivity class and provides the user with a list of all the things that they have chosen to manually track. Initially the list is empty. To add a new item to this list, the user starts another activity and enters the name of something to track, as well a unit for quantifying it, and optionally a description. This data is written as a new entry in the database. After the user creates a new item to track, it will appear in the list in the main activity. When the user selects any item in this list, a new activity will be displayed that allows the user to record a new measurement for that item. The user can select a start date, start time, end date, and end time for each measurement recorded, allowing the user to capture data about any period of time that may be relevant to the data; for example, a user might wish to record the start time and end time when a particular exercise was performed along with the number of repetitions completed or the distance covered. In the case where a time period is not relevant to the

data recorded, the values for starting and stopping can be set to the same date and time. Each time a new measurement is recorded, a new entry is added to the database containing the measurement and temporal data.

The following figures are screenshots of the various steps discussed above:

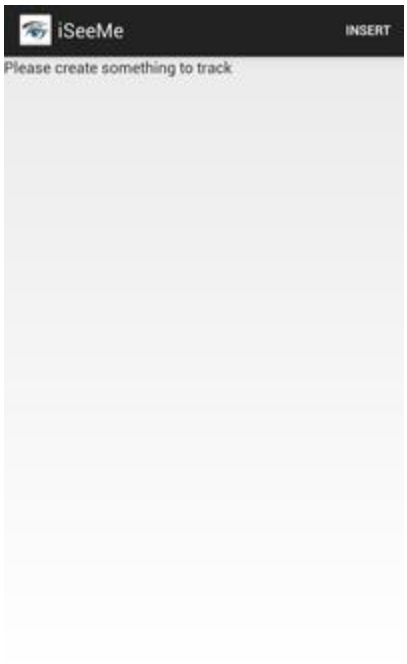

Figure 3: Main Screen Empty List

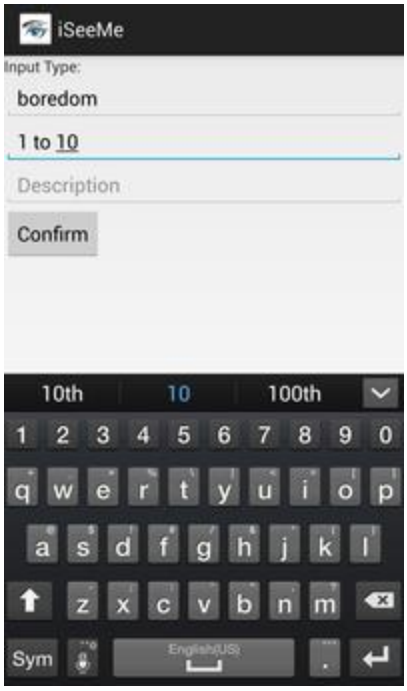

Figure 4: New Item to Track

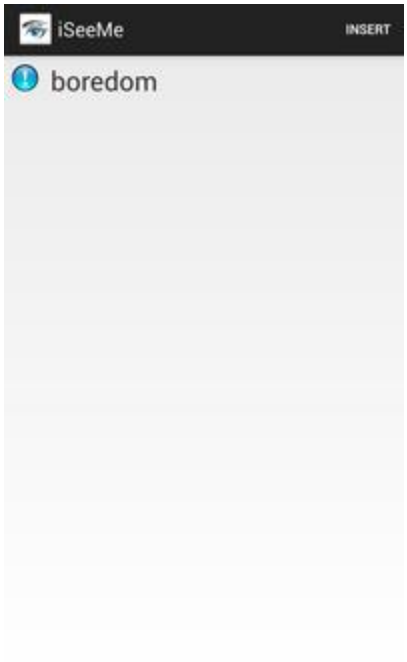

Figure 5: Main Screen List

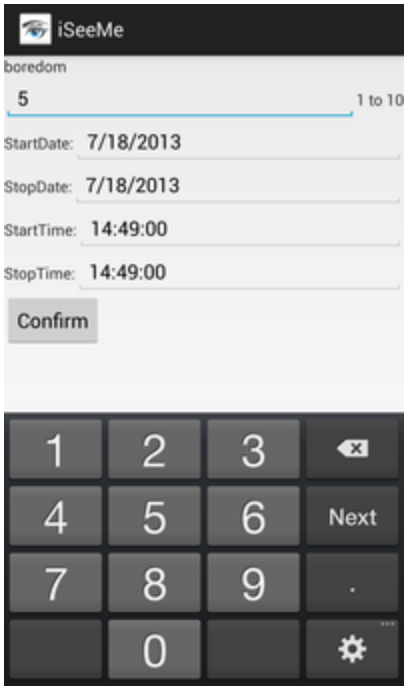

Figure 6: Enter Data

The iSeeMe database design does not restrict the source of data for user-defined data-types to just the user interface, though currently that is the only available option. Another possible source of this data is an external device capable of connecting to the phone and broadcasting an event that the user had previously associated with a manual input type. With not too much work, the iSeeMe service could be modified to receive this event and update the database on behalf of the device. More work would be required to develop the user interface component that allows the user to associate events from certain external devices with user-defined input-types. Identifying features of trusted external devices that the user wishes to receive data from could be stored in the rarely-used description column of the table in the iSeeMe database.

2.4.2 Data Visualization

2.4.2.1 Service Data

One option available to the user for viewing the data collected by the application focuses solely on the data collected by the iSeeMe service. This option was frequently used for debugging the implementation of the service, because it provided quick access to the most important data captured by the service. As the service runs, most of the data that is written to the database is also added to a static list object owned by the service. The "view service data" activity allows the user to fetch this list from the service and view it in textual form. By fetching data directly from the list, the performance hit associated with reading from the database and assembling meaningful string objects from the data is avoided.

The activity responsible for the screen that presents this list data also provides menu options that allow the user to manually stop and start the service, as well as an option to clear the list data. These capabilities were extremely useful when testing that the service could accurately capture the user starting, stopping, and switching between applications. The list maintained by the service is automatically cleared once every twenty-four hours, providing the user with a daily summary of the recently collected data and ensuring that the list does not consume too much memory. The following figure is a screenshot of the "view service data" interface discussed above:

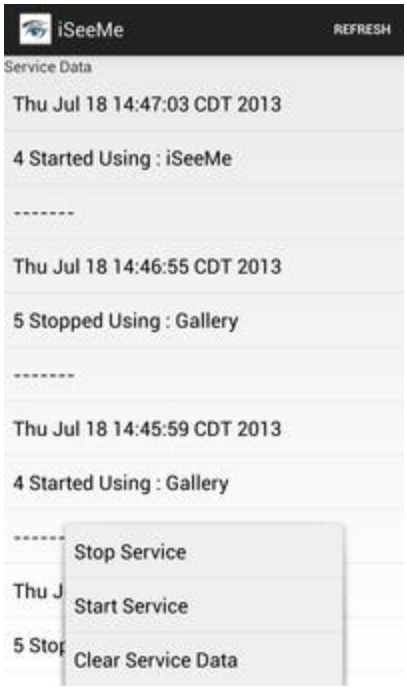

Figure 7: View Service Data

2.4.2.2 Database View

The iSeeMe application allows the user to view all of the data currently stored in the SQLite database. When the user selects the option to view the database, the application reads all the rows from the database and assembles an HTML table corresponding to the values for each column and row. This table is then displayed in a new activity screen utilizing an Android WebView. The WebView is a good choice for viewing the database, because it allows the user to easily scroll through the database and zoom in and out on areas of interest. The figure below is a screenshot of the "view database" interface discussed above:

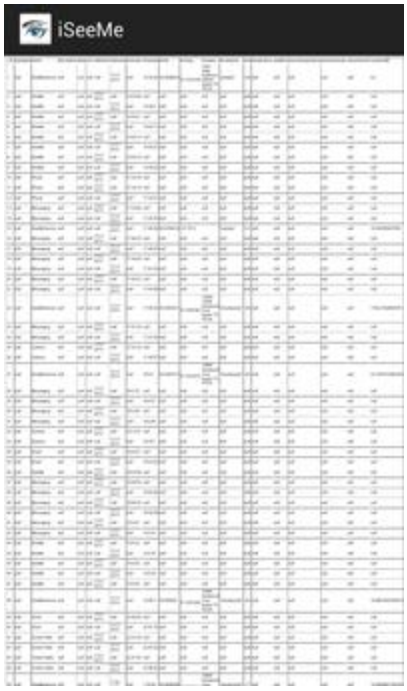

Figure 8: View Database Zoom Out

While the view shown above gives curious users a chance to see exactly how the application stores their data, the preferred solution would allow the users more control over what data is displayed and offer a means of querying the database for specific kinds of information. This capability is considered in the future work section of this report. The next figure shows an example of the current database view when zoomed-in to focus on a specific portion of the table.

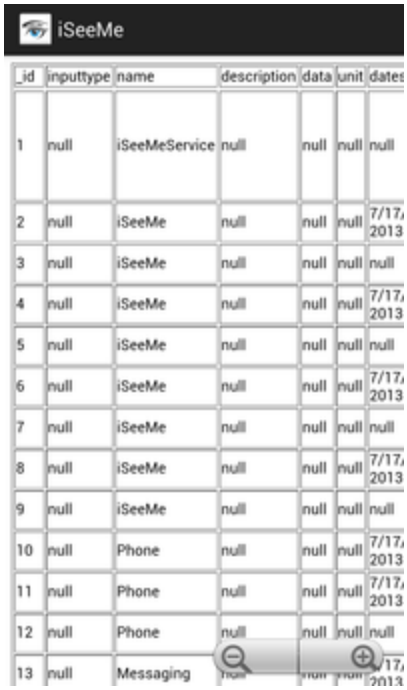

Figure 9: View Database Zoom In

2.4.2.3 Export and Analysis

The user may also choose to export the contents of the database to a .CSV file for analysis with an external application or for archival purposes. The process for generating the .CSV file is similar to that of generating the HTML table, with commas taking the place of HTML tags. A .CSV file is written to a folder called “iSeeMe” on the SD card (if available). This option is accessible through the menu on the main user interface screen, shown below:

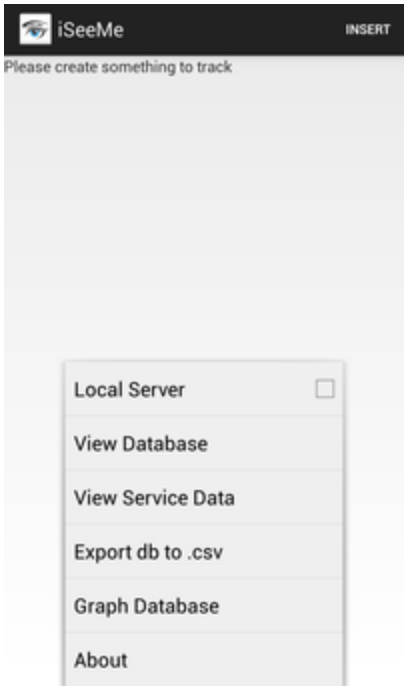

Figure 10: Main Screen Menu

The final screen in the iSeeMe user interface is accessible through the "graph database" menu option on the main screen. When this menu option is selected, the .CSV generation capabilities are exercised and a fresh .CSV file is written to the SD card. The filename is then passed to a new activity, which opens an HTTP connection to the iSeeMe server. The .CSV file is then sent to the server where analysis is performed using predefined R scripts. When the analysis is complete, the results are sent back to the application in the form of an HTML page. After the data has been received, a new activity is launched and the HTML output is displayed in a WebView. Examples of the output generated by the R scripts are shown in chapter 3.

Of course, sending personal data to an external server opens the door for potential security and privacy issues. Future versions of iSeeMe should include improvements to address these concerns. As the menu in Figure 10 shows, it is possible to host a local

version of the iSeeMe server. Highly concerned users could set up their own servers and only send data through their own local area networks. Or, if they chose to make their servers accessible from the Internet, they would at least know that they are in complete control over the servers that handle their data.

Chapter 3: Data Analysis

Currently, iSeeMe will generate a specific set of graphs based on the user's data. Ideally, iSeeMe would allow the user to choose exactly which things to graph and on which kinds of models the graphs should be based. Adding this support would require significant work not only to develop the Android user interface that allows the user to choose the data to graph and the type of model to build, but also to design and implement a protocol for communicating the user's choices to the server. If the user is only interested in a subset of the data, the best solution may involve pre-processing the data on the Android device—the decision of how to best handle the data in this case would require a substantial investigation into the pros and cons of the available options.

Right now, the iSeeMe server produces a single HTML report with individual summaries for each of the various data types. It also creates linear regression models for all two-factor combinations of manually-input data with data about each application's usage and produces graphs for the models where the correlation between the two factors appears to be significant. A similar procedure could be followed to generate models with three or more factors and/or utilize any of a variety of modeling techniques available in R without requiring any changes to the Android code.

The current process begins by reading the contents of the .CSV file into an R data frame. This .CSV file represents the entire database stored on an iSeeMe user's device and contains application usage data, location data, network data, and manually-input data. iSeeMe provides specific analytics for each of these sub-groups of data before trying to find correlations in the data between the sub-groups. The first step towards creating the graphs for each specific sub-group is to isolate the application data from the rest of the data.

3.1 Application Usage Graphs

After identifying which strings in the name column correspond to manual inputs, all records associated with those names are filtered out. All location and network data updates are also temporarily removed, so that only the records corresponding to the user opening and closing applications remain. Because start and stop times are stored in separate rows, forming a complete record of an application's usage requires matching these start and stop records together. After the data has been filtered, each "start record" for a particular application should be immediately followed by a "stop record"; the iSeeMe Android service writes a stop record for the last-most-recent application before writing a start record for the current-most-recent application when a user switches between two applications without first returning to the home screen. For several reasons, the database may not appear in the format described above. Due to bugs in earlier versions of the Android application, some situations could lead to the iSeeMe service failing to record some start and stop events. A stop event can also be missed when upgrading the iSeeMe application while the iSeeMe service is running. Finally, since the iSeeMe service records when the user starts and stops using the iSeeMe application itself, the last record in the .CSV file sent to the server when the user graphs the data will be a start record for iSeeMe, without a corresponding stop record (since the application is actively being used at that moment). All incomplete application records are filtered out before proceeding with the data analysis.

iSeeMe provides the user with a graphical representation of the number of times each application was used. The bar plot below is an example of how this data is displayed to the user:

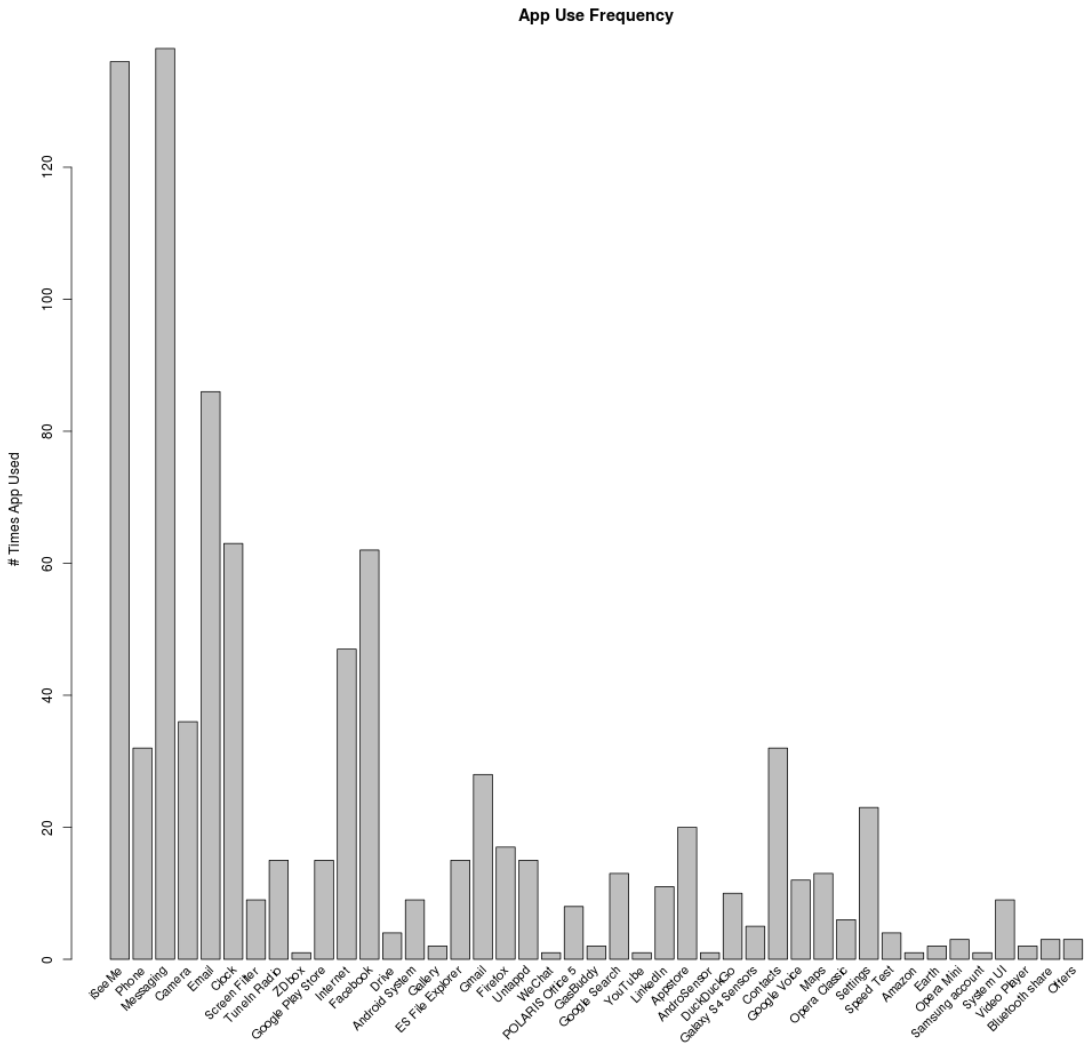

Figure 11: Application Usage Frequency

In addition to allowing the user to visualize the frequency that each application is used, iSeeMe also graphs the total duration that these applications were used in a similar bar plot:

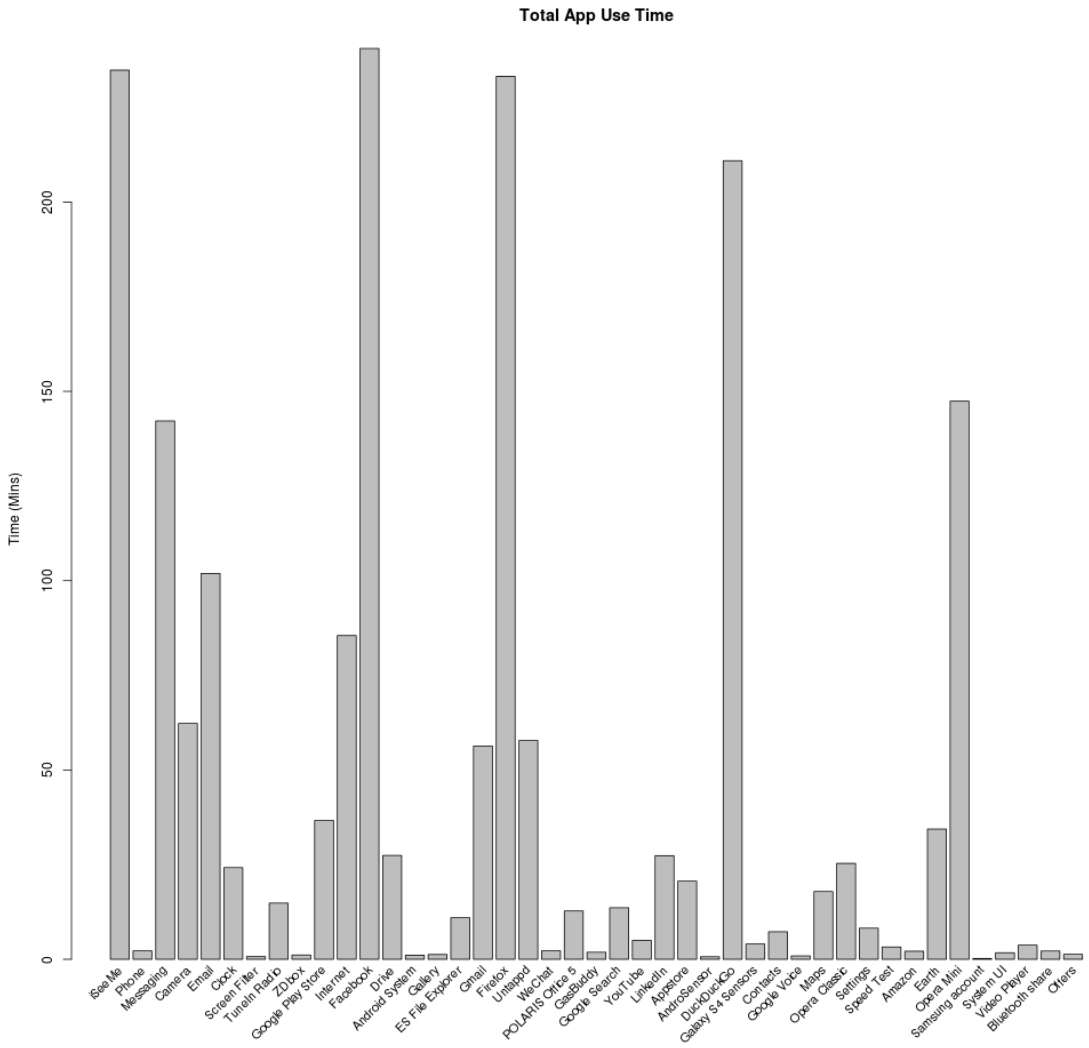

Figure 12: Total Duration of Application Use

Total duration is plotted against frequency for each application:

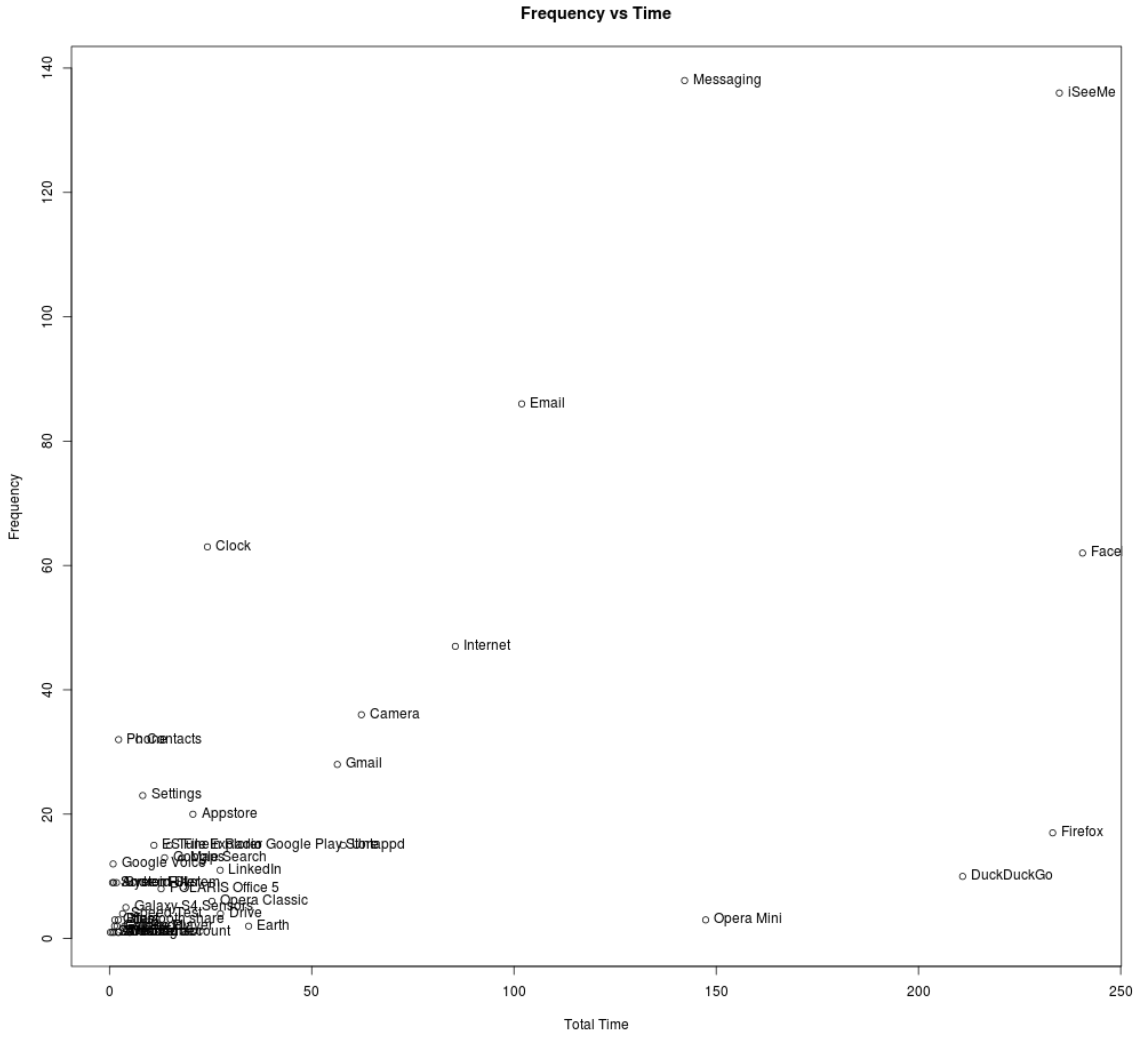

Figure 13: Application Usage Frequency vs. Total Duration

iSeeMe informs the user of the top five applications based on frequency of use (Figure 14), as well as duration (Figure 16):

App Name	Frequency
Messaging	138
iSeeMe	122
Email	86
Clock	63
Facebook	62

Figure 14: Top Five—Total Frequency

A graph, like the one below, shows the number times per day that each application above has been used:

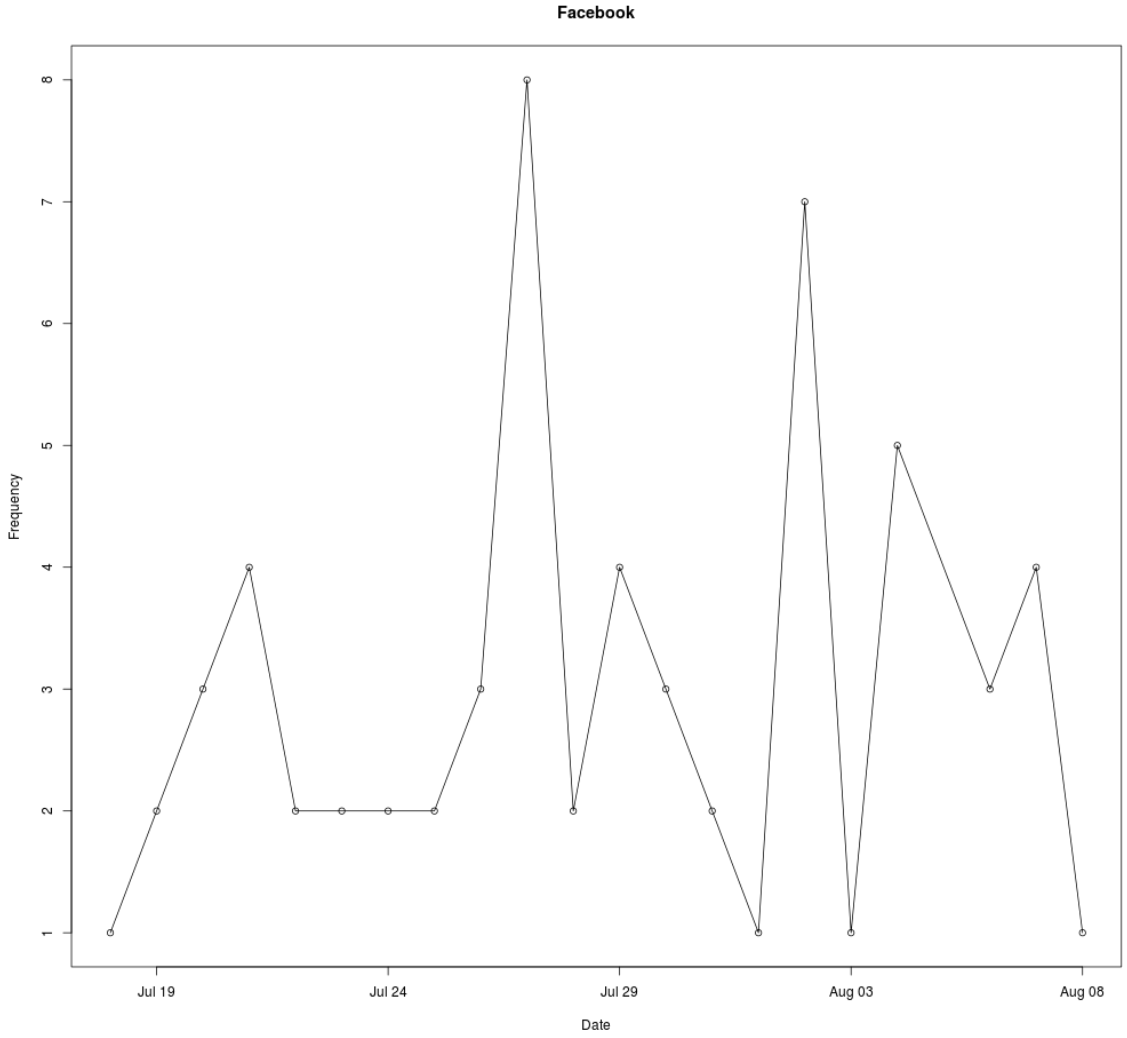

Figure 15: Single Application—Frequency per Day

App Name	Duration (mins)
Facebook	240.6
Firefox	233.2
DuckDuckGo	210.9
iSeeMe	205.1
Opera Mini	147.4

Figure 16: Top Five—Total Duration

A graph showing the total time spent using the application is also made for each app that would appear in the table above:

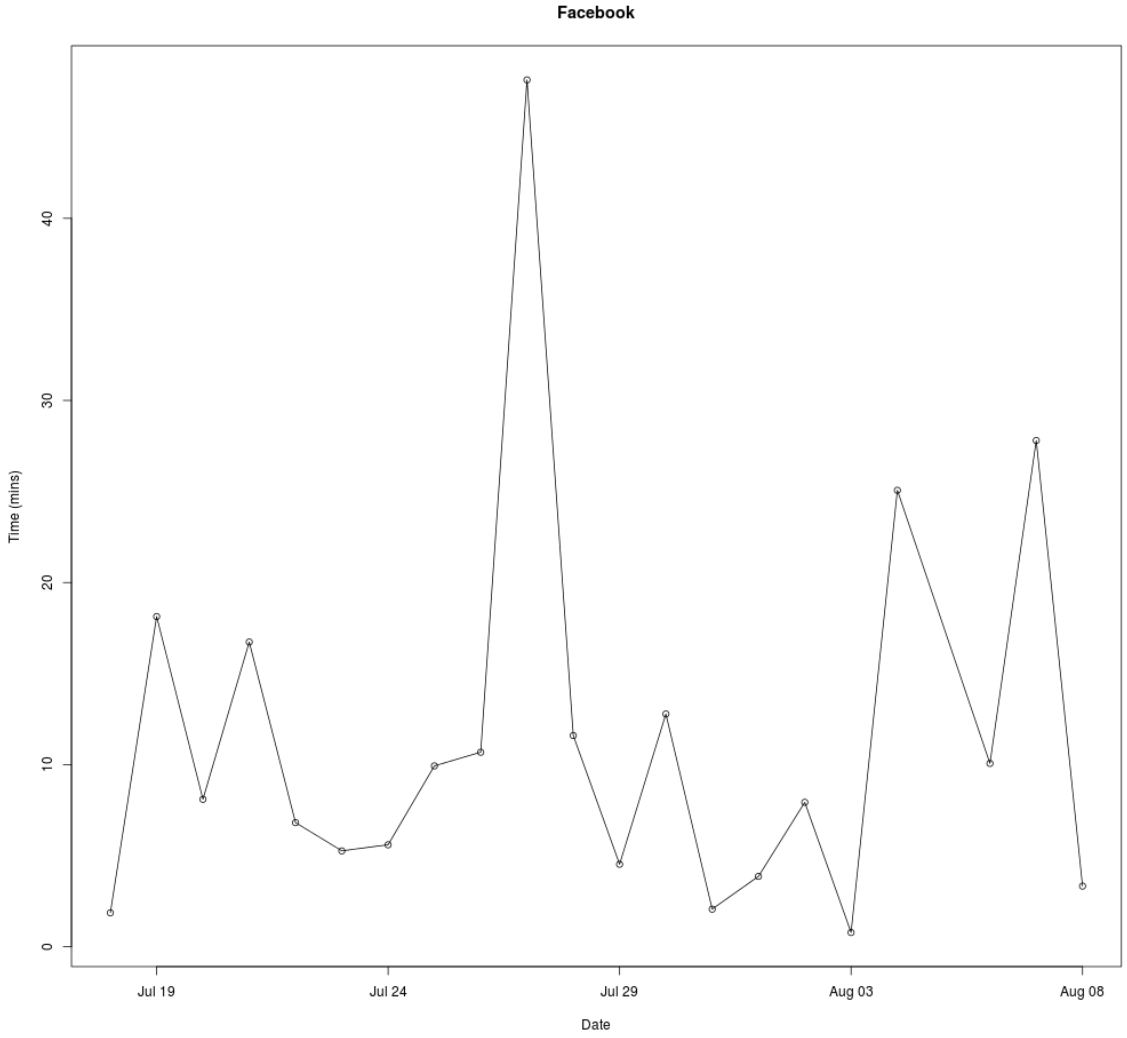

Figure 17: Single Application—Duration per Day

Similar graphs show the total combined frequency and total combined duration for all applications on a per-day basis:

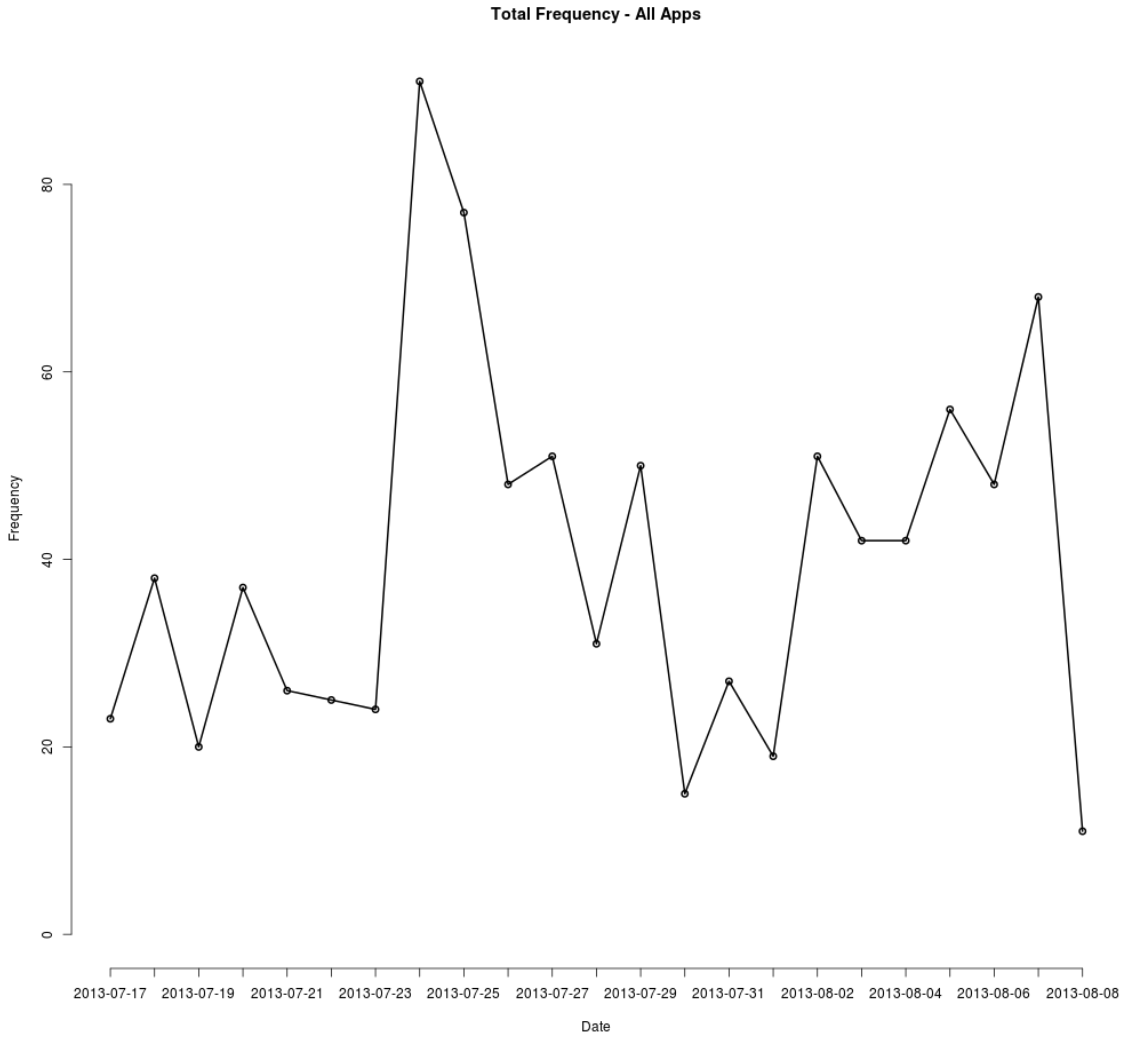

Figure 18: All Applications—Frequency per Day

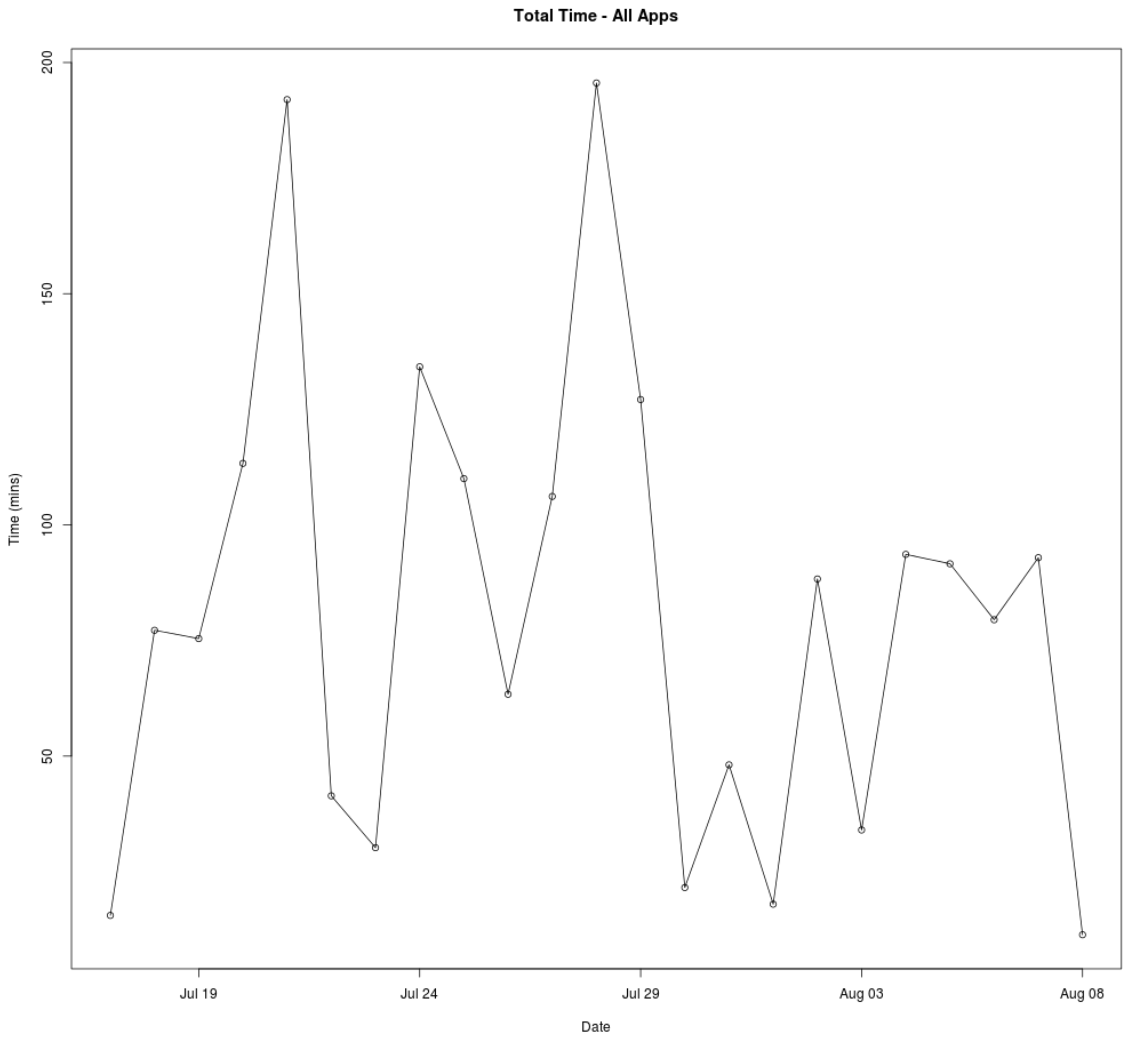

Figure 19: All Applications—Total Duration per Day

3.2 MANUAL INPUT GRAPHS

As the manually-input data was filtered out to generate the application usage graphs above, the opposite operation was performed to create graphs of just the manually-input data. The following simple graph shows the number of times the user manually input data for each of the user-defined manual input categories:

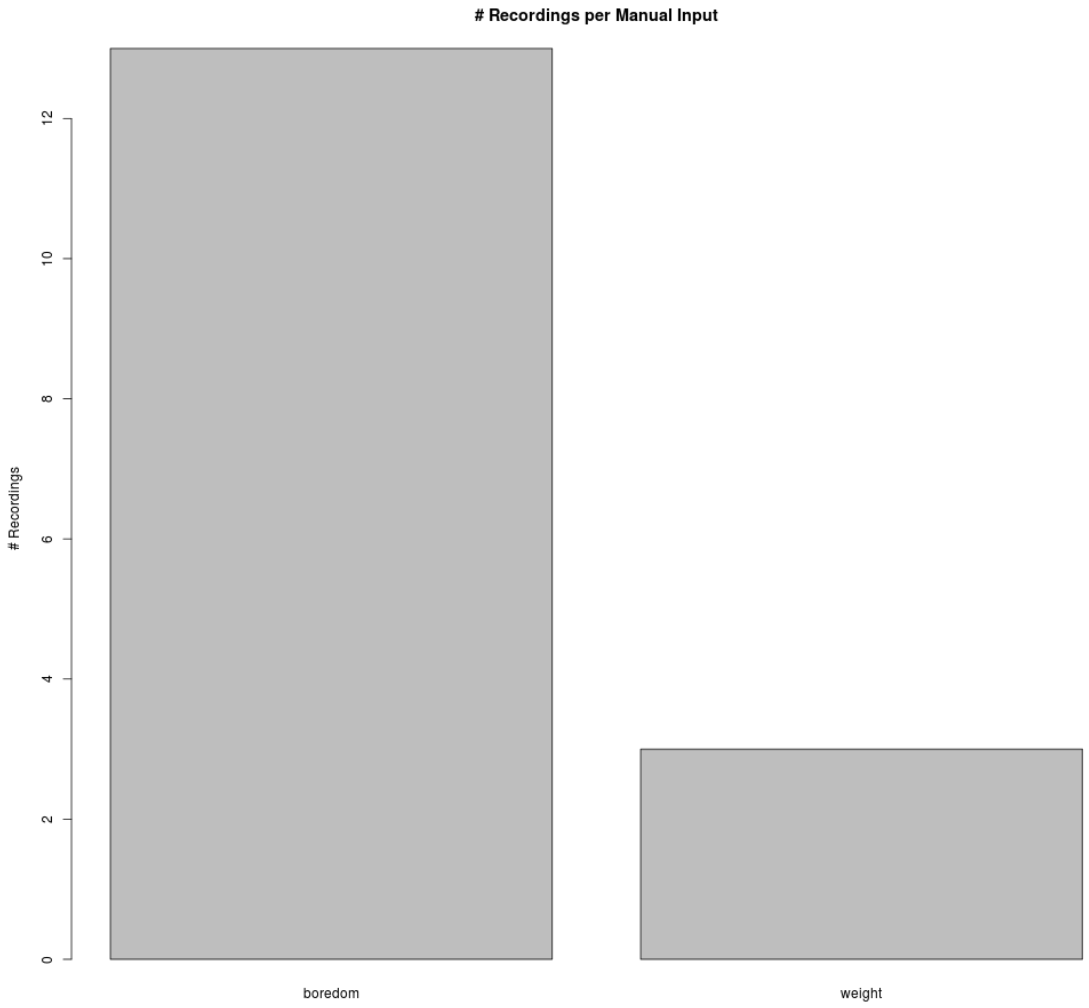

Figure 20: Recordings per Manual Input

The actual values that the user records under each category can also be visualized:

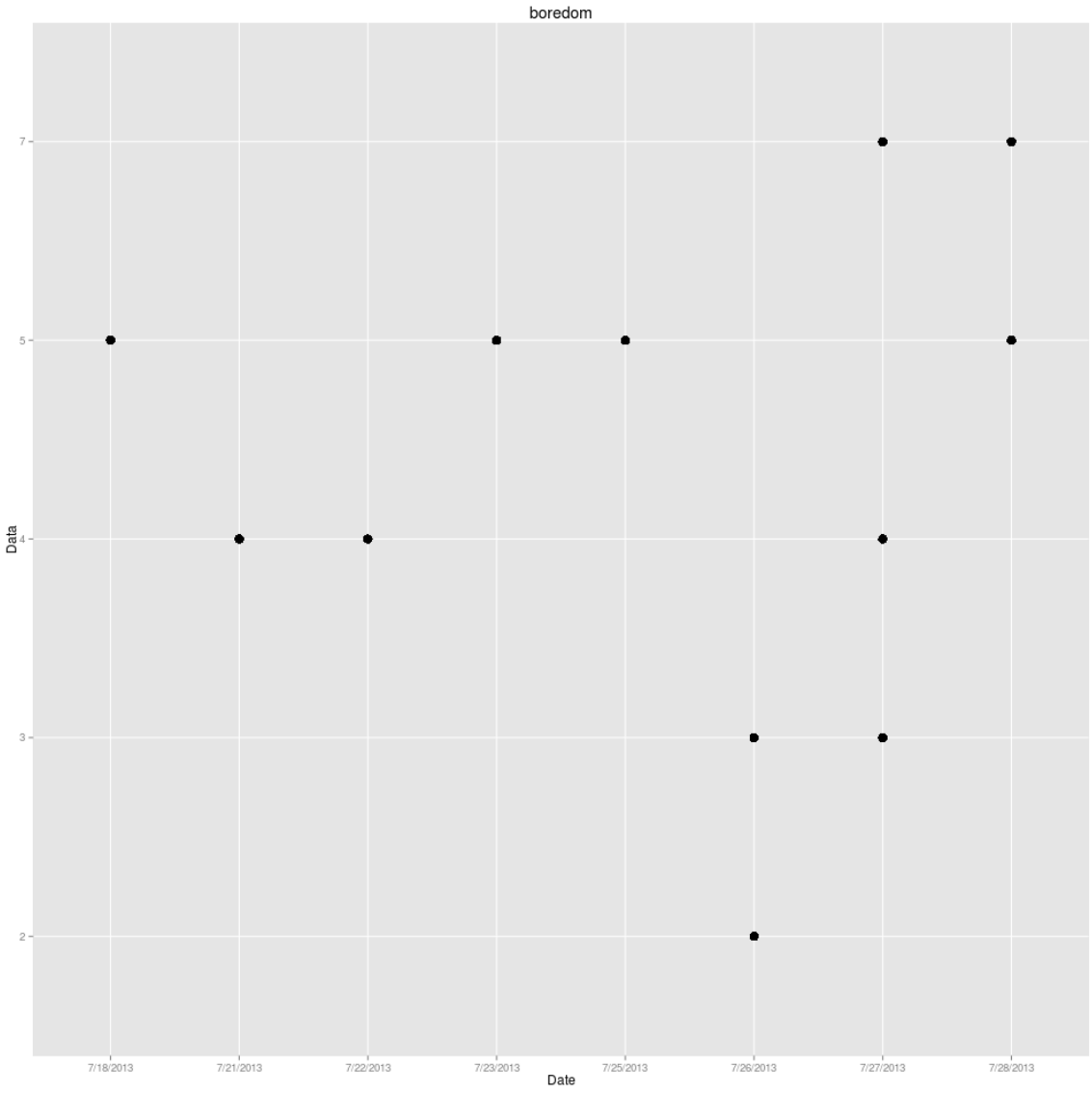

Figure 21: Manual Input over Time

Currently, the manually-input data is plotted on a per-day basis. When multiple records exist for each day all those records are displayed, but the exact time each record is created is not. Other options for displaying the data might include averaging all the records for a given day or plotting the data on a per-record basis. Several different

options were tested during the development of iSeeMe's graphing capabilities. Future versions of iSeeMe should allow the user some control over how to display this data.

3.3 Correlation Graphs

For each user-defined manual input, iSeeMe generates linear regression models of the manually input values against the frequency with which each application is used. iSeeMe provides graphs of these models if the Adjusted R Squared value of the model is above a certain threshold. A sample of one such graph is shown in Figure 16.

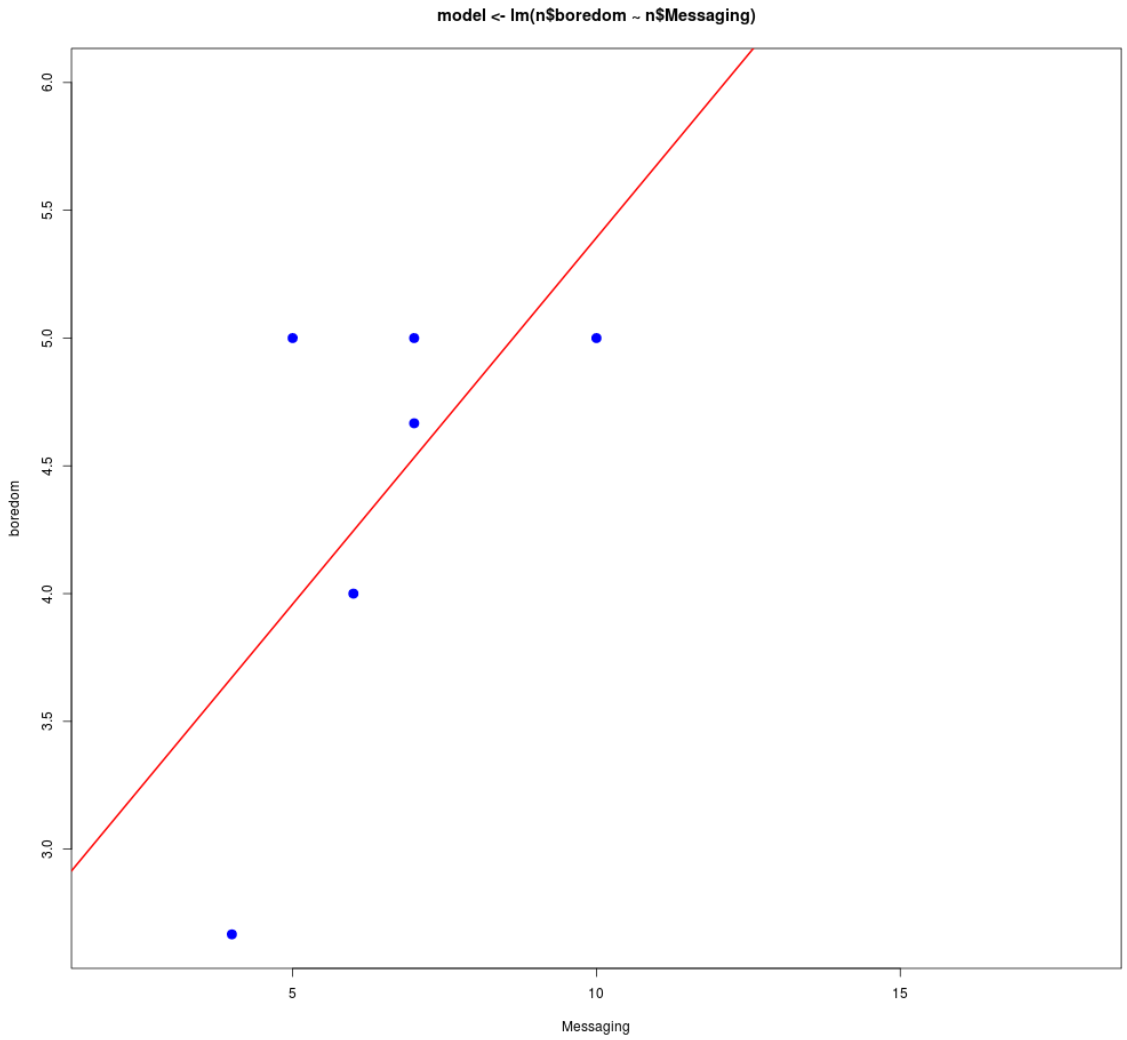

Figure 22: Linear Regression Example

The Adjusted R squared value for each model that is graphed is also reported as follows:

[1] "model <- lm(n$boredom ~ n$Messaging)"

[1] "Adjusted R Squared"

[1] "0.262401437269189"

A similar process could be used to graph total application use time against the manually-input data.

3.4 LOCATION GRAPHS

iSeeMe also creates a graph for the location data by plotting latitude against longitude on a map. The sizes of the red stars indicate the frequency that particular locations appear in the database:

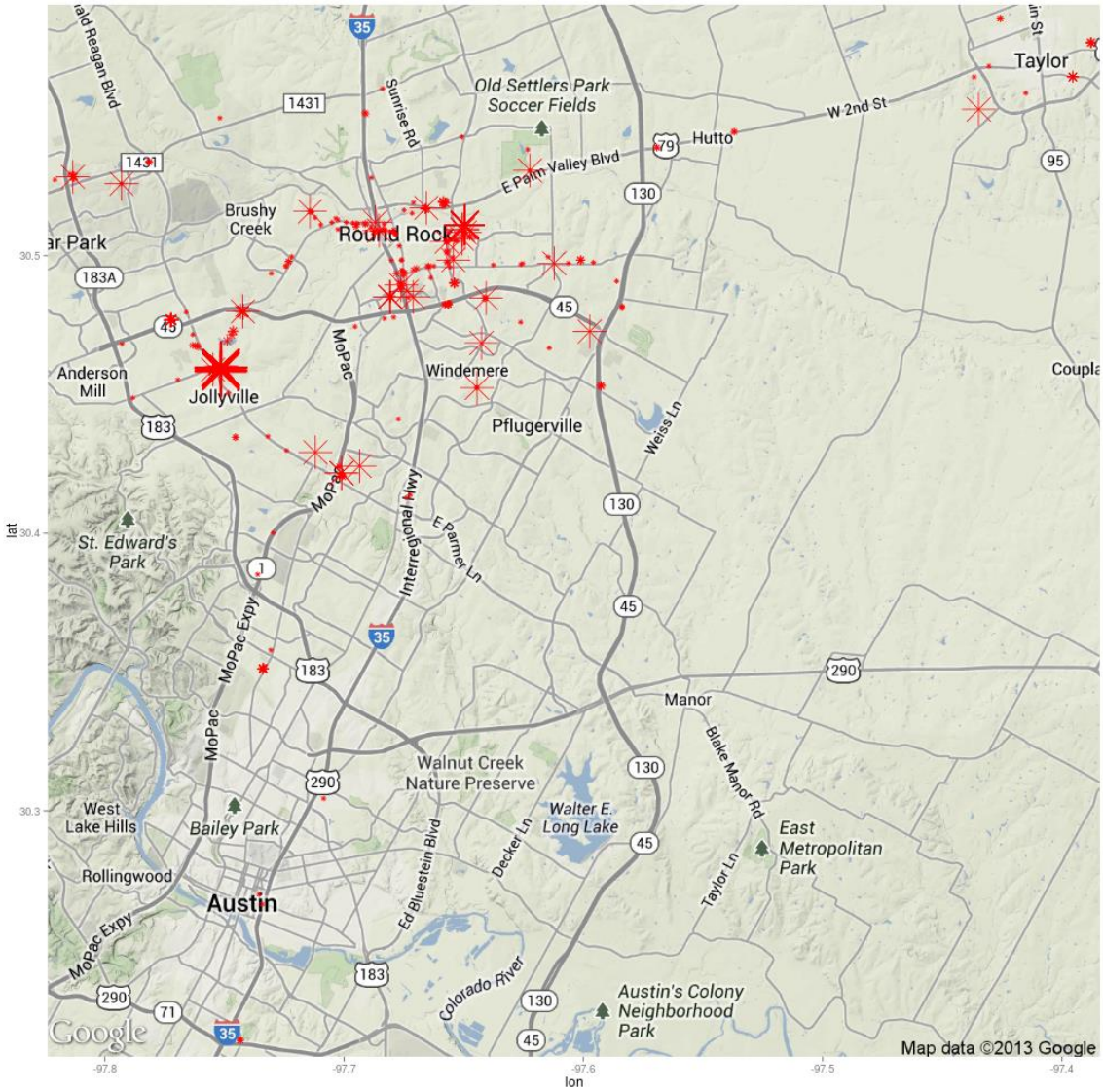

Figure 23: Location – Latitude & Longitude

Chapter 4: Challenges

4.1 Capturing the User's Activities

Android allows applications to implement certain features, only by requesting specific permissions from the operating system. These permissions are declared in an application's Android manifest file and are displayed to the user before the application is installed. A user can choose not to install an application based on the permissions that it requests. Most of these permissions involve access to specific kinds of data that the Android device can provide. The list of available permissions is quite extensive; currently, the Android developer documentation lists one hundred thirty different possible permissions. One particular permission of interest to the development of iSeeMe is SET_ACTIVITY_WATCHER. This permission is supposed to allow an Android application to, "…watch and control how activities are started globally in the system [12]." Unfortunately, this permission is not supported outside of a debugging or development environment in all recent versions of Android [13]. To overcome this challenge, iSeeMe must run a service in the background and constantly poll the operating system for the list of recently accessed activities, then read and parse the string representation of the currently running activity to determine whether an application has been started or stopped. Had the SET_ACTIVITY_WATCHER permission been supported, iSeeMe could simply receive a notification every time the current application changes, resulting in a simpler and likely more efficient design.

4.2 Battery Life vs. Sample Rate

An activity listed towards the end of the list may move towards the beginning of the list if it is used again. In that case, it no longer appears later in the list, so the history of activity usage is not preserved. For this reason, the service must run frequently,

requesting and storing previous lists of recent activities. If the service interval is too long, an instance of using an application could be completely missed. If the service interval is too short, battery life, performance, and the user experience suffer. A service interval of three seconds provides high assurance that that all the user's activities are captured. However, that assurance comes at the cost of incurring a significant drain on the battery.

Cellular phones maintain excellent battery standby time, because the device is allowed to enter low power states when it is not actively being used. The short service interval used in the early development of iSeeMe prevented the device from spending extended periods of time in these lower power states, wasting precious battery life. The solution to this problem was to dynamically adjust the frequency that the service runs, based on whether the screen was ON or OFF. When the service receives a notification that the screen has been turned ON, it sets the service interval to a short three seconds, enough to capture quick application changes. In a similar fashion, the service interval is extended to sixty seconds when the screen is turned OFF.

4.3 ANOMALIES IN DATABASE

A persistent challenge faced when developing this application was the fact that the database occasionally would not contain a record of when an application stopped being used, but did contain the record for it starting to be used. This inconsistency indicated that the iSeeMe service code responsible for determining when applications started and stopped being used contained bugs. While fixing those bugs required some effort, the fact that the database contained 'bad' data proved to be a challenge when developing the R code to analyze that data and exposed areas where the R code could be made more robust. Furthermore, attempts to fix the problems in the service required

updates to the application. In an attempt to provide the R code with a completely 'good' dataset, these application updates also updated the database version. Each time the database version was updated all the existing data was lost, meaning that smaller datasets were available to use when developing the R code used on the iSeeMe server.

Chapter 5: Conclusions

5.1 Evaluation

5.1.1 Flexibility

A primary motivation of this project was the need for an application that allows the user to record personally meaningful data without being limited by some preconceived notion of what kinds of things are important to the user. iSeeMe delivers this flexibility by providing a simple means for users to define categories for their data. By providing control over the service that collects data about the user, iSeeMe demonstrates that it can tolerate differences in user preference with respect to privacy. The single-table database design remains flexible to a dynamic range of user-defined data types.

5.1.2 Extensibility

iSeeMe features an extensible design with an emphasis on reducing the impact of future enhancements and bug fixes on its users. When iSeeMe matured to a point where the application could be used in an every-day, real-world environment, eight distinct versions were tested on a handful of different devices. While the results of some of the early testing required changes to the database, resulting in the loss of user data across upgrades, the final few versions upgraded seamlessly as iSeeMe became more tolerant of imperfections in the recorded data.

The fact that iSeeMe uses a content provider for access to the database also speaks to the fact that iSeeMe was designed with extensibility in mind. The iSeeMe database design also does not impose any restrictions on the sources of input data. iSeeMe could be marketed along with several other sister applications that easily plug-in to the existing architecture to add additional features. Future versions should be able to

support new devices capable of interfacing with Android phones as data sources without requiring any modifications to the database, ensuring that the user's data is not lost during any necessary application upgrades.

In addition to a flexible database design that allows iSeeMe to accommodate new sources of data in the future, the client-server architecture of the iSeeMe solution provides a means for extending iSeeMe's analysis capabilities without interfering with the code running on a user's device. By outsourcing the analysis and graphing functions, iSeeMe benefits from the powerful R language and the many add-on packages available that are continuously being developed by a community of experts working in the field of data analysis.

5.1.3 Transparency

iSeeMe provides the user with several different options for accessing the data captured by the application. The user can easily and quickly view a list of recently recorded data. The contents of the entire database can be viewed as a table or exported to a .CSV file for analysis or inspection using other software packages. None of the data that the application collects is hidden from the user. The graphing capabilities of R provide a means for the user to visualize the data in a variety of forms.

5.2 MARKETABILITY

While the self-quantifying apps listed in Table 1 provide the user with the option to track one or a few specific kinds of data, iSeeMe is capable of tracking just about any kind of data that may be of interest to the user. With the growing trend towards leading a data-driven life [7], an all-in-one solution like iSeeMe might be more marketable to individuals interested in tracking many different aspects of their lives. Unlike applications designed specifically for counting calories or recording sleep patterns that

target very specific groups of users, the potential market for an application like iSeeMe extends to a much wider audience.

The decision to use R to analyze and graph the data required the setup and configuration of a server to host the R environment. There is a growing trend in software development to use "cloud" computing services like Google App Engine[27] and Amazon Web Services[28], but iSeeMe does not rely on one these services and instead uses a more traditional client-server approach for several reasons. While one of these services could have been used for this proof-of-concept application at little or no cost, if iSeeMe were ever to be deployed on a large scale, the costs associated with relying on popular cloud offerings could quickly become prohibitive. In contrast, the iSeeMe server was constructed using freely available software: a Linux operating system, Apache webserver, PHP and R. These design choices not only avoid the costs associated with being tied to a proprietary commercial cloud solution, but also allow for the possibility that iSeeMe could be provided to the user as a complete client-server package, with the option of self-hosting an iSeeMe server.

The details of exactly how iSeeMe will be marketed in the future are not clear at this point. It is possible that instead of being released as a commercial product, iSeeMe grows as a freely available open source project. By not relying on commercial cloud providers, the options for how iSeeMe can be distributed remain open. Furthermore, certain users may be drawn to iSeeMe because it offers them access to the very data that is so often capitalized on by larger tech companies like Google, Facebook and Amazon. Some of these users may want to keep their personal data out of the hands of such companies to the greatest extent possible. They might be dissuaded from using a solution

[27] https://developers.google.com/appengine/
[28] http://aws.amazon.com/

that strives to shed light on the data-hungry nature of today's tech companies, while relying on the services provided by such companies to process their data.

5.3 Future Work

5.3.1 Security Enhancements

When dealing with personal data, security is a top concern. If iSeeMe were to be launched as a public product, several improvements would be made to address security concerns. At a minimum, the connection between the iSeeMe Android app and the iSeeMe server would require some form of authentication before any data is ever exchanged. Data stored on the device and sent to the server would also require strong encryption.

5.3.2 Database Optimization

While the current iSeeMe database design offers several benefits, many aspects of its implementation could be greatly improved. The data storage solution was one of the first components to be implemented. Because iSeeMe is a data-centric application, having a working means to store and retrieve data was essential to developing the rest of the system. An early decision was made to make every write to the database add an additional row, avoiding the complexity associated with finding and updating existing rows and the risk of overwriting existing important data. This strategy means that for each time a user opens and closes an application, two rows in the database are required to capture the event. While the Android code was made simpler by following this design, the R code on the server became more complex. This design choice also impacted application performance, particularly noticeable when the HTML table representation of the database is created, because of the additional string manipulation required to build the

table. A similar effect is noticed when exporting the database to .CSV format, but to a lesser degree. This database design choice would be reconsidered if version 2.0 of iSeeMe is ever developed.

Early versions of iSeeMe recorded application usage by storing the entire string representing the base intent of the application. These strings tended to be much longer than the actual application name and consumed additional space in the database, as well as more time during operations that manipulated those strings. The base intent string is also not as meaningful or visually appealing from a user perspective, so the capability to store the actual (usually much shorter) application name instead of the base intent was added. Since the 'name' column in the database is used for both application names and for manual input names, the possibility of a user creating a manual input with the same name as an existing application creates potential for confusion later. Future versions of iSeeMe may explore the option of using a unique ID derived from the application's base intent to clearly distinguish between applications and manual inputs.

Finally, since performance issues associated with transforming the database into HTML tables and .CSV files have been noticed when dealing with larger databases, iSeeMe 2.0 would include several changes to improve performance in those areas. When generating a .CSV or HTML representation of the database, iSeeMe would incrementally add to the results of previously generated files by only requesting and converting newer rows from the database. Additionally, the practice of sending data to the iSeeMe server in .CSV format would be reconsidered. Significant benefits might be realized by compressing the raw SQLite database file, sending it to the server, and using the additional resources available to the server to export the data in a format suitable for analysis with R.

5.3.3 User Customization and User Interface Improvements

Future versions of iSeeMe should provide more opportunities for users to customize their experience with the application. Just as iSeeMe allows users to decide exactly what information they would like to track, iSeeMe should provide a way for users to choose how that data should be interpreted. Currently, the user has no control over how much or what kind of data iSeeMe will display, export or analyze. For each of these three primary operations the entire database is read and processed; users should be able to select specific manual inputs, applications, and time periods of interest. Solving these challenges will likely require some method of translating the user's desires into SQL statements and fetching the appropriate data from the database.

By not restricting the values that the user can input, iSeeMe naturally allows the user to determine the scale by which that data should be interpreted. iSeeMe could also let the users define specific labels to associate with the data they wish to enter. For example, instead of rating happiness on a scale from one to ten, the user may associate a rating of zero with the label, "completely miserable" and ten with, "I just won the lottery!" Allowing users to associate ideas with values for the data they enter might make for a more meaningful and accurate means of recording data.

A major obstacle to iSeeMe's usefulness to the user is the sparseness of the available data. When users do not consistently and regularly record measurements under the categories they have defined, less data is available from which to draw conclusions. iSeeMe should allow users to set up reminders for entering data at regular intervals.

5.4 Final Remarks

As a proof-of-concept application, iSeeMe successfully meets the goals outlined at the start of this project. Users are capable of recording their mobile application usage and can also keep track of user-defined data. iSeeMe provides several methods to view

the collected data, as well as a means to export data for future analysis. The client-server architecture combined with a flexible data storage design solution facilitates future upgrades to both the iSeeMe Android application and the R code used to analyze the data. While the current capabilities of the analysis component only exercise a handful of the numerous possible modeling techniques that the R language can provide, the existing implementation works to provide users with summaries of their application usage and manually input data. An effort is made to automatically detect possible correlations between features present in these two classes of data using simple linear regression models.

As iSeeMe was being developed, a handful of people provided feedback relating to the usefulness, features, and user interface of the application. iSeeMe was seen as an interesting and potentially valuable application. The biggest criticisms of the application revolved around the presentation of the collected data. For example, iSeeMe originally displayed the location data as points on a grid representing latitude and longitude. Unanimously, the feedback to this presentation method was that the location data really needed to be shown on map for proper context. This suggestion was adopted and the current version of iSeeMe adds a map to the graph of location data.

Another common recommendation was that users should be able to choose specific applications and manual inputs when viewing their data. This feedback is acknowledged as an area for future work. While more testing, data and user feedback are necessary to evaluate the results of modeling the kind of data that iSeeMe captures, such evaluation is beyond the scope of this report. The present solution proves the feasibility of the Android-client, R-server design, clearing the road ahead for future enhancements.

Glossary

Activity	An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with setContentView(View). While activities are often presented to the user as full-screen windows, they can also be used in other ways: as floating windows (via a theme with windowIsFloating set) or embedded inside of another activity (using ActivityGroup) [11].
Intent	An intent is an abstract description of an operation to be performed. It can be used with startActivity to launch an Activity, broadcastIntent to send it to any interested BroadcastReceiver components, and startService(Intent) or bindService(Intent, ServiceConnection, int) to communicate with a background Service. An Intent provides a facility for performing late runtime binding between the code in different applications. Its most significant use is in the launching of activities, where it can be thought of as the glue between activities. It is basically a passive data structure holding an abstract description of an action to be performed [14].
BroadcastReceiver	Base class for code that will receive intents sent by sendBroadcast() [15].
View	This class represents the basic building block for user interface components. A View occupies a rectangular area on the screen and is responsible for drawing and event handling… There are many specialized subclasses of views that act as controls or are capable of displaying text, images, or other content. [16].
WebView	A View that displays web pages. This class is the basis upon which you can roll your own web browser or simply display some online content within your Activity. It uses the WebKit rendering engine to display web pages and includes methods to navigate forward and backward through a history, zoom in and out, perform text searches and more [17].

Bibliography

[1] Brenner, Joanna. Pew Internet: Mobile. Pew Internet & American Life Project, June 6, 2013, http://pewinternet.org/Commentary/2012/February/Pew-Internet-Mobile.aspx, accessed on June 13, 2013.

[2] Mullin, Joe. How much do Google and Facebook profit from your data? Ars Technica, October 9, 2012 http://arstechnica.com/tech-policy/2012/10/how-much-do-google-and-facebook-profit-from-your-data/, accessed on July 28, 2013.

[3] Davenport, Thomas H. and D.J. Patil. Data Scientist: The Sexiest Job of the 21st Century. Harvard Business Review, October 2012 http://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century/, accessed on July 1, 2013

[4] Troianovski, Anto. Phone companies sell customer data. The Wall Street Journal. May 22, 2013. http://money.msn.com/technology-investment/post.aspx?post=cd675890-6124-4ce7-9d82-5f5292d411be, accessed on July 1, 2013.

[5] Simonite, Tom. What Facebook Knows. MIT Technology Review, June 13, 2012. http://www.technologyreview.com/featuredstory/428150/what-facebook-knows/ , accessed on July 2, 2013.

[6] Facebook Inc. Facebook Home Settings & Security. 2013. https://www.facebook.com/help/559492187415760/, accessed on July 17, 2013.

[7] How your smartphone is keeping track of you: Apps secretly monitor users to target advertising campaigns. Mail Online, December 20, 2010. http://www.dailymail.co.uk/sciencetech/article-1340107/How-smartphone-keeping-track-Apps-secretly-monitor-users-target-advertising-campaigns.html, accessed on July 2, 2013.

[8] Ashbrook, Tom. The Quantified Self. On Point with Tom Ashbrook, April 26, 2012. http://onpoint.wbur.org/2012/04/26/the-data-driven-life, accessed on June 4, 2013.

[9] SQL As Understood by SQLite. SQLite. http://www.sqlite.org/lang_altertable.html, accessed on June 22, 2013.

[10] Buneman, Peter. Semistructure data. In Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems (PODS '97). ACM. http://doi.acm.org.ezproxy.lib.utexas.edu/10.1145/263661.263675.

[11] Google Inc. Activity. Android Developer Reference. http://developer.android.com/reference/android/app/Activity.html, accessed on June 22, 2013.

[12] Google Inc. Manifest Permission. Android Developer Reference. http://developer.android.com/reference/android/app/Activity.html, accessed on June 23, 2013.

[13] Google Inc. Manifest Permission. Android Developer Reference. http://developer.android.com/reference/android/Manifest.permission.html#SET_ACTIVITY_WATCHER , accessed on June 22, 2013.

[14] Google Inc. Intent. Android Developer Reference. http://developer.android.com/reference/android/content/Intent.html, accessed on June 23, 2013.

[15] Google Inc. BroadcastReceiver. Android Developer Reference. http://developer.android.com/reference/android/content/BroadcastReceiver.html, accessed on June 23, 2013.

[16] Google Inc. View. Android Developer Reference. http://developer.android.com/reference/android/view/View.html, accessed on July 5, 2013.

[17] Google Inc. WebView. Android Developer Reference. http://developer.android.com/reference/android/webkit/WebView.html, accessed on July 5, 2013.