Project Number: GFP-1301

Developing an Android Framework and Exemplar App for WPI Suite

A Major Qualifying Project Report:

Submitted to the Faculty of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

By:

Mark Fitzgibbon

Sam Lalezari

Nathan Longnecker

Date: April 29, 2014

Approved:

Professor Gary Pollice, Major Advisor

Keywords:
1. Software Engineering
2. WPI Suite
3. Android

Abstract

This project extended WPI Suite to add a framework, example code, and documentation that enables students to create their own Android apps that integrate into WPI Suite. The three parts of this project included creating Marvin, a framework that handles common login functionality, creating a calendar app for Android that implements Marvin, and extending the existing WPI Suite wiki to document the process of creating an Android app for WPI Suite with Marvin.

Acknowledgements

We would like to thank Professor Gary Pollice for his guidance and support during this project.

We would also like to thank the previous WPI Suite MQP teams for their work during their 2012-2013 MQPs and continued support this year.

Contents

List of Figures

1. Introduction

Software engineers work on large projects in teams, but in most computer science classes students do not experience such work environments. Many courses require students to work on small projects individually or with only one or two partners. Worcester Polytechnic Institute (WPI) offers a software engineering course that helps to prepare students for software development in industry. Student programmers must be able to learn how to effectively coordinate with their team, analyze requirements, and track defects, all while learning the structure of the large code base that they are building onto. Previous Major Qualifying Project (MQP) groups developed WPI Suite, a modular application platform that is designed to be extensible so that students can add modules without worrying much about the underlying structure of the service. WPI Suite consists of a core server application and Janeway, a client framework, which supports the development of desktop modules.

The software development industry does not focus only on desktop applications. Due to a rise in smartphone and tablet sales around the world, mobile apps are in high demand. Mobile development is becoming a major part of software engineering, and aspiring software engineers should be exposed to mobile development.

In order to extend the software engineering course to allow professors to teach mobile development, this MQP creates a mobile framework to extend WPI Suite and an exemplar calendar module using that framework. This mobile framework sets up an Android app that can easily be built upon and handles the connection with the WPI Suite core to expedite and simplify the student's development process. The exemplar module includes extensive documentation to help students who are tasked with developing their own mobile apps. The goal of the module is to help students get started quickly by providing clear example code that they can use as their guide during the development process.

2. Background

The first section of this chapter establishes the history of the Android™ platform and reviews the differences between the releases to provide a broad picture of the challenges involved in developing an app that supports all Android devices. In section 2.2, we discuss the support library that we used to maintain support of older versions of Android while developing for the newer versions. Next, we describe some of the mobile Graphical User Interface (GUI) design patterns put forth by Google Inc. that we followed when designing our calendar app. Lastly, we discuss previous mobile calendar app designs that influenced the design of our calendar app.

2.1 Android History

The Android operating system started in 2003 when Andy Rubin, Rich Miner, Nick Sears, and Chris White founded Android Inc. Google bought Android Inc. in July 2005, but it was not until November 5, 2007 that Google Inc., along with T-Mobile®, HTC Corporation, Qualcomm Incorporated, Motorola Mobility, Inc. and others, announced that they would be using the Android platform in their products. The group of thirty-four companies, called the Open Handset Alliance™ described Android as "a fully integrated mobile 'software stack' that consists of an operating system, middleware, user-friendly interface and applications" (Open Handset Alliance). The first version of the Android Software Development Kit (SDK) was released a week later, giving the open source community its first look at the new operating system.

The first device to use Android was the T-Mobile G1 (Figure 1), which launched in the United States on October 22, 2008. Android 1.0 included features such as a notification window, widgets, and Gmail™ integration, allowing the phone to make use of some of Gmail's more unique features like archiving and labelling. The notification window provides a single screen for users to view different kinds of incoming information such as text messages, emails, and alarms; a feature that the top Android competitor, the Apple® iOS, did not implement until three years later. Android 1.0 included the Android Market™ media store, which allowed users to download new apps for their device (Verge Staff).

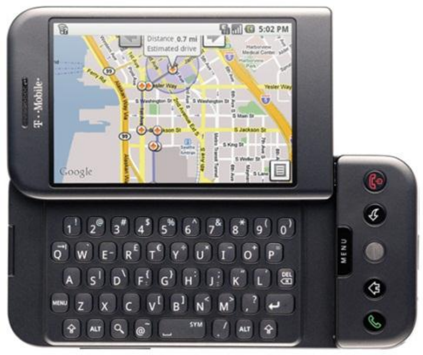


Figure 1 – The T-Mobile G1

Since its release, Android has grown rapidly, and as of September 2013 Android can be found on 80% of all smartphones worldwide (Deleon). This has helped make Android one of the top platforms for third party app developers, and as of 2013 seventy-two percent of developers were developing for the Android platform (Developer Nation).

Each major release of Android has conventionally been named after a something sweet. The first two names—Android 1.0 and 1.1—were never released to the public, but each of the versions since then have been named after sweets alphabetically, starting with Android 1.5 "Cupcake." Next was Android 1.6 "Donut," Android 2.0 and 2.1 "Eclair," Android 2.2 "Froyo," Android 2.3 "Gingerbread," Android 3.x "Honeycomb," Android 4.0 "Ice Cream Sandwich," Android 4.1, 4.2, and 4.3 "Jelly Bean," and the most recently released is Android 4.4 "Kitkat."

2.2 Support for Older Versions of Android

Since Android's initial release, regular updates have added a multitude of new features. These features can create inconsistencies between versions, creating challenges for developers wishing to create an app that is supported on all versions of Android. In section 2.2.1, we summarize the main features that were introduced in each version, and in section 2.2.2 we discuss the Android support library as a tool for overcoming the challenges of supporting software across different versions of the development platform.

2.2.1 New Features in Android Releases

Android 1.5 "Cupcake" added an on-screen keyboard, video capture and playback, and extensible widgets. The on-screen keyboard allowed touchscreen-only Android phones to enter the market, and extensible widgets allowed third party developers to create simple apps that could be installed on the home screen of Android devices (Verge Staff). Android 1.6 "Donut" added Code Division Multiple Access (CDMA) support and support for different screen resolutions. These improvements allowed more carriers to support Android and gave device makers greater freedom when creating devices with different screen sizes (Verge Staff).

The next major release was Android 2 "Eclair" in November 2009. This release added support for multiple accounts, Google Maps™ mapping service, as well as keyboard and browser improvements. Multiple account support allowed users connect their device to more than one Google account and sync data across those accounts (Verge Staff). Android 2.2 "Froyo" added Wi-Fi hotspot functionality, Adobe® Flash® support, and Android Cloud to Device Messaging (C2DM), which enabled push notifications. Push notifications allow servers to notify clients when new information is available, saving clients from needing to periodically poll the server for updates.

Android 2.3 “Gingerbread” added a number of relatively minor improvements that created a large overall improvement in the platform. Gingerbread also improved Android support for mobile gaming by giving app developers lower-level access to audio, graphics, storage, and controls, allowing developers to write apps that performed better than in earlier versions. These improvements helped to give Google a foothold in the mobile gaming market (Verge Staff).

Android 3 “Honeycomb” was released in 2011. Honeycomb was designed for tablets in addition to mobile phones. The platform stopped requiring the standard physical buttons that were on all previous Android devices, instead those buttons were moved to a new system bar at the bottom of the screen. This gave the system control of when to show or hide them, and the ability to change them in situations that they were not needed. On the system bar, a multitasking button was added that allowed users to easily see open apps and switch between them quickly. Honeycomb introduced the action bar, similar to a dedicated status bar for each app. The action bar can display frequently accessed options and context menus. Since Honeycomb’s release, the action bar has become a key attribute of many apps, and a special support library was added that allowed developers to implement action bars on previous versions of Android as well (Verge Staff).

In Android 4.0 “Ice Cream Sandwich,” the notifications system was updated to allow individual removal of notification items with a swipe, to improve upon the original choices of either removing all notifications or opening them. Android also introduced home screen folders, originally an iOS exclusive feature, and a fully modifiable home row, compared to the original home row which always fixed the phone and browser apps in place. Android also brought improvements to Near Field Communication (NFC) functionality to allow “Android Beam,” an open feature which allows two phones to transfer data by physically touching them together, with no need for a cable (Verge Staff).

The Android 4.1, 4.2, and 4.3 updates are referred to as “Jelly Bean,” and introduced dynamic notifications, allowing notifications to hold more viewable info without opening the related app. The updates also allowed widgets to have variable size based on the amount of space available on a home page, and allowed for “daydream” widgets which are viewable from the lock screen. Additionally, 4.3 introduced OpenGL to the Android platform, allowing developers more control of the graphics portion of their apps. Android 4.4 “Kitkat” provided a visual overhaul as well as better resource management, allowing the operating system to be run on systems with as little as 512 MB of RAM (Verge Staff).

2.2.2 Support for Different Versions of Android

One of the largest challenges for Android developers is dealing with version fragmentation. Currently there are eight Android versions still in use (Open Signal). This makes it extremely challenging for developers to make use of new features while still maintaining support for users on older versions of Android. The problem is exacerbated by the many different manufacturers of Android devices, who do not always design phones for the newest version of Android due to the resource cost (Verge Staff).

Until the fragmentation issue is resolved, it remains a challenge for developers. To deal with this issue, Google introduced a support library that aids developers by creating a unified API that allows early versions of Android to run many of the features that were introduced in later versions of Android. This API helped make developing for the many versions of Android easier, but did not entirely solve the problem.

Google Inc. has attempted to eliminate the fragmentation issue with its latest release of Android 4.4 “Kitkat” by improving memory management so that the operating system can run on older devices. Google hopes that this will also allow manufacturers to update existing devices and have new devices to ship with the newest version of Android instead of resorting to older versions (Verge Staff).

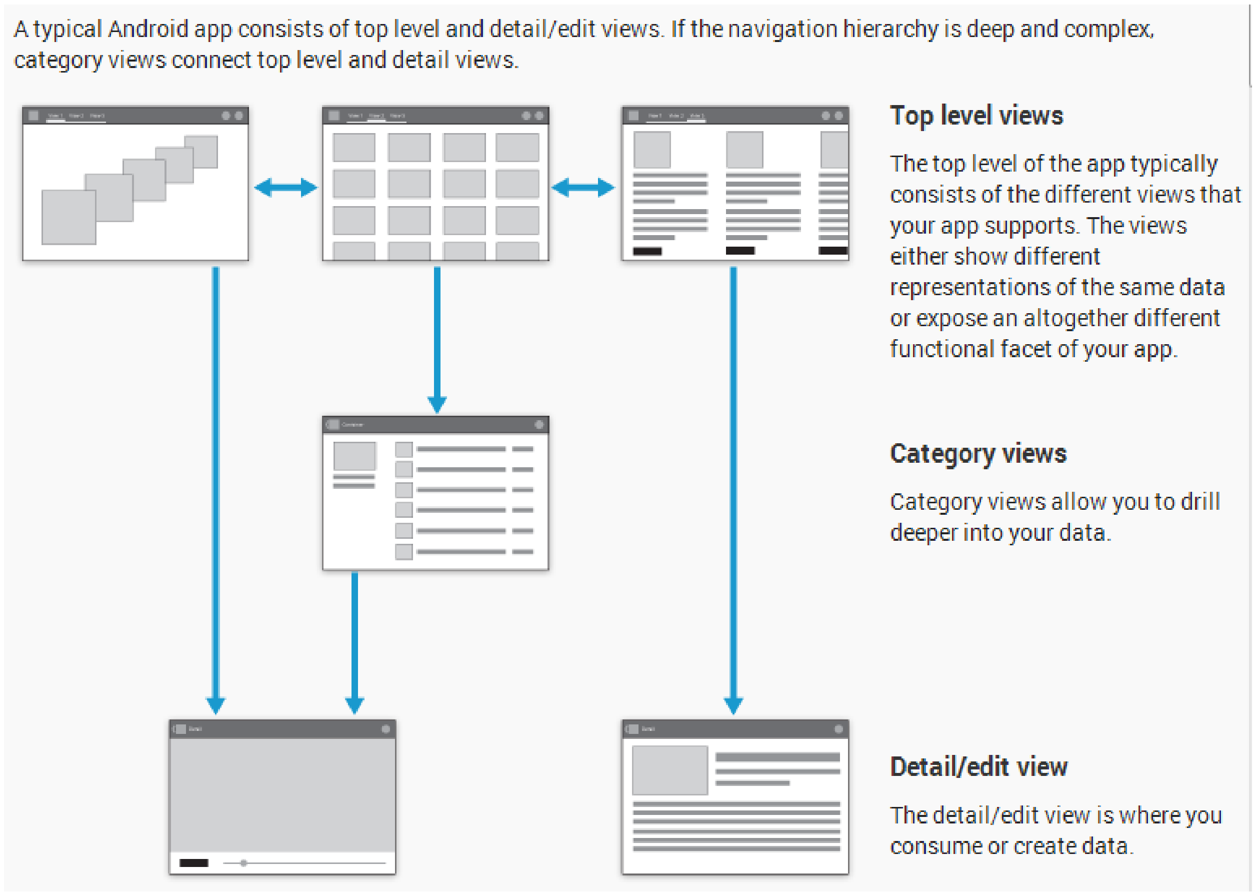
As of March 2014, only nineteen percent of Android devices still used a version of Android prior to 4.0 “Ice Cream Sandwich,” and that number is declining (Ruddock). With the number of users on older versions of Android declining, in the near future the support library will likely no longer be necessary, and developers will be able to write code for Android 4.x without needing to worry about version compatibility affecting their users.

2.3 Android Design Guidelines

Google offers documentation on good User Interface (UI) design. While designing our Calendar app and deciding on a default UI design for our Android framework, we reviewed the high level Android design patterns that Google has put forth, and we provide a summary of these guidelines in our documentation for future students to follow (Google Inc.).

2.3.1 General Structure

Apps should have a hierarchical structure. Users can navigate laterally among the top level views before they select content. Detail views are used for displaying and editing content, and category views may be added for particularly complex apps (Google Inc.). Figure 1 is a visualization of proper hierarchical app structure.


A typical Android app consists of top level and detail/edit views. If the navigation hierarchy is deep and complex, category views connect top level and detail views.

Top level views

The top level of the app typically consists of the different views that your app supports. The views either show different representations of the same data or expose an altogether different functional facet of your app.

Category views

Category views allow you to drill deeper into your data.

Detail/edit view

The detail/edit view is where you consume or create data.

Figure 2 – General structure of Android apps (Google Inc.)

The start screen should avoid being strictly navigational, and should instead focus on content display. This helps users to become engaged right away and makes the app interesting for both new and repeat users.

Every app should have an action bar on each screen. Action bars display the name or icon of the app and provide some consistency within the app. The action bar can also contain view controls for navigation, and a search bar for cutting through the hierarchy.

2.3.2 Top Level Navigation

Users should always have some form of top level navigation available to them. This allows users to return to the start screen easily or navigate to other top level views quickly and efficiently. Android provides three top level navigation view controls: fixed tabs, spinners, and navigation drawers. Using multiple top level view controls can confuse users, so only one top level view control should be used in each app (Google Inc.).

Fixed Tabs

Android provides fixed tabs as a top level hierarchy navigation technique for apps that expect users to frequently be switching between views. Tabs should always allow users to navigate between the views by swiping left or right on the content area. Tabs should be used if the app has a limited number of top level views, and the users will be switching between views frequently. Figure 3 shows an example of what fixed tabs could look like in an app.

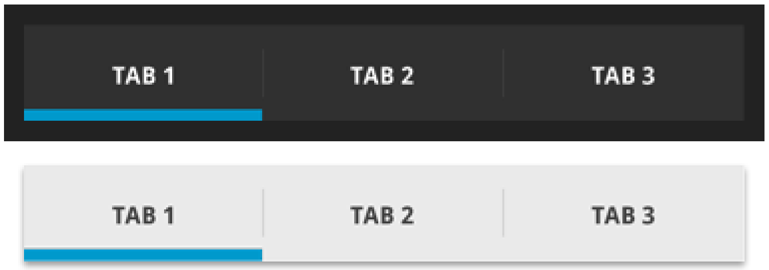


Figure 3 – Fixed tabs (Google Inc.)

Spinners

A spinner is a drop down menu attached to the action bar of an app that allows users to switch between views. Spinners should be used in apps that switch between views of the same data set or data sets for the same type. Spinners can also be used instead of tabs if the designer does not wish to give up screen real estate for a tab bar. An example spinner is shown in the figure below, Figure 4.

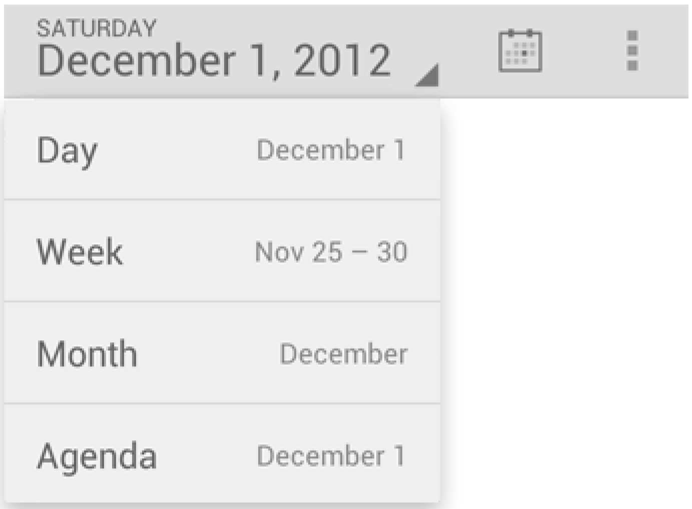


Figure 4 – Action bar spinner

Navigation Drawers

A Navigation drawer is a slide out menu that is accessible at any time and allows users to switch between views. Navigation drawers have space for many items, so they can contain shortcuts to top level views as well as some lower level screens, so they are ideal for complex apps. Navigation drawers may also be used to allow quick navigation between low level views that would not otherwise be related. Figure 5 shows an example navigation drawer that slides out when the user swipes the screen.

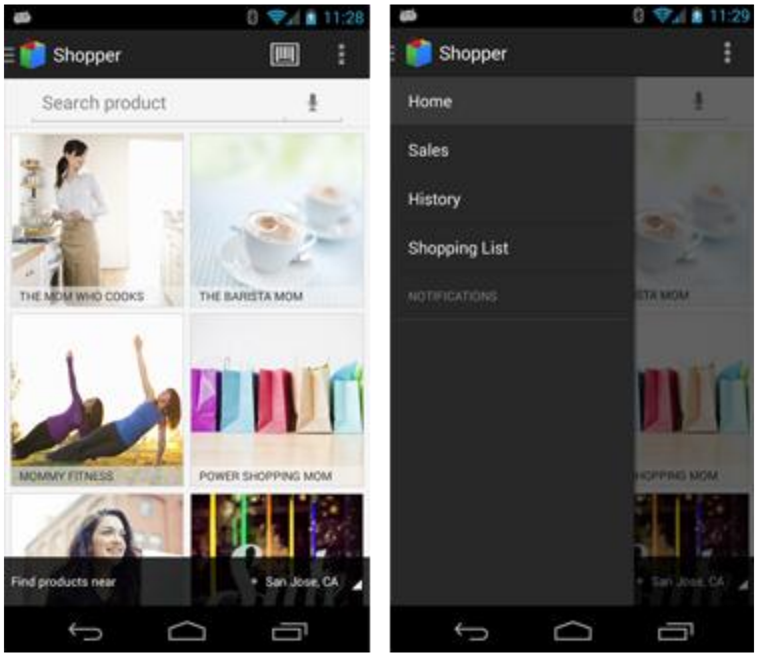


Figure 5 - Navigation drawer

2.3.3 Horizontal Navigation and Shortcuts

Speed is important. Users should not have to traverse the entire hierarchy to get to the view that they want, so apps should provide shortcuts to allow for faster navigation. One technique for faster navigation is to use dropdowns that provide quick actions or shortcuts to detail views (Google Inc.). Figure 6 shows one possible implementation of dropdowns as a shortcut technique.

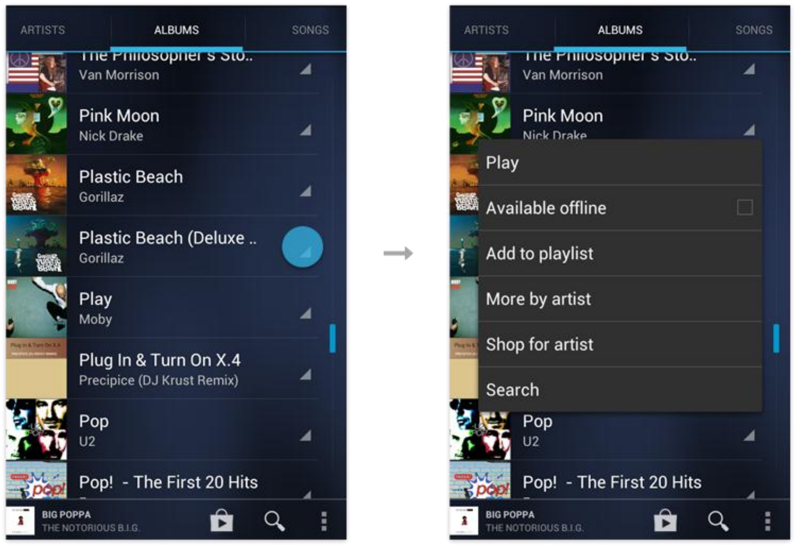


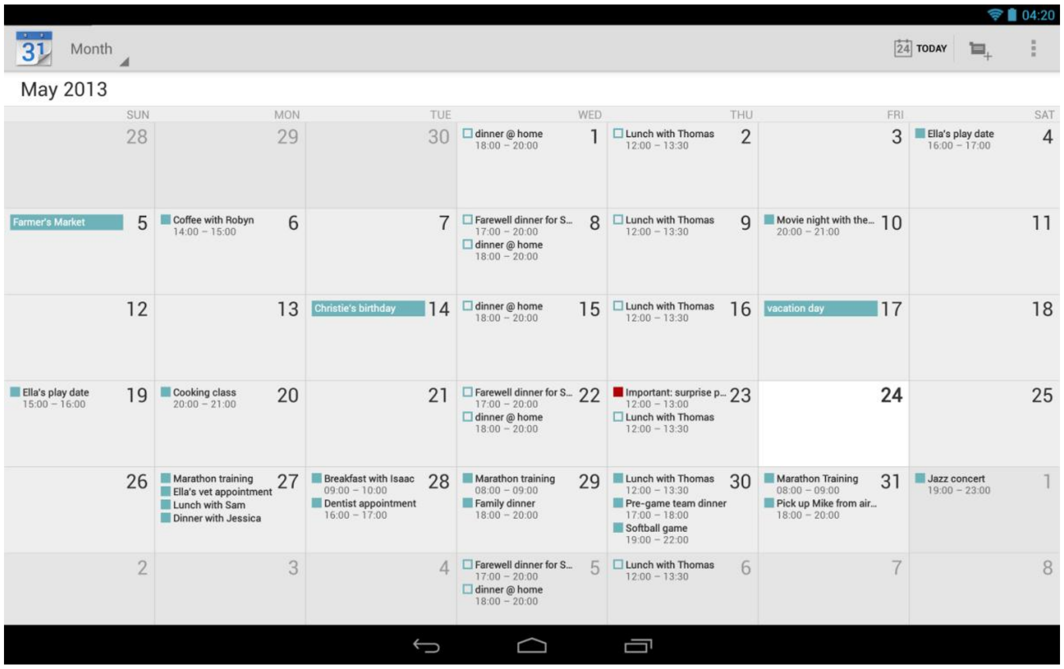
Figure 6 - Dropdown view shortcuts

2.4 Mobile Calendar App Designs

There are many mobile calendar apps available, with a range of different features and designs. While planning our calendar app, we examined three mobile calendar apps so that we could make an informed decision about how best to structure our calendar app.

2.4.1 Google Calendar

Google Calendar is the default app installed on most Android devices. It maintains the basic functionality of a date & time calendar, with extra functionality of adding, editing, and deleting events. Additionally, the Google Calendar is capable of importing and viewing multiple calendar sources at the same time (such as multiple accounts, including non-Google calendars). The calendar also includes the ability to send an email invite from within the app itself. Figure 7 shows the view as soon as the app is opened on a tablet.



May 2013

SUN	MON	TUE	WED	THU	FRI	SAT
28	29	30	dinner @ home 18:00 – 20:00 1	Lunch with Thomas 12:00 – 13:30 2	3	Ella's play date 16:00 – 17:00 4
Farmer's Market 5	Coffee with Robyn 14:00 – 15:00 6	7	Farewell dinner for S... 17:00 – 20:00 dinner @ home 18:00 – 20:00 8	Lunch with Thomas 12:00 – 13:30 9	Movie night with the... 20:00 – 21:00 10	11
12	13	Christie's birthday 14	dinner @ home 18:00 – 20:00 15	Lunch with Thomas 12:00 – 13:30 16	vacation day 17	18
Ella's play date 15:00 – 16:00 19	Cooking class 20:00 – 21:00 20	21	Farewell dinner for S... 17:00 – 20:00 dinner @ home 18:00 – 20:00 22	Important: surprise p... 12:00 – 13:00 Lunch with Thomas 12:00 – 13:30 23	24	25
26	Marathon training Ella's vet appointment Lunch with Sam Dinner with Jessica 27	Breakfast with Isaac 09:00 – 10:00 Dentist appointment 16:00 – 17:00 28	Marathon training 08:00 – 09:00 Family dinner 18:00 – 20:00 29	Lunch with Thomas 12:00 – 13:30 Pre-game team dinner 17:00 – 18:00 Softball game 19:00 – 22:00 30	Marathon Training 08:00 – 09:00 Pick up Mike from air... 18:00 – 20:00 31	Jazz concert 19:00 – 23:00 1
2	3	4	Farewell dinner for S... 17:00 – 20:00 dinner @ home 18:00 – 20:00 5	Lunch with Thomas 12:00 – 13:30 6	7	8

Figure 7 - Main view of Google Calendar, tablet

The app UI follows the design of many of the default apps from Google installed in Android devices. The calendar automatically uses the user's Google account as the base calendar, but can synchronize with other accounts simultaneously (Google, Inc.).

2.4.2 IOS Calendar

One of the largest competitors of Android is the Apple iOS, which also ships with its own calendar app. In this section, we describe the features of the calendar the shipped with iOS 6, not the newly redesigned app that was released with iOS 7. Apple's calendar app is a full featured app that supports multiple calendars, and syncs with services like iCloud, Microsoft® Exchange, Google, and Yahoo!®. The main screen displays the current month, with the ability to switch between a list view of events, a day view, and the month view (Figure 8). At the top of the screen, the current calendar's name is displayed, as well as a button that let the user choose which calendars to view, and a button for adding events. At the bottom is a list of events for the selected day and options to switch the view. Also at the bottom are buttons to highlight the current day, and view event invitations.

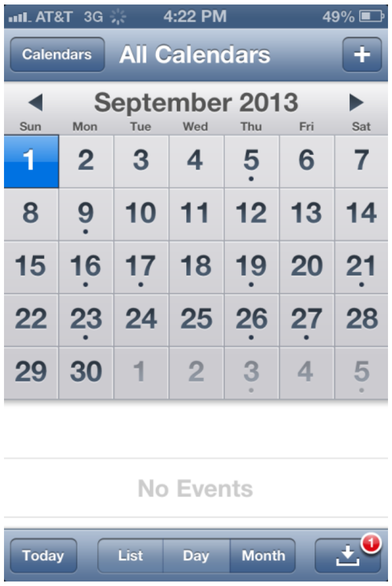
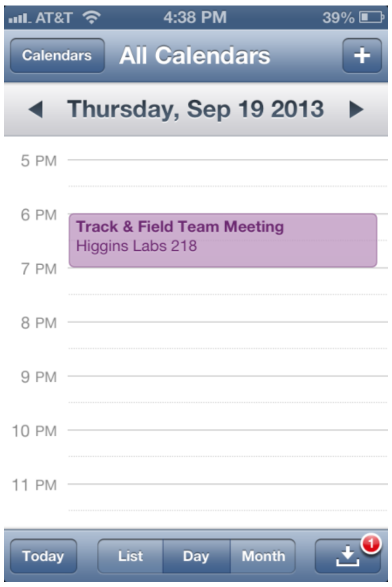


Figure 8 - The Month View for the iOS Calendar

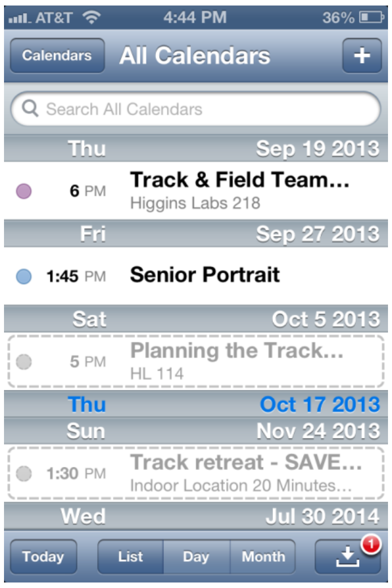
In the month view, the current day is highlighted by default, and events are represented as a dot underneath the day on which they occur. At the top of the view, the selected month is shown, with the ability to cycle through months using arrows. In the middle is the calendar for the selected month, with events listed for a selected day underneath that. The current day is highlighted by default, but users can switch days by tapping on the date. The day view shows all events for a selected day in chronological order. Days can be switched using arrows at the top of the screen, or by swiping left or right (Figure 9).



Track & Field Team Meeting
Higgins Labs 218

Figure 9- The Day View for the iOS Calendar

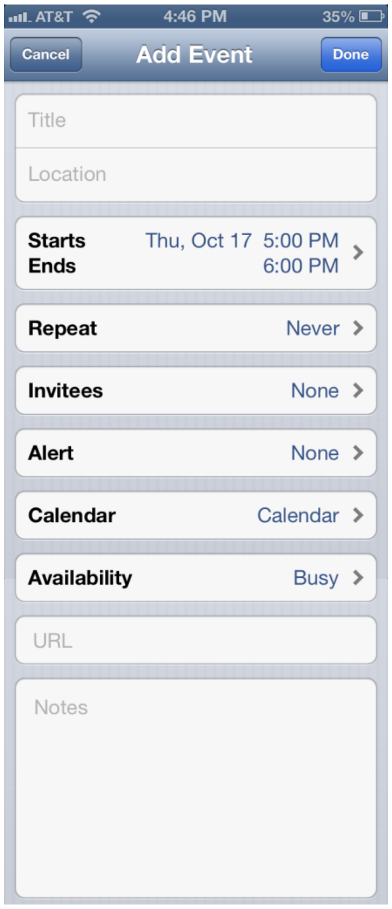
In the list view, users can see a list of all events in the calendar. Above each event is the date, with multiple events being grouped together if they occur on the same day. The calendar color, time, name, and location are shown for each event, and invitations are shown in a grey color (Figure 10).



Thu Sep 19 2013
6 PM Track & Field Team...
Higgins Labs 218
Fri Sep 27 2013
1:45 PM Senior Portrait
Sat Oct 5 2013
5 PM Planning the Track...
HL 114
Thu Oct 17 2013
Sun Nov 24 2013
1:30 PM Track retreat - SAVE...
Indoor Location 20 Minutes...
Wed Jul 30 2014

Figure 10 - The List View for the iOS Calendar

When adding a new event, users can add the following information: title, location, time, URL, and notes. Users can also choose to have the event repeat, send invitations, set an alert, set the calendar, and set their availability (Figure 11).

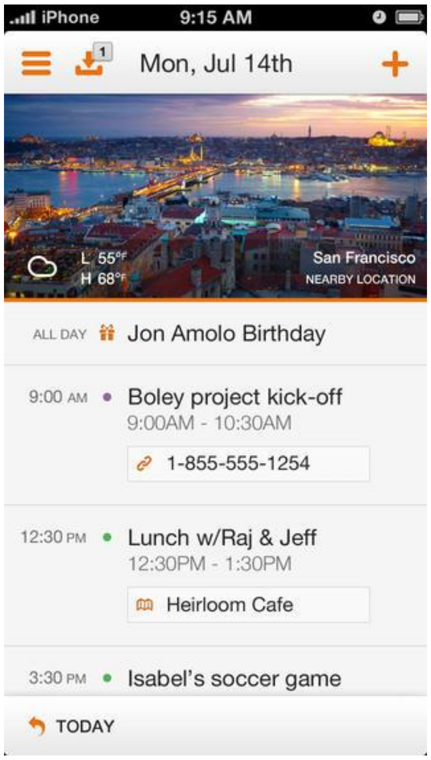


Starts Thu, Oct 17 5:00 PM
Ends 6:00 PM
Repeat Never
Invitees None
Alert None
Calendar Calendar
Availability Busy

Figure 11 - The Add Event View for the iOS Calendar

2.4.3 Tempo Smart Calendar

The tempo smart calendar is a free calendar app for iPhones. The home screen of the app displays an attractive image of a nearby location to catch the eye (Figure 12). Below that, all of the events for the day are listed by time. Users can select an event to view more details about the event. The action bar at the top of the screen displays the current date and gives quick access to the add event page. Other event views are accessible through the menu list.



ALL DAY Jon Amolo Birthday

9:00 AM Boley project kick-off
9:00AM - 10:30AM
1-855-555-1254

12:30 PM Lunch w/Raj & Jeff
12:30PM - 1:30PM
Heirloom Cafe

3:30 PM Isabel's soccer game

Figure 12 - Home screen and day view of Tempo Smart Calendar

The month view of the calendar displays days that have events with a small dot under the day (Figure 13). Users can scroll through months, select days and view the events for that day in a scrolling section beneath the calendar view. Similar to the day view, the current date is displayed on the action bar, and the new event page is easily accessible. Consistent with the Android design guidelines, the action bar stays consistent between the month and the day views to help users stay oriented.

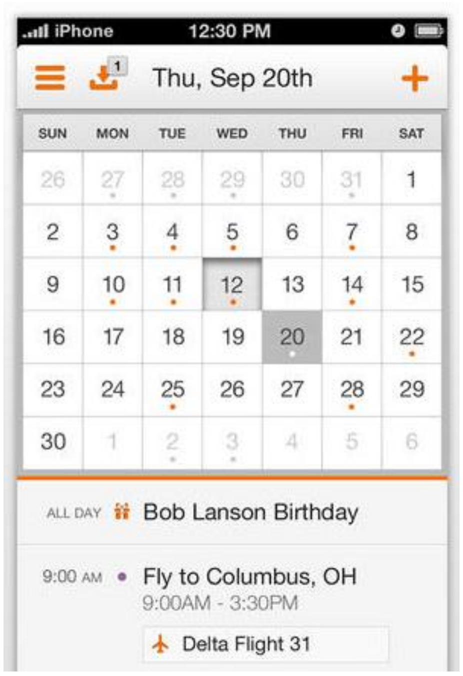
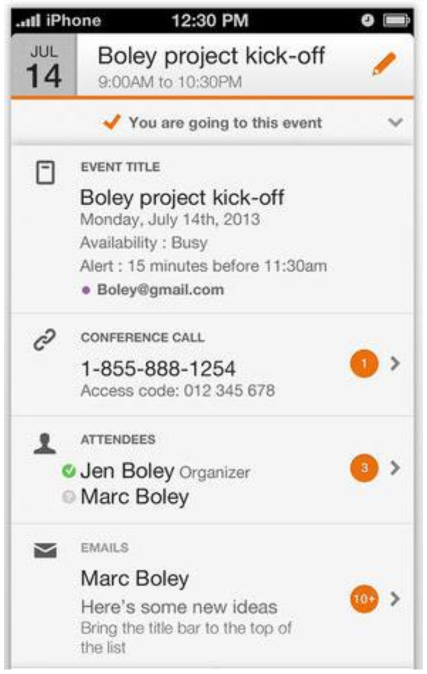


Figure 13 - Tempo Smart Calendar Month View

The event detail view shows all the details about the event: title, description, time, alert settings, attendees, and any other relevant information that Tempo finds in the user's email history or social networks (Figure 14). The action bar is different on this page because this view is a detail view instead of a top level view like the month or day views. Instead of displaying the current date, the action bar displays the date of the event, title and time of the event. The add event button in the top right corner has been replaced with an edit button that allows the user to change any of the information about the event. The user would need to navigate back to a top level view before they can access the create new event option.



Boley project kick-off
9:00AM to 10:30PM

You are going to this event

EVENT TITLE
Boley project kick-off
Monday, July 14th, 2013
Availability : Busy
Alert : 15 minutes before 11:30am
Boley@gmail.com

CONFERENCE CALL
1-855-888-1254
Access code: 012 345 678

ATTENDEES
Jen Boley Organizer
Marc Boley

EMAILS
Marc Boley
Here's some new ideas
Bring the title bar to the top of the list

Figure 14 - Tempo Smart Calendar Event Detail View

3. Methodology

We organized our project around three main objectives. These objectives defined each phase of the project, and helped us to achieve our goal: to enable future software engineering students to create mobile apps that plug into the existing WPI Suite core server by creating an exemplar module for mobile Android devices and clearly documenting the process. Our three objectives for the project were to:

- Develop an Android app framework to facilitate communication between the core and an Android app.
- Write an exemplar calendar module for Android that implements our Android framework.
- Document the exemplar module and process of creating an Android app to enable future students to create their own apps.

We developed a list of requirements for each objective of our project; the Android framework, the exemplar Android app, and the documentation for future WPI Suite users. Next, we developed the Android framework and the exemplar Android app, and then created tutorials so that our work remains accessible and usable by future students after we have graduated. This chapter of the paper describes each step of our process in detail.

3.1 Requirements

After we had identified each of our objectives, we created a list of requirements for each objective that defined the essential parts of each. These requirements listed what components each aspect of our project required to be successful. This section describes the requirements that we came up with for each objective.

3.1.1 Marvin Requirements

We named our Android framework "Marvin." We developed four specifications for Marvin that, when achieved, constituted success of the first objective of the project. In these objectives, we define 'supported app' as any app built using the Marvin framework.

- Marvin sets up a session with the core for the supported app to use.
- Marvin provides a simplified interface for the supported app to communicate with the core server.
- Marvin does not constrain the user interface of apps that it supports.
- Marvin stores session information so that it may be accessed by the supported app.

These specifications were defined in stages. At the beginning of the project, we drafted our original specifications, but as we developed our exemplar app that was supported by Marvin, we revised the specifications for Marvin based on new insights.

We originally intended to have Marvin lay guidelines for the user interface of the app that it was supporting, forcing developers to follow a set user interface design. This would ensure that developers followed Android design guidelines. However, we quickly realized that that was unnecessarily constraining and could limit the usefulness of the framework. On an Android device, screen space is limited, so we decided to allow development teams to design their own interfaces entirely from scratch. This allows for maximum flexibility in apps that are created using Marvin.

We also did not originally plan for Marvin to store session information for the users. However, while developing our exemplar app, we realized that it was impossible for the developers to identify who the current user was based solely on the cookie information from the session, so we modified our requirements for Marvin so that Marvin saves the current username and project so that the supported app can access that information.

3.1.2 Calendar Requirements

We made a Calendar app to provide a concrete example of how to use our framework. The Calendar app was also intended to demonstrate some of the commonly used features in Android apps to allow future software engineering teams to see how they could make use of those features in their own apps.

Before beginning development of our calendar module, we created ten user stories so that we had a clear idea about what features we aimed to have in the app by the end of the project. The user stories that we developed may be found in Appendix A: User Stories for the Android Calendar. When creating these user stories, we referenced other Calendar apps such as the Android Calendar, iOS Calendar, and the Tempo Smart Calendar for inspiration in which features we should add to the app.

3.1.3 Documentation Requirements

One of the most important aspects of our project was the documentation. The documentation that we provided needed to guide students through the process of setting up their development environment, setting up their app to work with our Marvin framework, and getting started on developing for Android. In addition, the code needed to be well documented so that it would be clear and easy to follow when students use it as a reference for developing their own Android apps. The majority of the students using this framework will not have had prior experience developing with Android, so all of this documentation had to be clear enough that a new Android developer would be able to understand it and feel comfortable developing his or her own Android app after reading our documentation.

3.2 Development

Our development process was divided into two steps: development of Marvin, the Android framework, and development of the Android Calendar, the exemplar app.

3.2.1 Marvin Development

We developed Marvin using Eclipse with the Android development tools provided by Google. These tools allowed us to build and test our application on an emulated Android device, without requiring us to install it on a physical device each time.

We kept our code in a GitHub repository that was branched off of the main WPI Suite repository, so that it could be merged back into the main repository when we had finished our project. This code is open source, and available here: https://github.com/lalezaris/wpi-suite.

3.2.2 Calendar Development

Because Marvin is intended to be a common starting point for future development of apps for WPI Suite, we used it as our starting point for our calendar app. We were able to identify issues that students would likely encounter when developing off of our framework on their own, and were able to return to Marvin to revise and improve the framework.

Throughout the development of our calendar app, we focused on adding common features students would likely need in their own apps to provide examples of how those features could work. We attempted to make those features adequately abstracted so that they would be easy for students to emulate in their own apps.

3.3 Accessibility for Students

As the purpose behind this project was to allow the Software Engineering class to introduce Android development, it is important to make the project easily understandable for students new to Android development. To make the code as understandable as possible, we commented our code extensively to make it understandable by students. We also created tutorials and linked to Android developer documentation discussing key Android features that the students might want to make use of in their apps. The code that we developed is intended to be a concrete example of how to implement some of the features that we introduce in the tutorials. Additionally, because setting up the WPI Suite development environment is a long, complicated process, we also created a virtual machine that contains the development environment set up and ready to go.

3.3.1 Tutorials

Many of the Android developer tutorials cover proper use and implementation of common Android features, so we developed tutorials that relate to aspects of Android development unique to WPI Suite and Marvin. These involved tutorials on how to set up the Android development environment, how to create an app with Marvin, how to communicate with the core, and how to install an apk file on an Android device. Additionally, we expanded the tutorials on the wiki from previous years by creating a tutorial about the advancedGet method, which allows users to make more complex queries to the core.

We also created a set of wiki pages to help get new Android developers started. These pages describe some of the common features that can be implemented in Android apps and provide links to various pages on the Android developer pages. These pages are intended to be a starting point for new developers so that they can see what sorts of features they have available to them. They will then be able to explore the developer pages to find other features that they could implement in their apps.

3.3.2 Virtual Machine

To make the setup process as simple as possible, we created a virtual machine containing a Linux based development environment for Android and WPI Suite. This virtual machine can be downloaded by the students, and is ready to run and begin development. This way, students can choose to either set up the environment on their own machine or just download the environment ready to run as a virtual machine.

4. Results

The goal of this project was to allow software engineering students to create mobile apps that work with the existing WPI Suite architecture. We accomplished this by completing our three objectives for the project: to create an Android app framework, to create an exemplar Android app that implements that framework, and to document the process for future software engineering students. The following sections discuss the products we created when accomplishing each objective.

4.1 Android App Framework

The first objective of our project was to create an Android app framework. The purpose of the framework is to enable students to begin developing an Android App that interfaces with the WPI Suite core, without needing to worry about setting up a connection or handling network requests.

We accomplished our first objective by creating an Android framework that we named “Marvin.” Marvin was modeled after the “Janeway” desktop framework created by a previous year’s MQP, “WPI Suite Exemplar Module” (Casola, Hurle and Page).

Marvin simplifies the process of connecting to the WPI Suite server. When an app that uses Marvin is launched, Marvin handles logging in to the core before the main activity of that app is started. This enables developers to send requests for data to the core server as soon as the main activity of their app starts, without needing to worry about logging the user in first.

Marvin also includes the “Network” project that was developed by the “WPI Suite Exemplar Module” MQP, which greatly simplifies the process of sending network requests to the core. The Network project allows developers to create, observe, and send HTTP requests with no background knowledge of network protocols.

Additionally, Marvin stores some session information so that developers can access them from within their app. Marvin stores the username and project of the current session. This information can be used to construct advanced queries to the core, allowing requests to filter data models based on current user.

Marvin provides a single view for users to login to the WPI Suite core. A screenshot of the Marvin login screen is shown in figure Figure 15 – The Marvin login screen, with custom styles. This view prompts users to input their login information, handles the login process, and then starts the main activity of the app with a network connection already set up.

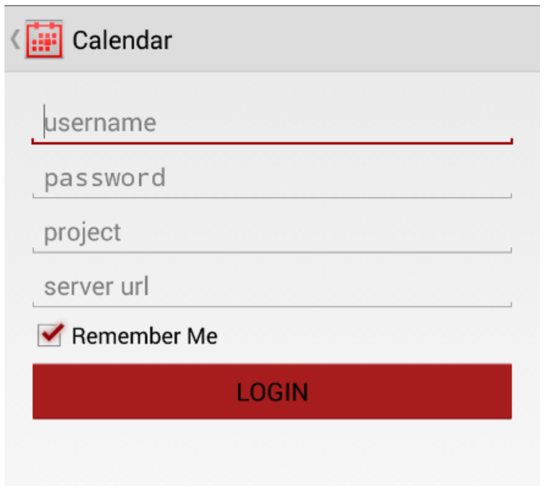


Figure 15 – The Marvin login screen, with custom styles.

Marvin does not constrain the UI of any Android app developed using it. Once Marvin has handled logging in, developers can choose any layout for their app, and Marvin does not require them to implement any features.

4.2 Exemplar Calendar Module

Our second objective was to create an exemplar app for Android that implements our Android framework. The purpose of this exemplar app was to demonstrate how our framework could be used to create an app. We wanted this app to be used as a reference for future software engineering students, so we emphasized good code structure and documentation. Additionally, we wanted to include examples of different features that we thought software engineering students might want to implement in their apps, so students could reference our project when implementing those features.

We created a calendar app to serve as our exemplar app. The calendar app includes features such as creating, editing, and deleting events, and viewing events from a month view, week view, day view, or list view. In addition to basic functionality, our app provides examples of how an app interfaces with Marvin, how to send advanced queries that take multiple arguments to the core server, and how to use a number of Android features. Students will be able to use this project as an example for how to implement these features in their own apps.

Our app demonstrates a number of Android features including setting up and adding items to an action bar, changing activities and sending extras between activities, using fragments and popups, providing alert and toast notifications, selecting and deleting items from a list, and changing styles. These features will likely be useful in any Android app that students will create, so our exemplar app provides a concrete example of how those features may be implemented.

4.3 Documentation

Our third objective to complete this project successfully was to document the process of creating an app for Android that uses our framework. We needed to ensure that students would be able to use our work to create apps. To document the process of using our project, we added instructional wiki pages to the WPI Suite GitHub repository. Those pages may be found at https://github.com/lalezaris/wpi-suite/wiki.

The wiki contains pages on how to set up a development environment to work with Android, how to create a project that uses Marvin, and how to connect to your server once you have created the app. Additionally, the wiki contains some background information about developing with Android, and highlights some of the features that we found most helpful when developing our calendar app. We have included links to Android’s documentation within our wiki so that students will have easy access to the most up to date information on the features that we have highlighted.

5. Future Work

Over the course of the term we have identified some weaknesses and limitations of Marvin and of our exemplar Android calendar. Later improvements could improve and add to Marvin functionality and improve the usability of our calendar.

5.1 Marvin Improvements

Our goal with Marvin was to create a simple Android framework for WPI Suite that would unify WPI Suite's Android apps while also making it easier for students to start developing for WPI-Suite on the mobile platform. While Marvin contains the core functionality of logging in and storing a user's credentials, we have identified some possible improvements. These include adding a settings page to Marvin, and improving login security.

Adding a settings page to Marvin would make it easier for app users to perform simple tasks, like changing a password, creating projects, and modifying user roles. The settings page would perform the same functions as the "WPI Suite Admin Console," but from within the mobile apps. Future development teams could add their own app settings, without the need to start from scratch. Furthermore, development teams could use the settings page as an example on how to set up their own set of configuration options.

Currently, when a user chooses to login automatically, their login information is stored as plain-text to their device. The information needs to be encrypted to better protect a user's information. Should the device become compromised, it would be very easy for an attacker to steal login information. Encrypting it would make it more difficult for attackers to gain unauthorized access to the system.

We also would like to implement a universal logout function that is incorporated into Marvin. During this project, we did not accomplish that, because we would have needed to construct a menu for the app that contains a logout option, and we did not wish to constrain the UI in that way. However, with more thought and design, we could provide the logout feature in such a way that it does not constrain the UI of the app. Instead, we provided an example of how to implement a logout function in our Calendar app. Details on how to implement such a logout feature are provided on the "Using Marvin to Connect to a WPI Suite Server" page of our wiki.

5.2 Calendar Improvements

We identified two potential improvements to the app that we developed that would enhance a user's experience, but these would not significantly add to the main purpose of the app: an exemplar module for students to reference during their own mobile development. These improvements include searchable events and recurring events.

Searchable events would allow users to search for events by name, creator, date and time, or location. This would allow for quicker and easier access to specific events, without the need to find the date within the month, week, or day views. There are examples of how to do searching within the WPI-Suite code base already, in the requirements manager module, should students need an example of how to do searches of the database. Additionally, we created some detailed tutorials on how to use WPI Suite's advanced API for querying items from the database. The advanced API could handle searching quickly and easily.

Recurring events would allow users to set events to repeat on certain days of the week for either a fixed period of time, or until the user cancels the recurrence. This would make it easier to set up events that occur frequently, like meetings. We would like users to have the option of setting an event to recur daily, weekly, or monthly, and be able to select the number of times an event repeats, by either setting a fixed number of times to recur, or by setting an end date to the recurrence.

Bibliography

Beavis, Gareth. *A Complete History of Android: Everything You Need to Know About Google's Mobile Operating System*. 23 September 2008. 6 October 2013. <http://www.techradar.com/us/news/phone-and-communications/mobile-phones/a-complete-history-of-android-470327>.

Casola, Christopher, Andrew Hurle and Christopher Page. *WPI Suite Exemplar Module*. MQP. Worcester: Worcester Polytechnic Institute, 2013. Online Database.

Deleon. *A Brief History of Android*. 14 September 2013. 6 October 2013. <http://technoblimp.com/2013/09/14/a-brief-history-of-android>.

Developer Nation. "Developer Economics 2013: The Tools Report." 2013. <http://www.visionmobile.com/product/developer-economics-2013-the-tools-report/?utm_source=DE13Distimo&utm_medium=link&utm_campaign=doritos_report_13>.

Google Inc. *Application Structure*. n.d. 6 October 2013. <http://developer.android.com/design/patterns/app-structure.html#general-structure>.

Google, Inc. *Build.VERSION_CODES*. 3 October 2013. 6 October 2013. <http://developer.android.com/reference/android/os/Build.VERSION_CODES.html>.

—. *Google Calendar*. 1 August 2013. App Store. 12 October 2013. <https://play.google.com/store/apps/details?id=com.google.android.calendar&hl=en>.

Open Handset Alliance. *Industry Leaders Announce Open Platform for Mobile Devices*. 5 November 2007. 6 October 2013. <http://www.openhandsetalliance.com/press_110507.html>.

Open Signal. "Android Fragmentation Visualized." 2013.

Ruddock, David. *Android Distribution Numbers Updated*. 4 March 2014. Web. 24 March 2014. <http://www.androidpolice.com/2014/03/04/android-platform-distribution-numbers-updated-kitkat-at-2-5-jelly-bean-62-gingerbread-down-to-19/>.

Verge Staff. *Android: A Visual History*. 7 December 2011. 6 October 2013. <http://www.theverge.com/2011/12/7/2585779/android-history>.

—. *Android: A Visual History*. 7 December 2011. 6 October 2013. <http://www.theverge.com/2011/12/7/2585779/android-history>.

Appendix A: User Stories for the Android Calendar

User Story 1: Create Events

Users can create events.

Events have the following attributes:

- Title
- Location
- Start Date/Time
- End Date/Time
- Invitees
- Description
- Event Organizers
- Attendees (Not editable)

User Story 2: Update Events

Users should be able to edit already existing events if they are marked as Event Organizers for that event.

Users can edit:

- Title
- Location
- Start Date/Time
- End Date/Time
- Invitees
- Description

- Event Organizers

User Story 3: Delete Events

Users can delete events if they are marked as Event Organizers for that event.

User Story 4: View Events

Users should be able to view events that they are invited to or marked Event Organizers for.

User Story 5: Invite Users to an Event

Event organizers should be able to invite users to an event.

Users who are invited will receive a notification and can accept or deny.

Users that accept an event invitation will be moved off of the invite list and onto the attendees list.

User Story 6: Users can set alerts for events that they are invited to

Users can be alerted of upcoming events.

Users specify a number of minutes before an event that they want to be notified.

User Story 7: Users can view a list of events

Users can view a list of all events that they are organizing, invited to, or attending.

User Story 8: Users can view events by week

Users can view events through a weekly view that displays events categorized by day.

User Story 9: Users can view events by day

Users can view events through a daily view that displays events categorized by time.

User Story 10: Users can view events by month

Users can view events through a monthly view that displays events categorized by day.

Attributions: