January 2014

Analysis Of Multi-Platform Mobile Application Development

Courtney B. Thaden

Follow this and additional works at: https://commons.und.edu/theses

ANALYSIS OF MULTI-PLATFORM MOBILE APPLICATION DEVELOPMENT

by

Courtney B. Thaden
Bachelor of Science, University of North Dakota, 2010

A Thesis

Submitted to the Graduate Faculty

of the

University of North Dakota

in partial fulfillment of the requirements

for the degree of

Master of Science

Grand Forks, North Dakota
May
2014
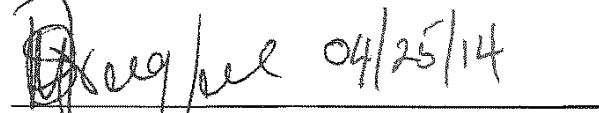
This thesis, submitted by Courtney B. Thaden in partial fulfillment of the requirements for the Degree of Master of Science from the University of North Dakota, has been read by the Faculty Advisory Committee under whom the work has been done, and is hereby approved.

Naima Kaabouch, Chairperson

Arthur Miles

04/25/14

Saleh Farque

This thesis is being submitted by the appointed advisory committee as having met all of the requirements of the School of Graduate Studies at the University of North Dakota and is hereby approved.

Wayne Swisher
Dean of the Graduate School

april 28, 2014
Date

iii

Title   Analysis of Multi-Platform Mobile Application Development

Department   Electrical Engineering

Degree   Master of Science

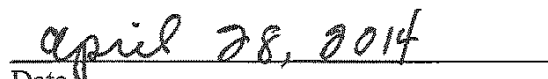   In presenting this thesis in partial fulfillment of the requirements for a graduate degree from the University of North Dakota, I agree that the library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by the professor who supervised my thesis work or, in her absence, by the Chairperson of the department or the dean of the School of Graduate Studies. It is understood that any copying or publication or other use of this thesis or part thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of North Dakota in any scholarly use which may be made of any material in my thesis.

                              Courtney B. Thaden
                              April 25, 2014

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I dedicate this paper to all the strong females in my life, past and present, especially my

mother Betsy, my sister Sarah, and my late grandmother Winnie.

ABSTRACT

The variety of mobile devices and their operating platforms has rapidly increased. With this increase come separate standards, programming languages, and distribution markets. Typically developers want to deliver their products to a variety of users encompassing various platforms; however choosing to develop using a native program for a platform can delay the development and release on another platform. Multi-platform development applications were created in order to deploy applications to various platforms in a more timely and cost efficient manner by using a single code base.

The purpose of this study was to investigate the multi-platform development applications MoSync, Appcelerator, and PhoneGap, create a test application using each multi-platform development application to run on the Android emulator and iOS simulator to determine performance, and also determine which multi-platform application was best suited for allowing a developer to create a mobile application that could be utilized on a variety of platforms.

CHAPTER I

INTRODUCTION

Introduction

Mobile device application development has increased with the rising number of smartphones on the market (Boardman, 2012; Tech Terms, 2014). The variety of smartphone devices is ever expanding, as well as their powerful operating platforms (Charland & Leroux, 2011). Each platform involves separate standards, programming languages, and distribution markets (Corral, Sillitti, & Succi, 2012b). Typically developers want to deliver their products to a variety of users encompassing various platforms; however choosing to develop using a native program for a platform can delay the development and release on another platform. Multi-platform development applications were created in order to deploy applications to various platforms in a more timely and cost efficient manner, with the principle idea of "develop once, deploy everywhere" (Blom, Book, Gruhn, Hrushchak, & Köhler, 2008; Corral et al., 2012b).

Statement of the Problem

Studies have shown that many people are turning to multi-platform applications to develop a mobile application once which can then be deployed on multiple platforms, but what remains to be shown is which multi-platform development application would be best. The multi-platform applications were analyzed to determine ease of use and proper functionality on two target platforms. Determining the answers to these questions may

lead to discovering new capabilities and functionalities that are needed within these applications and may also help developers identify the development application that could be most efficient to use in the creation of applications for multiple platforms.

Purpose of the Study

The purpose of this study was to investigate multi-platform development applications currently on the market used to develop mobile applications. Differences between mobile application platforms have been studied and documented. Each platform was unique and possessed different behaviors, capabilities, and features. What remained to be determined was which multi-platform application would be best suited for allowing a developer to create a mobile application that could be utilized on a variety of platforms.

Significance of the Study

The analysis of multi-platform development applications could provide a developer a better understanding of the differences among multi-platform development applications and may lead to discovering new capabilities and functionalities that are needed within these applications. It also may assist developers in identifying the most efficient development application to use when creating applications for multiple platforms. Analysis and subsequent findings could possibly allow developers to have more time to focus on improving applications rather than spending their time on slow, individual platform development. Findings may also reveal areas where existing multi-platform development applications are lacking, thus allowing for improvements within multi-platform development applications to be created.

Definition of Terms

The following terms are defined to provide meaning and understanding in relation to this study:

Application (Apps): A software program that runs on a computer or mobile device and most commonly referred to as "apps" (Tech Terms, 2014).

Application Programming Interface (API): A set of commands, function, and protocols which programmers can use when building software for a specific operating system. An API allows programmers to use predefined functions to interact with the operating system instead of writing them from scratch (Tech Terms, 2014).

Closed System: Is licensed computer software carrying a copyright in which the source code is not made available to the general public. It is also known as proprietary software or closed source software (Wikipedia, 2014).

Debug: To eliminate software program errors commonly called "bugs" (Tech Terms, 2014).

Developer: A person or organization that designs and writes software and is often referred to as an application developer. The term generally refers to designers and programmers in the commercial software field (PCMag, n.d.).

Event Listener: An interface that is the primary method for handling events within computer software (W3C, 2003).

Extensible Markup Language (XML): A metalanguage that is used to create markup languages for specific applications and is used to define documents with a standard format that can be read by any XML-compatible application (Tech Terms, 2014).

3

Graphical User Interface (GUI): refers to the graphical interface of a computer that allows users to click and drag objects with a mouse instead of entering text at a command line (Tech Terms, 2014).

Hybrid App: An application in which some or all of your UI and business logic is written in HTML, CSS, and JavaScript running within a "native wrapper" such as a Titanium WebView or PhoneGap container. Hybrid apps have limited access to the device hardware, though such access varies by mobile operating system and development framework. Hybrid apps offer app store distribution and operation without a live network connection (Appcelerator, n.d.).

Interface: An interface can refer to either a hardware interface that connects two or more electronic devices together or the means in which a person controls a software application or hardware device (Tech Terms, 2014).

Internet: A communications network consisting of countless networks and computers that allow people to share information (Tech Terms, 2014).

Model-View-Controller (MVC): A software pattern that divides a given software application into three interconnected parts for implementing user interfaces (Wikipedia, 2014).

Multi-Platform Application: An application which is developed for multiple operating systems or platforms. Typically some or all of the user interface and logic is written in HTML, CSS, and JavaScript running within a "native wrapper." These applications have limited access to the device hardware, though such access varies by mobile operating system and development framework. Sometimes multi-platform applications are also called hybrid applications (Appcelerator, n.d.; Tech Terms, 2014).

Native Application: An application that runs directly on a mobile device and has access to the hardware features of that device. Typically these applications can be run without a live network connection (Appcelerator, n.d.).

Open System: Licensed computer software in which the copyright holder makes the source code available to the public and provides the rights to study, change, and distribute the software to anyone and for any purpose. Also known as open software standard or open standard (Wikipedia, 2014).

Operating System (OS): An operating system "OS" is software that communicates with the hardware and allows other programs to run (Tech Terms, 2014).

Platform: A computer's operating system that allows the running of certain software. Platform examples include Windows and MacIntosh operating systems (Tech Terms, 2014).

Portable Operating System Interface (POSIX): This refers to a family of standards specified by the IEEE for maintaining compatibility between operating systems. POSIX defines the application programming interface (API), along with command line shells and utility interfaces, for software compatibility with variants of Unix and other operating systems (Wikipedia, 2014).

Smartphone: A smartphone is a mobile phone that includes advanced functionality beyond making phone calls and sending text messages and may be capable of running third party applications (Tech Terms, 2014).

SMS: "Short Message Service." SMS is used to send text messages, typically up to 160 characters in length, to mobile phones (Tech Terms, 2014).

Software Development Kit (SDK): A collection of software used for developing applications for a specific device or operating system (Tech Terms, 2014).

Tablet: A portable computer that uses a touchscreen as its primary input device (Tech Terms, 2014).

Web App: A mobile-ready web page accessed from a desktop or mobile browser, and typically formatted specifically to address the screen sizes and user interaction expectations of a mobile device. Web apps excel at platform reach, a "no-download" installation process, and instant application updates for all users. Web apps typically require a constant network connection (Appcelerator, n.d.).

World Wide Web Consortium (W3C): An international community that develops open standards to ensure the long-term growth of the Web (W3C, 2012).

Organization of the Study

This study has been organized in five chapters. Chapter I provides an introduction to the study, statement of the problem, the purpose of the study, significance of the study, and definitions of terms. Chapter II provides a literature review regarding the development of mobile applications, mobile development platforms, mobile platform languages, and multi-platform application development. Chapter III provides the methodology and design of the study. Chapter IV provides the results of this study, while Chapter V provides a conclusion and discussion.

CHAPTER II

REVIEW OF THE LITERATURE

Introduction

Mobile devices use a variety of powerful operating systems or platforms, each of which involves separate standards, programming languages, and distribution markets (Corral et al., 2012b). Typically developers want to deliver their products to a variety of users using various platforms, but choosing to develop for one platform can delay development and release on another platform (Corral, Janes, & Remencius, 2012a). It can also be very expensive to develop native applications for each platform as there are numerous platforms (Corral et al., 2012a). Developers are tasked with having to make the tough decision of which platform to develop for first, on their list of targeted platforms (Corral et al., 2012b). These problems, when developing mobile applications, have led to the growth in creation of multi-platform applications (Holzer & Ondrus, 2011).

Developing Mobile Applications

Mobile application development has become very popular among people of varying programming skills (Boardman, 2012). This could be due to the relatively low cost and short time commitment an application takes to cultivate from the conception of an idea to readying it for distribution (Boardman, 2012). Novice developers have many useful resources readily available which allow them to learn the necessary skills while attempting to develop applications. Some of these resources include: online tutorials,

developer forums, and books (Boardman, 2012). Online tutorials and books are offered

for various experience levels ranging from amateur to advanced. Although operating

platforms change quite frequently making it difficult to find a current book containing the

latest version, the changes typically are not drastic enough to make the book obsolete

(Boardman, 2012). Also developer forums should not be overlooked as many questions

that a developer might ask are typically answered on some forum (Boardman, 2012).

According to Computerworld magazine's editor in chief, Scot Finnie (2013), the

following are five tips for developing successful mobile applications that developers

should keep in mind:

1. In order to succeed, a mobile application must solve a problem.

2. Focus on one thing and do it well.

3. If you build it…nope, they probably won't come.

4. Applications need optional user notifications.

5. Don't force users to run your application instead of visiting the corporate

website. (p. 40).

In regards to tip number one, a mobile application must offer a useful benefit to the user

or people will not use it (Finnie, 2013; Wong, Khong, & Chu, 2012). The mobile

application could be designed to solve a variety of problems, including saving time or

money, entertaining or enlightening, delivering important functionality, or offering a

novel service (Finnie, 2013). Tip number two, Finnie (2013) believes to be the most

important recommendation. It is better to do one thing very well, than to do multiple

things mediocre because going feature crazy could wind up derailing a project

(Finnie, 2013). Tip number three simply means that although a designer may have his or

her application in a store available for download, the store is not a direct channel to everyone and does not guarantee that people will want it, need it, or use it (Finnie, 2013). Tip number four reminds developers that notifications are not appropriate for all applications, so use them only as needed (Finnie, 2013). The final tip, number five, expresses the idea that a mobile application should concentrate on improving the user experience and utility of the mobile version of the website rather than replace the corporate website altogether (Finnie, 2013).

Mobile Development Platforms

Apple's iOS and Google's Android™ mobile platforms have been the two front runners in the mobile market in the past few years, but the two are very different platforms (Emmanouilidis, Koutsiamanis, & Tasidou, 2013; Sharma, 2011).

Android

Android, an Apache-free software platform for mobile devices based on Linux, was launched by Google in 2007 to advance open standards for mobile devices (Gavalas & Economou, 2011). The openness of Android allows the analysis and understanding of code which can lead to better feature comprehension, bug fixes, further improvements regarding new functionalities, and the ability to port to new hardware (Gandhewar & Sheikh, 2011). An open source software allows for customization to suit specific needs in different ways, but also allows for collaboration between developers (Proffitt, 2011).

The Android software stack includes an operating system, middleware, and key applications. To break it down even further, the Android Architecture, shown in Figure 1, contains four distinct layers; Linux Kernel, Libraries, Application Framework, and Applications (Gandhewar & Sheikh, 2011).

9
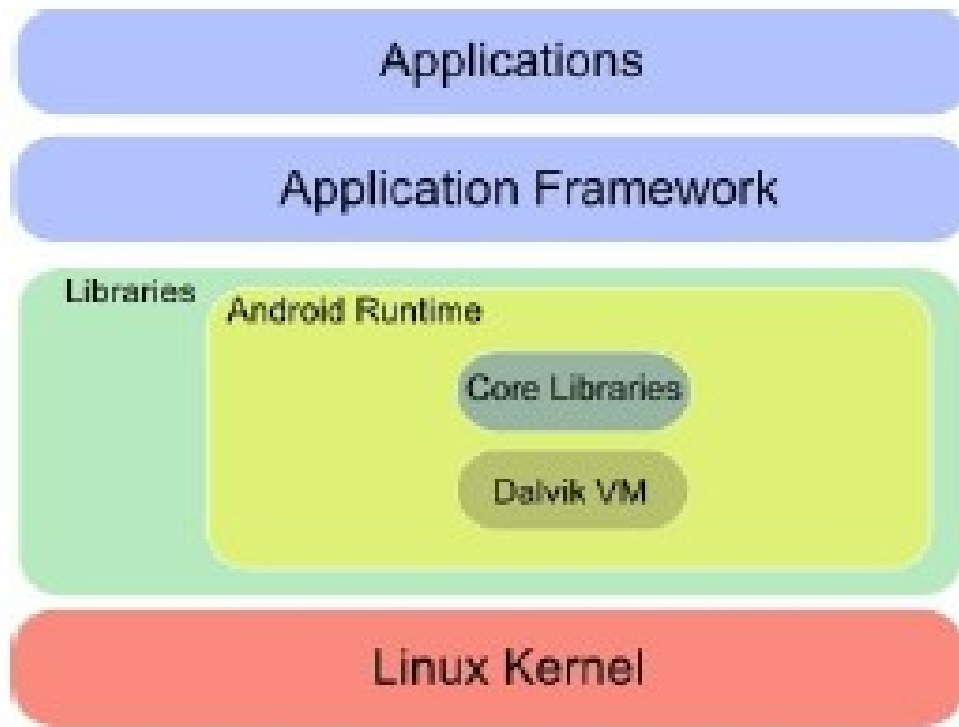
Figure 1. Android Architecture. (Gandhewar & Sheikh, 2011, p. 13, reprinted with permission).

The Linux Kernel, which was built with Linux version 2.6 operating system (OS), that Android relies on for core system services such as security, memory management, process management, network stack, and driver model acts as an abstraction layer between the hardware and the rest of the software stack (Gandhewar & Sheikh, 2011). For example, the camera driver is found in the Linux Kernel and allows the user to send commands to the camera hardware (Sharma, 2011).

The layer above the Linux Kernel is the Libraries (Gandhewar & Sheikh, 2011). The Libraries layer entails two parts; C/C++ libraries and the Android Runtime (Gandhewar & Sheikh, 2011). The C/C++ libraries are all written in C and C++ and get called up through a Java interface (Gandhewar & Sheikh, 2011). Examples of C/C++

libraries found in the Libraries layer are the Surface Manager, 2D and 3D graphics, Media Codecs like MPEG-4 and MP3, media frameworks, accelerometers, the SQL database SQLite, and the web browser engine WebKit (Gandhewar & Sheikh, 2011; Sharma, 2011). Within the Android Runtime layer is a set of core Java libraries and the Dalvik Virtual Machine (VM) (Gandhewar & Sheikh, 2011). The core set of Java libraries includes a large subset of the Java Standard Edition (SE) 5.0 library, which allows for reduced migration cost from Java desktop applications (Gavalas & Economou, 2011). The Dalvick VM is a Java byte code interpreter (Sharma, 2011). Previously Java was a slow platform, but Dalvick was optimized for performance on mobile devices (Gandhewar & Sheikh, 2011; Sharma, 2011). Some of these optimizations were for low memory requirements and a register-based VM architecture instead of the typical stack-based architecture (Gandhewar & Sheikh, 2011). Java applications are compiled in the Dalvick executable format (.dex) which are more compact and efficient than class files (Gandhewar & Sheikh, 2011; Gavalas & Economou, 2011). Within the Dalvick VM is the Java VM Tool Interface (JVM TI), which provides functionalities to inspect the state of a VM, gather information during runtime, and control the execution of applications running on the Java VM (Gandhewar & Sheikh, 2011). One advantage Android has with the use of VMs is that each application is run as its own process in its own VM, so no application is dependent upon another (Sharma, 2011). This means that if an application crashes, it should not affect any other application running on the device (Sharma, 2011).

The next layer is a software framework known as the Application Framework layer and includes programs that manage the phone's basic functions (Gandhewar & Sheikh, 2011; Sharma, 2011). This layer implements a standard structure of an

application for a specific operating system (Gandhewar & Sheikh, 2011). The basic

functions of the phone are items such as resource allocation, telephone applications,

switching between processes, and keeping track of the phone's physical location

(Sharma, 2011). Full access of the Application framework is available to developers in

order to allow the creation of applications using the basic functionalities (Sharma, 2011).

The Application layer, which is the upper most layer of the Android software

stack, is where core applications are provided (Gandhewar & Sheikh, 2011). These

applications include basic functions of the device such as email, short message service

(SMS), calendar, maps, browser, and accessing contacts (Gandhewar & Sheikh, 2011;

Sharma, 2011). The Java programming language is what all applications are written in for

Android (Gandhewar & Sheikh, 2011). Any application can use any other application in

order to simplify component reuse, subject to security constraints enforced by the

framework (Gavalas & Economou, 2011).

Anyone developing for Android must understand the programming framework

used. The programming framework for Android, shown in Figure 2, consists of the

Software Development Kit (SDK), the Eclipse Integrated Development Environment

(IDE) and the Java Development Kit (JDK) (Gandhewar & Sheikh, 2011). The Eclipse

IDE must be version 3.2 or later and the JDK must be version 1.6 or later (Gandhewar &

Sheikh, 2011). The JDK must be preinstalled for the installation of the Android SDK and

Eclipse IDE to work (Gandhewar & Sheikh, 2011). A comprehensive set of development

tools including libraries, an emulator, documentation, sample code, a cross assembler,

packaging tool, and debug software are included in the Android SDK (Gandhewar &

Sheikh, 2011). The emulator allows developers to prototype, develop, and test

applications without using a physical device (Gandhewar & Sheikh, 2011). The Android

emulator specifically supports Android Virtual Device (AVD) configurations which

allow the specification of the Android API, the hardware options, and skin files to be

used (Gandhewar & Sheikh, 2011).

Figure 2. Android Programming Framework. (Gandhewar & Sheikh, 2011, p. 14, reprinted with permission).

Developers must pay a one-time registration fee of $25 before publishing their

first application (Sharma, 2011). Android applications can be acquired from any source,

not just the Android Market (Gandhewar & Sheikh, 2011). However, the Android Market

is an open system, so applications do not have to be approved before being available in

the market to consumers (Boardman, 2012). Some say this open system allows for more

creativity and a better chance for an application to make it to the market (Boardman,

2012). Google has, without explanation, removed some Android applications from the

Android Market which led to some rumored talk of creating a store for "banned apps"

(Boardman, 2012, p. 47). The Android platform also takes the standard thirty percent of application revenues (Sharma, 2011).

Android has given device makers flexibility in their hardware choices (Proffitt, 2011). There is not one single smartphone or one single tablet that defines Android unlike Apple with its iPhone and iPad.

iOS

According to Nicholas C. Zakas's 2013 article, "The Evolution of Web Development for Mobile Devices", published in Communications of the ACM, the iPhone opened up the "real" Internet to smartphone users (p. 42). Zakas emphasizes this importance because developers no longer had to write mobile-specific interfaces in custom languages such as Wireless Application Protocol (WAP) (Zakas, 2013).

Apple's default operating system, iOS, was introduced to the market in January, 2007 (iOS, 2014; Sharma, 2011). Originally known as OS X, the name of the operating system was changed to iOS with the introduction of the iPhone 4 in June, 2010 (iOS, 2014). iOS is created using Apple's SDK which includes an IDE and is known as XCode (Dupont, 2012). iOS is derived from Apple's desktop operating system, Mac OS X, and is a Unix like operating system (Sharma, 2011). Apple takes a different approach than Android due to the fact its system is a closed proprietary system with peerless marketing (Sharma, 2011). Objective-C is the language iOS is written in, which is an object oriented version of C that uses messages (Dupont, 2012; Sharma, 2011). Kavita Sharma (2011) described the use of the Objective-C iOS as similar to having every phone call go through an operator who relays the message to the intended receiver rather than just calling the intended receiver directly.

There are four distinct abstraction layers within iOS; the Core OS layer, the Core Services layer, the Media layer, and the Cocoa Touch Layer (Sharma, 2011). Fundamental interfaces such as those used for accessing files, low-level data types, Boujour services, and access to POSIX threads and network sockets are found in the Core OS and Core Services layers (Sharma, 2011). The graphics, audio, video, and animation technologies which are written in a mixture of C-based and Objective-C based interfaces are contained in the Media layer of iOS (Sharma, 2011). The final layer, the Cocoa Touch layer, offers the fundamental infrastructure such as file management, network operations, and support for collections used by applications (Sharma, 2011).

There are many reasons people choose to develop for specific platforms, but one advantage to Apple's iOS is there is only one operating system for all Apple devices (Boardman, 2012). People who develop for Apple have the security of knowing as long as the user has an updated iOS on an Apple device the application will be able to run (Boardman, 2012). Before the iPhone 5 and iPad Mini, there were only two Apple screen sizes: the iPhone/iPod and the iPad which meant the developer could develop for one and recycle most of the code to be used for the other (Boardman, 2012). This meant developers needed to maintain two separate applications (Boardman, 2012). With the addition of the iPhone 5 and iPad Mini, two more application screen sizes were added that need to be maintained. Some developers write and compile an application so that the application can work well on all screen sizes (Boardman, 2012). The downfall to this is that the full screen may not be utilized fully on every device. Maintaining one set of code may be easier for some developers. Developers who use a Windows-based PC face the obstacle of there being no "official" way to develop applications for iOS on their

machines (Boardman, 2012, p. 45). One way around this without having to purchase an

Apple computer is to purchase a Windows-based program that allows development for

iOS (Boardman, 2012). These programs however do not do the compiling of the

application (Boardman, 2012). That would still need to be done on an Apple operating

system (Boardman, 2012). There are companies that allow developers to send in their

code to be compiled and sent back to them for a price, but this could end up being costly

especially if debugging is needed.

Mobile Platform Languages

The differences between iOS and Android are only a small part of the platform

differences mobile developers face. There are numerous programming languages a

developer would need to know in order to develop native applications for the various

platforms on the market, as seen in Figure 3 (Charland & Leroux, 2011). Each platform

also includes separate families of devices, programming languages, development kits,

and distribution markets (Corral et al., 2012a). This has led many developers to use

multi-platform development applications in the process of creating mobile applications

(Corral et al., 2012a).

Required skill sets for nine mobile OSs.

Mobile OS Type	Skill Set Required
Apple iOS	C, Objective C
Google Android	Java (Harmony flavored, Dalvik VM)
RIM BlackBerry	Java (J2ME flavored)
Symbian	C, C++, Python, HTML/CSS/JS
Windows Mobile	.NET
Window 7 Phone	.NET
HP Palm webOS	HTML/CSS/JS
MeeGo	C, C++, HTML/CSS/JS
Samsung bada	C++

Figure 3. Required Skill Sets for Mobile OSs. (Charland & Leroux, 2011, p. 51, reprinted
with permission).

Multi-Platform Development Applications

Multi-platform development applications offer a solution of "develop once, deploy everywhere" (Blom et al., 2008; Corral et al., 2012b, p. 742). The software life cycle is significantly shortened with the use of multi-platform development applications as illustrated in Figure 4 (Corral et al., 2012a). The only item all of the mobile platforms have in common is that they all ship with a mobile browser that is accessible programmatically from the native code (Charland & Leroux, 2011). A browser instance can be instantiated on each platform and interact with its native code through the use of the JavaScript interface (Charland & Leroux, 2011). There are differences among browsers, but they are very minimal in comparison to native coding on each platform (Charland & Leroux, 2011). Developers using multi-platform development applications can create mobile applications that run in the mobile browser (Corra et al., 2012b). This would allow the use of common web development programming languages, such as HTML, CSS, which operate the functionality of the mobile device through a set of application program interfaces (Corral et al., 2012b).
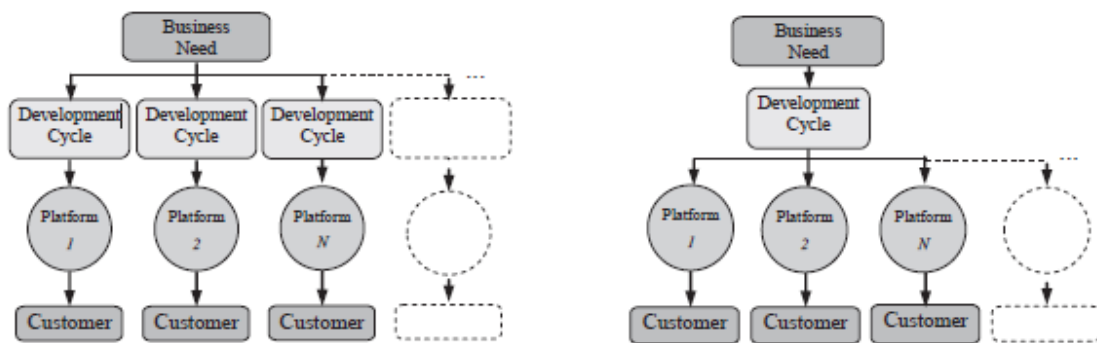
Figure 4. Traditional Development Model (left); Multi-Platform Development Model (right). (Corral et al., 2012a, p. 1203, reprinted with permission).

Just as developing mobile applications individually on each target platform has advantages and disadvantages, so too does developing mobile applications using a multi-platform development application. See Tables 1-3 for an overall summary of trade-offs, advantages, and disadvantages.

Table 1. Potential Tradeoffs of Single-Platform Versus Multi-Platform Development. (Corral et al., 2012a, reprinted with permission).

	Single-platform paradigm	Multiplatform paradigm
Development Tools	Offers native development tools exploiting the potential of a specific platform	Overcomes the constraint of utilizing different languages and frameworks for each platform.
Development Practices	Requires mastering the use of diverse languages, operating systems and development tools.	Takes advantage of knowledge and expertise already attained by programmers.
Development Cycles	Requires repeating platform-specific efforts for each target, for each development cycle.	Develop once, deploy anywhere
User's Experience	Delivers applications with a true native experience, exploiting all device's resources.	Do not allow access (or provide limited access) to some features of the mobile device.
Application Marketing	Bounded to a single application marketplace	Applications can be distributed through a variety of marketplaces.

Table 2. Potential Advantages of Multi-Platform Development. (Corral et al., 2012a, p.1205, reprinted with permission).

	Software development	Application marketing
Customer	Users may experience applications developed for a single platform, compare and prefer.	Applications availability is not limited to a single distribution market. Applications available with more quality, at less price.
Developer	Reduces the costs of conducting redundant activities, receiving training, purchasing tools.	Allows developers to promote and profit from different distribution markets.
Platform Provider	Platforms may take advantage of applications originally developed for another OS.	Promotes competition across platforms. More quality, less price for their customers.

18

Table 3. Potential Disadvantages of Multi-Platform Development. (Corral et al., 2012a, p.1205, reprinted with permission).

	Software development	Application marketing
Customer	Applications do not offer a native user experience or do not exploit all device's capabilities.	Attractive applications from other platforms will not be available.
Developer	Development tools still require improvements: (limited access to some features of the mobile device). Deployment requires platform-specific troubleshooting and customization.	Introduces the need to upgrade and maintain applications in diverse marketplaces.
Platform Provider	Investment made on research and development, and company's best practices may be involuntarily shared.	Since applications are available in different operating systems, applications are not a driver to prefer a platform.

According to a 2011 article by Andre Charland and Brian LeRoux titled "Mobile Application Development: Web vs Native", "the performance argument that native apps are faster may apply to 3D games or image-processing apps, but there is a negligible or unnoticeable performance penalty in a well-built business application using Web technology" (Charland & Leroux, 2011, p. 49). A 2012 study on multi-platform application performance stated, "The discussion on target-agnostic development on mobile devices has been covered by works that forecast a promising growth on the use of the web browser as execution environment" (Corral et al., 2012b, p. 737). However, this study also found a significant gap in performance between a native mobile Android application and a multi-platform mobile web application developed using PhoneGap (Corral et al., 2012b). This study concluded that the web-based implementation was slower due to an architecture that required invoking methods using at least one callback and waiting for its response which is only increased with the complexity of the

application (Corral et al., 2012b). The results of this performance study can be found in Table 4.

Table 4. Comparison of Execution Time Between Android Native Application and PhoneGap Web Application (Corral et al., 2012b, p. 740, reprinted with permission).

| Measured Job | Arithmetic Mean (milliseconds) | | Standard Deviation | | Geometric Mean (relative) | |
	Native App	Web App	Native App	Web App	Native App	Web App
Access to accelerometer	0.7136	2.0021	0.9984	3.0025	1.0000	2.5974
Launch sound notification	18.4835	26.7481	13.3665	47.5036	1.0000	0.6534
Trigger vibrator	1.5134	3.2222	1.2234	4.1248	1.0000	2.2593
Request data from GPS	2.1881	809.2352	6.7244	12.5523	1.0000	528.9298
Request network information	1.1015	1.01419	1.2052	0.6096	1.0000	1.1044
Write a file	4.7146	7.9221	9.2085	6.4558	1.0000	3.3657
Read a file	13.3036	255.7381	13.8829	74.1943	1.0000	29.9005
Retrieve data from contact list	95.8686	1841.4689	13.8747	491.5454	1.0000	18.7518

Summary

Mobile application development has become very popular with the growing number of mobile platforms on the market, and not just among experienced developers but also among novices. Mobile application development can be expensive and time consuming as developers typically want to deliver their products to a variety of users using various platforms, but choosing to develop for one platform can delay development and release on a subsequent platform. The large variety of platforms each involving separate standards, programming languages, and distribution markets, has led some developers to turn to multi-platform development applications in the race to create the next popular mobile application.

Multi-platform development applications are based on the premise, "develop once and deploy everywhere" (Corral et al., 2012b, p. 742). With this in mind, developers can focus on which problem their application is solving and how to improve the application rather than slow individual platform development. However, there are trade-offs such as performance issues that are now being recognized when switching from native application development to multi-platform application development.

CHAPTER III

RESEARCH METHODS

Introduction

The purpose of this study was to investigate multi-platform application use for mobile application development, ease of use, missing capabilities, and proper functionality on two target platforms; iOS and Android. Chapter III describes the research methodology and procedures used in the study.

Research Methodology

Despite the growing number of developers, whether novice or expert, many are switching from single platform development to multi-platform development applications (Corral et al., 2012a; Corral et al., 2012b). Limited research was available regarding distinctions among the various multi-platform development applications and the developers creating them. Before a developer chooses a multi-platform application for developing mobile applications, capabilities, features, ease of use, and functionality for each multi-platform application needed to be identified in order to make the best decision for his or her needs. Three multi-platform development applications were identified based on popularity and cost. Investigation into these multi-platform development applications was conducted by examining user guides and developing a basic test application on each multi-platform development application for iOS and Android. The test application outputs were then examined and analyzed.

User Guides and Examples

Each multi-platform development application came with user guides, documentation, and examples on how to get started. The user guides and documentation were thoroughly read in order to better understand each multi-platform development application's abilities and functions. Examples were run and functionalities tested.

Simple Application Design

Each multi-platform development application used was analyzed based on experience of creating a simple application which included but was not limited to reading through documentation and examples, code portability, application performance, ease of use, and development time. The simple application was written in HTML, JavaScript, and CSS. It included five parts: device/platform information displayed at the top, a button that when pushed played a sound, a button that when pushed changed the background color of the main screen, an implementation of the game of rock, paper, scissors, and the visual appearance of a simple submission form that would include input from a user. The idea for the first three parts of the test application came from an example application in MoSync in order to fully compare, contrast, and analyze the various multi-platform development applications. A rough sketch of the test application can be seen in Figure 5. Each test application was run using the Android emulator and iOS simulator.
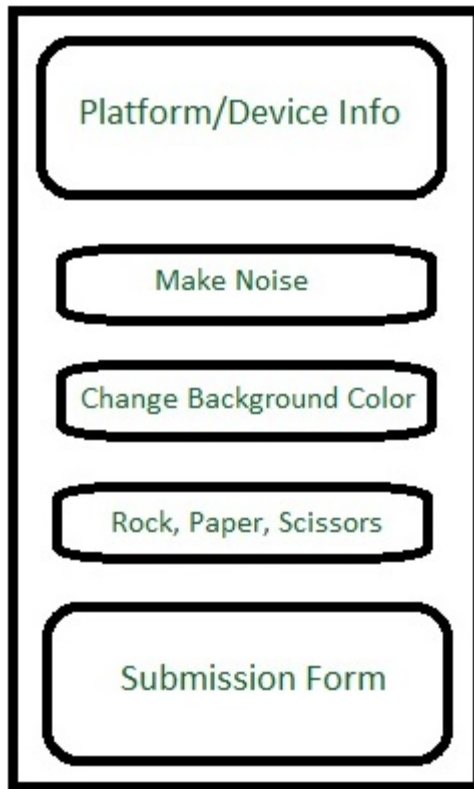
Figure 5. Rough Sketch of Test Application.

CHAPTER IV

RESULTS AND ANALYSIS

Overview

The purpose of this study was to investigate the multi-platform development applications MoSync, Appcelerator, and PhoneGap, create a test application using each multi-platform development application, run the test application on the Android emulator and iOS simulator to determine performance, and determine which multi-platform application was best suited for allowing a developer to create a mobile application that could be utilized on a variety of platforms.

Chapter IV contains descriptive analysis of documentation of MoSync, Appcelerator, and PhoneGap, components of MoSync, Appcelerator, and PhoneGap, the test application developed in MoSync, Appcelerator, and PhoneGap, the results of running the test application on the Android emulator and the iOS simulator, and documentation of problems encountered.

The following research questions guided the study:

1. Was each multi-platform development application well-documented for ease of use for any skill level of a developer?

2. What comprised each multi-platform development application?

3. What was the level of difficulty in using each multi-platform development application?

25

4. Did the test application work as expected on each target platform?

Pre Development

Three applications, MoSync, Appcelerator, and PhoneGap, were the multi-
platform development applications chosen for this study. Each application was free to
download and use. Although Android applications could only be compiled on a Mac or
Linux, iOS applications could only be locally compiled on a Mac. Thus a 2013 MacBook
Pro with OS X Mountain Lion 10.8.5 was the computer used for development. A
prerequisite to development on any of the multi-platform applications was to download
the target platforms' SDKs (software development kits). The Android SDK download,
called the SDK ADT (Android Developer Tools) Bundle, included a version of the
Eclipse IDE (integrated development environment) with ADT plugin, Android SDK
Tools, Android Platform-tools, the latest Android platform, and the latest Android
emulator. The SDK ADT Bundle (identified as the October 30, 2013 build) required Mac
OS X 10.5.8 or later and was approximately 3.1 GB in size when installed. The iOS SDK
download was called Xcode and was approximately 6.1 GB in size when installed. Xcode
(version 5.0.1) included the Xcode IDE, LLVM compiler, instruments, iOS simulator,
and the latest OS X and iOS SDKs.

In order to use the Android emulator, profiles needed to be created within the
Android Virtual Device Manager. These profiles specified the type of device, Android
API Level, CPU, and memory and storage options. A profile also needed to be created
within XCode. The main option to be specified within XCode was the Base SDK to be
used, however, the iOS simulator allowed the device to be changed on the fly as well as

the iOS version to be chosen for each device either using version 6 or 7. The devices used

in this study were the iPhone (3.5-inch) and iPhone Retina (4-inch).

MoSync

The first multi-platform development application used was MoSync and was

located online at www.mosync.com. MoSync's website offered two free open source

tools for building cross platform mobile applications; MoSync Reload and MoSync SDK.

According to MoSync's website, "MoSync Reload is targeted exclusively at

HTML5/JavaScript development, while the SDK is targeted at both C/C++ and

HTML5/JavaScript development" (MoSync, 2013c). It also highlighted the fact that SDK

would produce native applications for multiple platforms and contained the Eclipse-based

MoSync IDE. The SDK could target up to nine platforms, using one single code base.

Figure 6 highlights the capabilities of MoSync Reload compared to MoSync SDK. Based

on the capabilities outlined in Figure 6 MoSync Reload versus MoSync SDK, the

MoSync SDK could do everything MoSync Reload could do plus it possessed additional

capabilities. For this reason, the MoSync SDK was the chosen tool within MoSync.

The MoSync SDK was downloaded after creating a user account on the MoSync website.

The version 3.3.1 MoSync SDK took approximately 459 MB of disk space when

installed. The MoSync IDE layout can be seen in Figure 7. There was a project explorer

window on the left, code editor in the center, target profiles to the right, and console

output at the bottom of the MoSync IDE. The code editor did not offer error detection or

code completion. At the top of the IDE there was a set of buttons for building a project,

Capability	MoSync Reload	MoSync SDK
HTML5/JS development	Yes	Yes
C++ development	No	Yes
Fast JS/HTML turn around	Yes	Yes
Interactive JS Workbench	Yes	No
JS Debugging	Inspector	Full debugger
Use external editor	Yes	Possible
Eclipse-based IDE	No	Yes
Quick to get started	Very easy	Easy
Build native apps	No	Yes
Supported operating systems	Windows, OS X, Linux	Windows, OS X

Figure 6. MoSync Reload Versus MoSync SDK. (MoSync, 2013c).

choosing a target simulator/emulator, and launching a project in the selected target.

MoSync required the user to create and manage run configurations within its IDE, in

addition to the normal emulator/simulator setup within the target SDKs. If there were

errors in the code there was no way to easily identify where the error occurred due to the

fact that the emulator/simulator screen within the application simply shown black.

MoSync's webpage offered links to user guides, JavaScript and C/C++ API

references, forums, an issue tracker, GitHub repositories, example applications, and

videos. As seen in Figure 8 Create Your First App, found on the main developer webpage

of the MoSync website, MoSync offered a starting point for developers to become

familiar with MoSync.
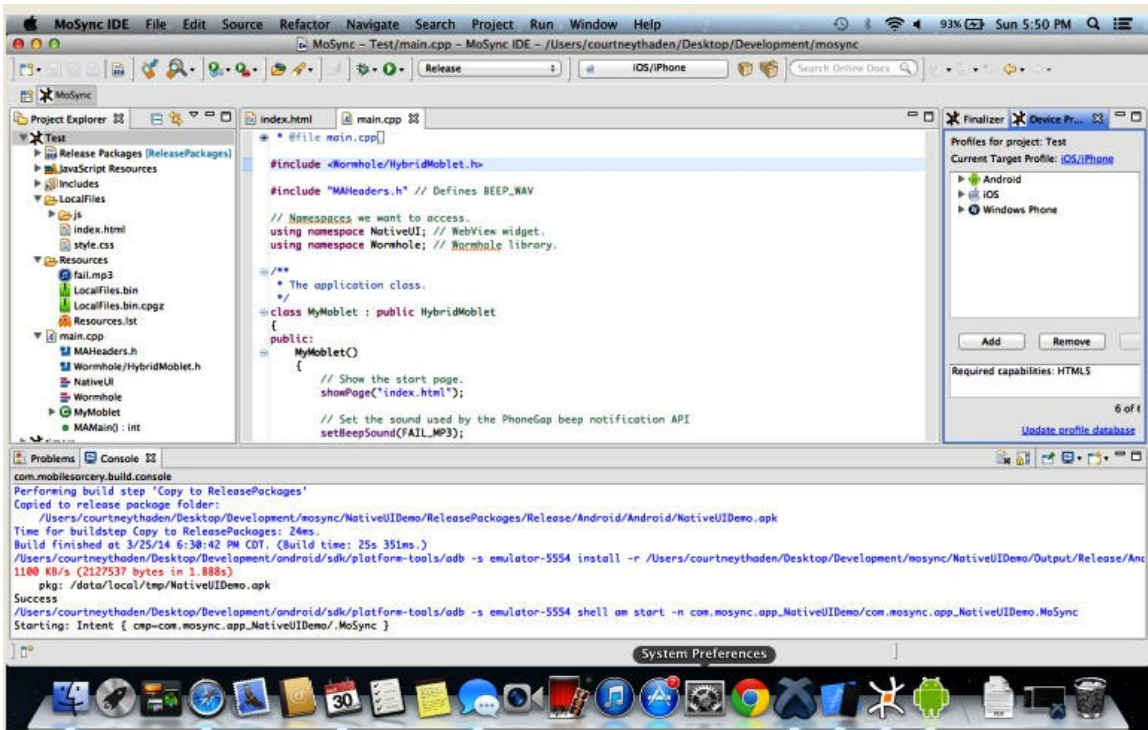
Figure 7. MoSync IDE Screenshot.
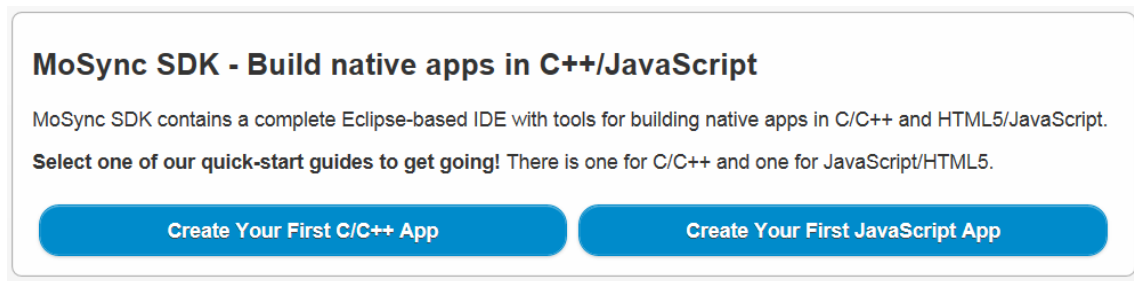
Figure 8. MoSync Create Your First App. (MoSync, 2013a).

The JavaScript directions were used to create a new project. By following the directions and with the use of the HTML5/JS WebUI project template, the project was permitted to be built and allowed to run on a target platform. No extra code was added. The results of running the application on the Android emulator as an Android 4.4 (API level 19) device are shown in Figure 9.
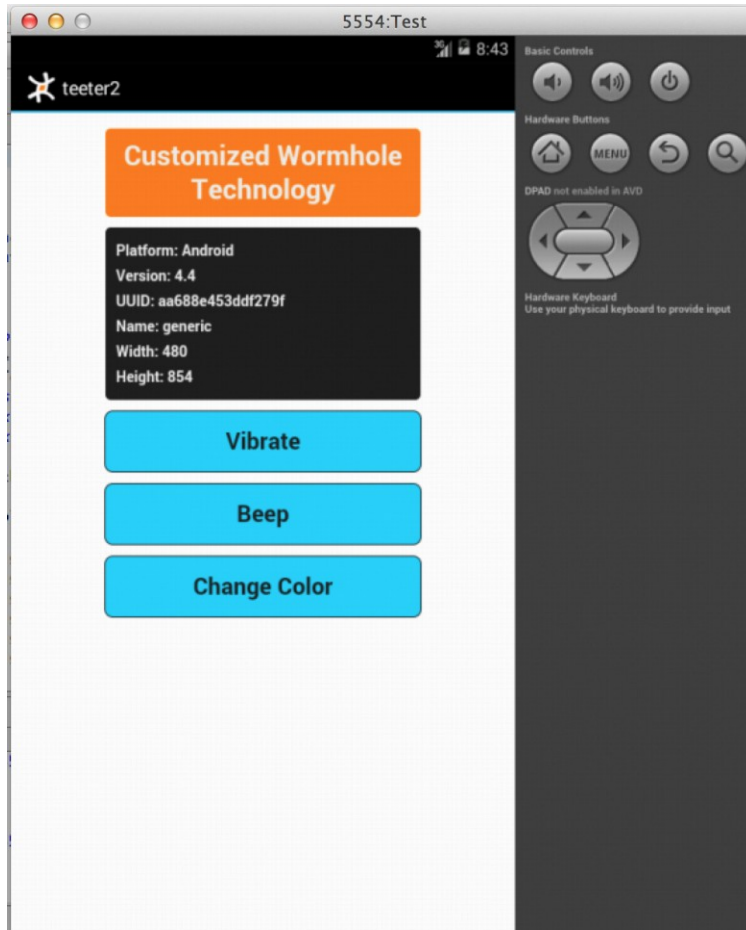
Figure 9. MoSync HTML5/JS WebUI Template Project Screenshot.

Building from the example provided, the test application project called Test was

developed in the same manner. The chosen target profiles were Android and iOS. The

project structure is shown in Figure 10. The LocalFiles folder contained the JavaScript

and HTML5 files for the project. The code files that were changed and added for the Test

application are presented in Appendix B, while Appendix A shows the original files

before they were edited (HTML5/JS WebUI Template Project). The script.js, style.css,

and fail.mp3 files were added to the Test application project through the course of

development. The script.js file contained the code for the comparison made in order to

determine a winner for the rock, paper, scissors game. The wormhole.js file was

automatically included by MoSync and was part of the MoSync Wormhole Library. This

library contained two parts; the JavaScript API and the C/C++ API. The MoSync

Wormhole JavaScript Library gave access from the HTML5 application to the native user

interface components and hardware of the device. The index.html file was the main file to

which the developer would add code. The code added was in HTML5 and JavaScript and

contained the elements of the application. The style.css file gave the specific style to the

elements. The main.cpp file contained the main function that was called when the

program started. The Resources folder contained any media files used within the

application.
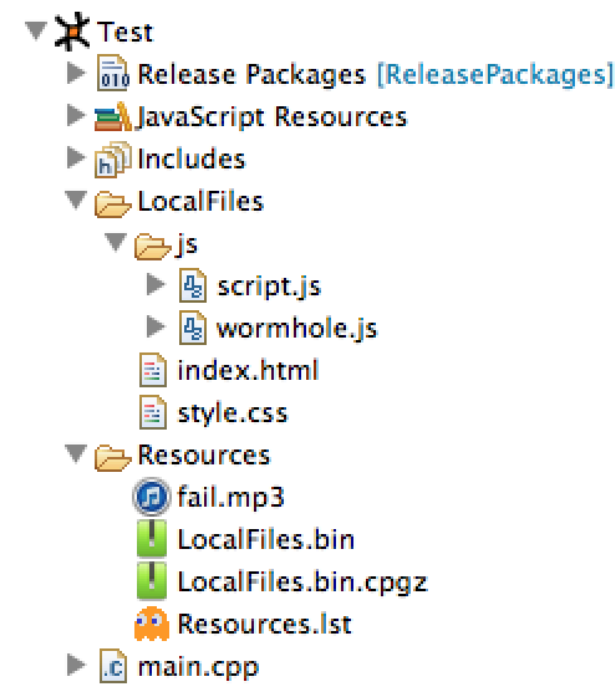
Figure 10. MoSync Project Structure Screenshot.

Device profiles were created within the MoSync IDE in order to build and run the

application on the iOS simulator and Android emulator. MoSync automatically started

the Android emulator without any Android software being opened when that device type

was selected for the target. However the iOS simulator did not act in a similar manner.

XCode and MoSync did not work well together. XCode needed to be open in order to

install and launch the test application in the iOS simulator. This however did not

guarantee the application would install properly on the iOS simulator. After the

application performed properly a few times on the iOS simulator, MoSync began to give

a console error every time any application was targeted for the iOS simulator. The

console error read "iOS simulator failed to install the application." Many forums were

searched and it was discovered many developers had the same problem not specific to

MoSync, but to XCode. There were various fixes people had insisted worked to fix their

problems, but there were still many people that had no solution and believed it to be a

bug of XCode. The only fix that temporarily worked for the MoSync Test application

was to uninstall and reinstall XCode. This was only a temporary fix though as after a few

times of the iOS simulator working successfully, it again returned the same console error

continuously when iOS was the target. The few times that the iOS simulator was

working, screen shots of the application were taken. Android screen shots were also

captured.

Figure 11 shows the Test application as it appeared on the iOS simulator as the

iPhone (3.5-inch). The application was longer than the screen but it was scrollable, so the

rest could be easily seen. The initial background color started out as white until the

change color button was pressed which then changed the background color to a randomly

selected color.
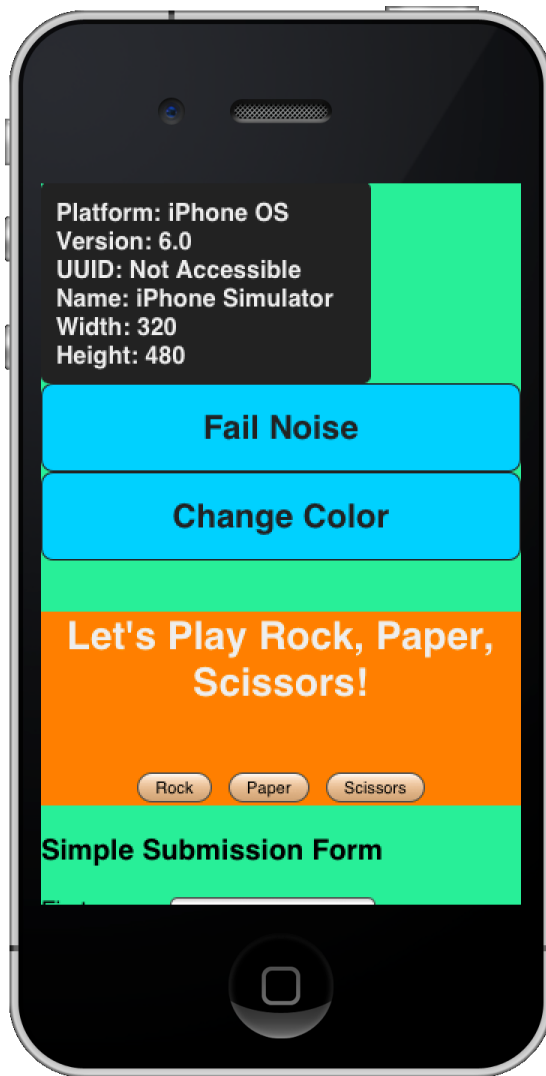
Figure 11. MoSync iOS Test Application View 1 Screenshot.

Figure 12 shows the Test application as it appears on the iOS simulator as the iPhone (3.5-inch). The application was displaying the alert that appeared after playing the rock button in the game of rock, paper, scissors.

Figure 13 shows the Test application as it appears on the iOS simulator as the iPhone (3.5-inch). The application was scrolled down to show the simple submission form. The submission form was not fully coded to send the results anywhere. That would need to be added if this was an application that was going to be released and published.
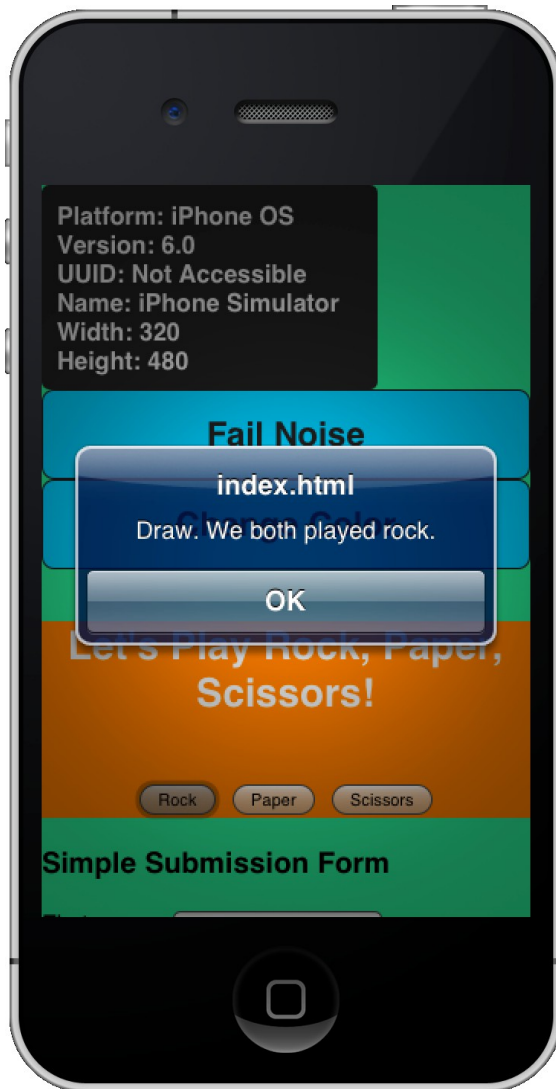
Figure 12. MoSync iOS Test Application View 2 Screenshot.

Figure 14 shows the Test application as it appears on the Android emulator as an Android 4.4 (API level 19) device. The application was displayed as it would appear when it initially opened. The entire application fit within the screen.

Figure 15 shows the Test application as it appears on the Android emulator as an Android 4.4 (API level 19) device. The application was displaying the alert that appeared after playing the scissors button in the game of rock, paper, scissors.
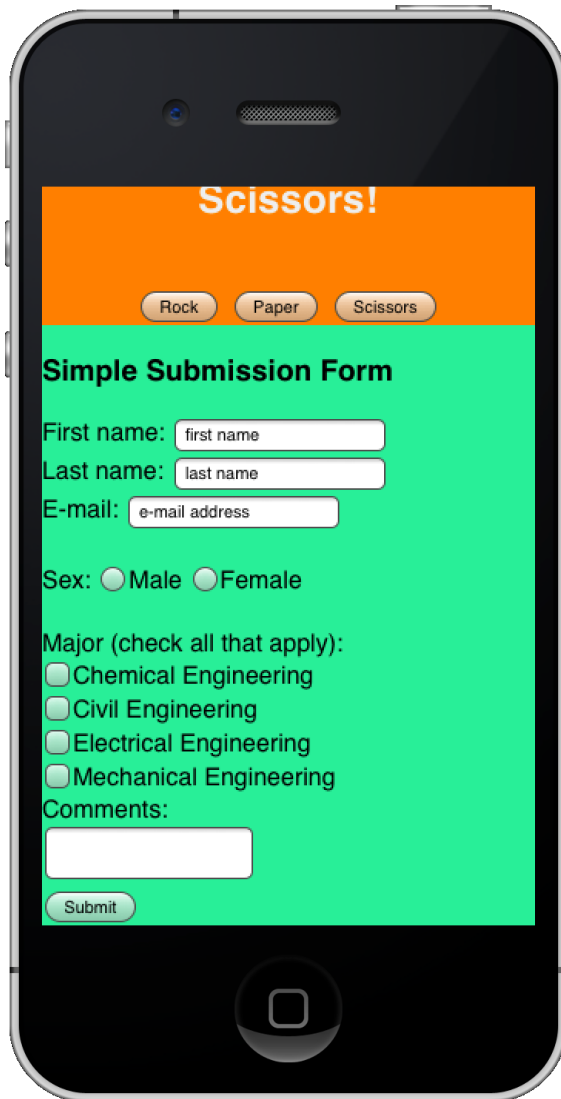
Figure 13. MoSync iOS Test Application View 3 Screenshot.

After further investigation of MoSync it was discovered that although the first

sentence within Figure 8 leads a developer to believe the directions described how to

build your first "native app" in JavaScript, in reality the application being created was

closer to being a web application and not necessarily a native application. According to

the MoSync (2013b) website, the definitions for the types of projects that could be

created are listed as follows:
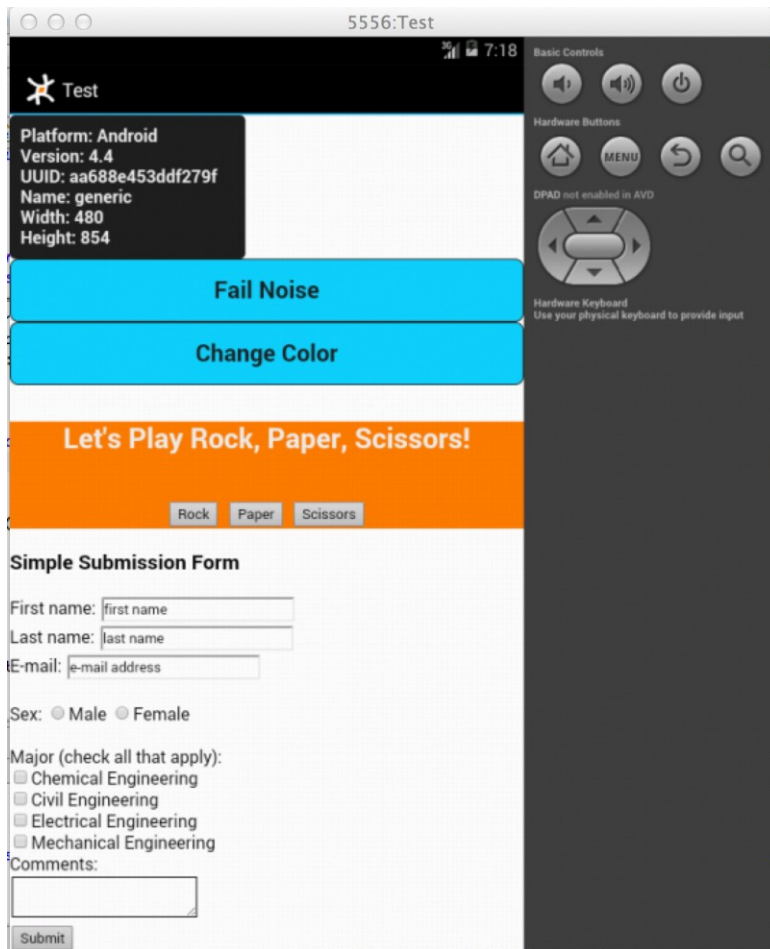
Figure 14. MoSync Android Test Application View 1 Screenshot.

HTML5/JS WebUI Project - Gives you an app with the user interface in

HTML/CSS, set up with libraries for accessing device functionality from

JavaScript.

HTML5/JS NativeUI Project - Gives you an app with a native user interface,

written in HTML/JavaScript, set up with libraries for accessing device

functionality from JavaScript.
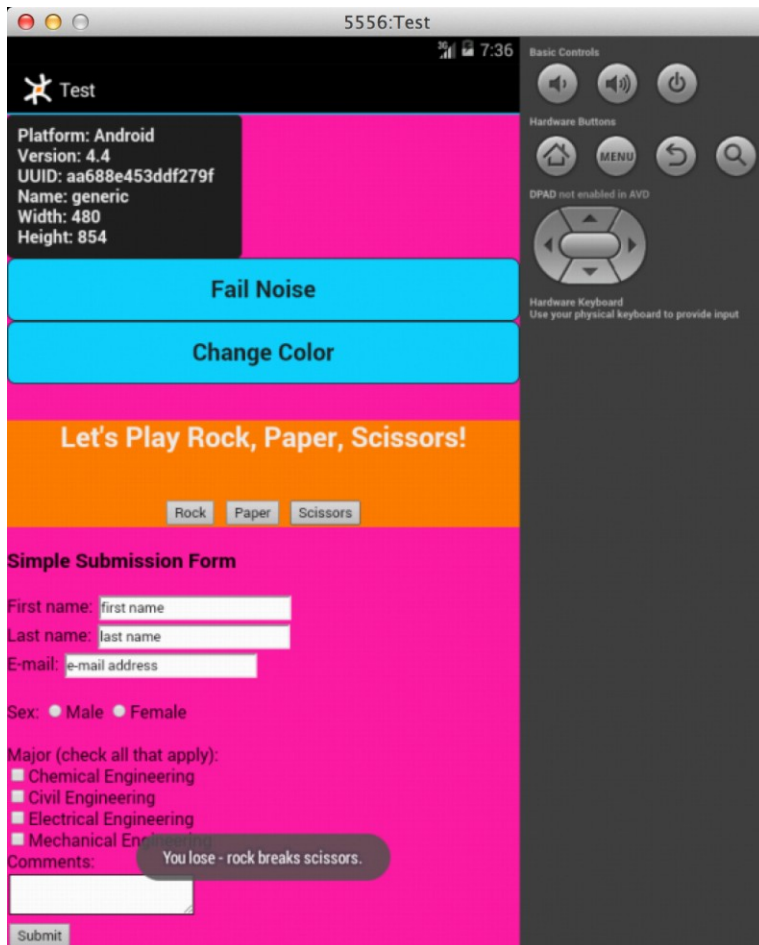
Figure 15. MoSync Android Test Application View 2 Screenshot.

HTML5/JS/C++ Hybrid Project - This is like the HTML5/JS WebUI template

app, but it is set up to show you how to extend your app with code written in

C/C++ (MoSync, 2013b).

Noting the difference in the definition between the MoSync WebUI Project and

the MoSync NativeUI Project, the HTML5/JS NativeUI Project template was built and

run without any extra added code. Figure 16 shows the result of running on the Android

emulator as an Android 4.4 (API level 19) device.
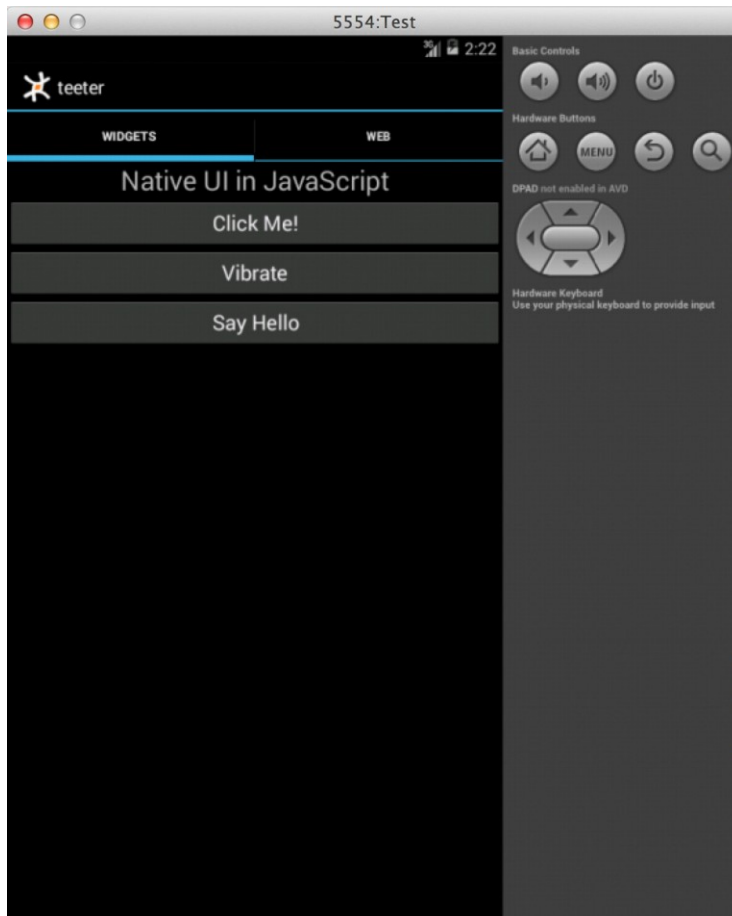
Figure 16. MoSync HTML5/JS NativeUI Template Project Screenshot.

The test application was then built using the MoSync HTML5/JS NativeUI

Project template with the project name of TestApp. The project structure was the same as

the WebUI project. The code files that changed are presented in Appendix D, while

Appendix C shows the original files before the code was edited (HTML5/JS NativeUI

Template Project).

The NativeUI application used widgets which contained HTML markups and

JavaScript. Within the MoSync JavaScript API document there were descriptions of

native user interface widgets that could be used. These widgets allowed a developer to

create common components found in applications such as screens, buttons, images,

labels, etc. and use the predefined property attributes to personalize each. There was a big gap in the documentation as it was missing values that each property attribute would accept and recognize. Through a web search a more detailed document for MoSync 3.3 was found at the MoSync website. The document gave details on values for property attributes, but the widgets were defined using names that were slightly different than the ones found in the MoSync JavaScript API due to it being the MoSync C/C++ API. The various types of documentation MoSync offered on its website were very basic and missing many important details. There were links to pages that were buried very deep within the website which made it hard to navigate through the documentation.

The TestApp project replicated the look of the Test project, but instead of a submission form which required a web connection to send, it was replaced with a game that allowed the user to choose pieces of an outfit for a monkey named Jack. Buttons allowed the user to place items on Jack one at a time. The items included were a suit, hat, and bow tie. These image files were added to the project. In order to implement the buttons, event listeners were required to make an image appear on the screen when the corresponding button was clicked. There was also a reset button that would place Jack back in his undressed state. In order to set each item of clothing to the correct location in relation to Jack, it was first necessary to define each image to appear right away in the development process. Next the event listener needed to be implemented to allow the buttons to make the images appear. Many various methods were attempted in order to try and implement this, but it was to no avail. One of the attempted implementations required setting the event listener to create the image upon a button click. The second attempt required the image to be defined initially but setting the image path property to be empty

and upon a button click setting the image path property to the appropriate image. The problems with the documentation resulted in not being able to properly implement these buttons. The images for Jack's clothing were defaulted to appear upon the application opening due to the button difficulties. The result of the TestApp project can be seen in Figure 17 which was run on the Android emulator as an Android 4.4 (API level 19) device. The iOS simulator again would not work even after uninstalling and reinstalling XCode.
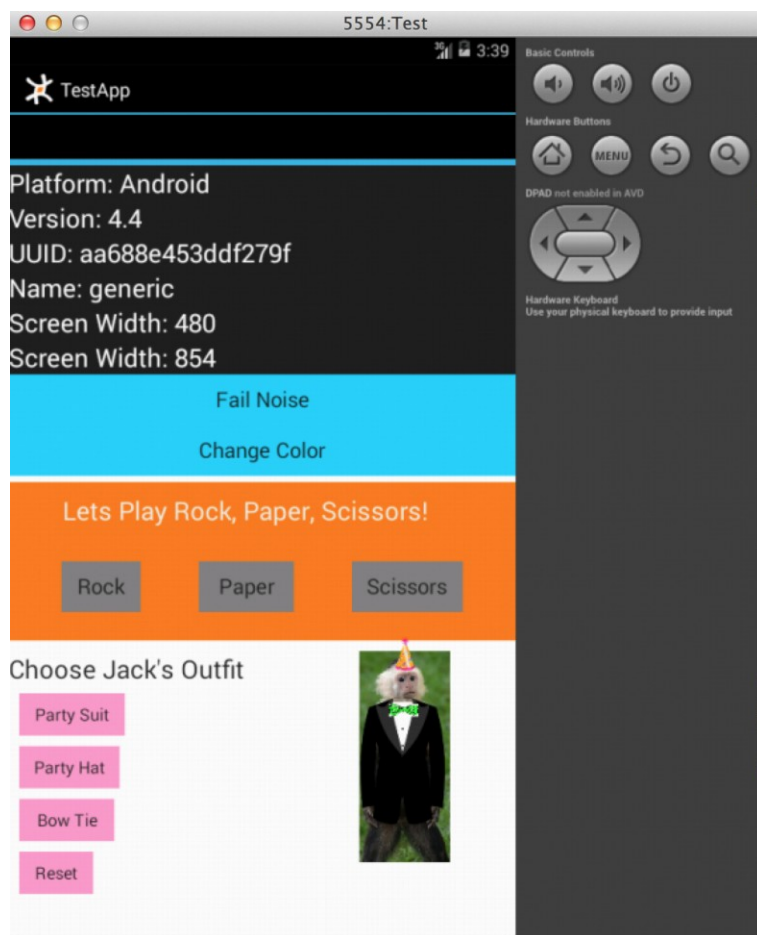
Figure 17. MoSync Android TestApp Application Screenshot.

MoSync IDE did offer a C/C++ debugger and JavaScript debugger. The build configuration needed to be set to "Debug" and the debug runtime needed to be used. The

debugger allowed hit counts and breakpoint conditions to be set that enabled a developer to stop program execution at a specific line number, stop program execution after a script loaded, or suspend a script whenever an exception was thrown in the code. The MoSync IDE layout would change to a debug layout that allowed the developer to view variable values in real time, stepping into, over, and returning to code lines, suspending execution, and terminating execution. Some common JavaScript debugger troubleshooting issues were noted on MoSync's website such as: making sure if a real device was being tested with the debugger that the device was connected to the same network as the debug server, reloading the program if the debugger was hanging because sometimes the client and server would end up out of sync, and reloading the program if an error dialog said the session had timed out because the IDE would time out and lose its connection after half a minute or so. MoSync also noted that due to the single thread nature of JavaScript, that when stepping into, over, and returning to a line of code, sometimes the debugger would end up in a completely different place especially for timed executions triggered by the JavaScript setTimeout function. MoSync claimed that issue happened rarely, but they planned to fix it in a future release of the MoSync JavaScript On-Device Debugger.

Appcelerator

Appcelerator, the second multi-platform development application used in this project, was accessed from www.appcelerator.com. Appcelerator offered two platforms known as the Appcelerator Platform and Appcelerator Titanium. The fee based Appcelerator Platform, according to the Appcelerator website, "supports the entire mobile lifecycle along with commercial support, service level agreements and technical training," (Appcelerator, n.d.a) whereas Titanium was free, and was "primarily a solution

41

for the development of mobile applications" (Appcelerator, n.d.d). The Titanium SDK

was an "open source JavaScript-based development environment" that offered over 5000

device and mobile operating system APIs, Studio, a powerful Eclipse-based IDE, Alloy,

an MVC Framework and Cloud Services (Appcelerator, n.d.e). Titanium SDK could be

used to create "beautiful native apps across different mobile devices and OSs including

iOS, Android, and BlackBerry, as well as hybrid and HTML5" (Appcelerator, n.d.e).

The native applications, according to the website, were to behave like they were

written in Objective C for iPhone and iPad or Java for Android phones and tablets with

60% - 90% code reuse. This would allow "applications to be built faster and at a lower

cost than any other environment." (Appcelerator, n.d.e). It was also emphasized that

Titanium was not a "write once, run everywhere" cross platform development tool

(Appcelerator, n.d.c). Instead it was a "write once, adapt everywhere" cross platform

development tool (Appcelerator, n.d.c).

Developers who used Titanium were supposed to "accept and embrace platform

differences" (Appcelerator, n.d.c). An example given by the company regarding platform

differences stated Android application tabs were typically found at the top of the screen

whereas iOS tabs were typically found at the bottom of the screen. The company kept

these kinds of specificities in mind when creating Titanium. They employed these

differences by utilizing "if else statements" based upon the operating system name.

Titanium SDK was downloaded after a user account was established. The account

was free and allowed Appcelerator to identify their customers and send updates and

product information to them. The Titanium SDK version 3.1.3.GA took approximately

222 MB of disk space when installed. Titanium Studio could then be opened and easily

used. The graphical user interface's (GUI) main window was the code editor and

included code completion as well as error notation if there was an error in a line of code.

The breakdown of the project's folders and files were located in the left window called

App Explorer. At the top of the App Explorer window were buttons that allowed the

project to be built and run on the selected target which included emulators/simulators as

well as actual devices. When running an application in the emulator/simulator if there

was an error that was not identified in the code editor such as an undefined variable, the

emulator/simulator displayed an error screen which referenced the undefined variable or

error. Titanium Studio also let the user know when there were available software updates

via a pale yellow box in the lower right of the main screen. See Figure 18.
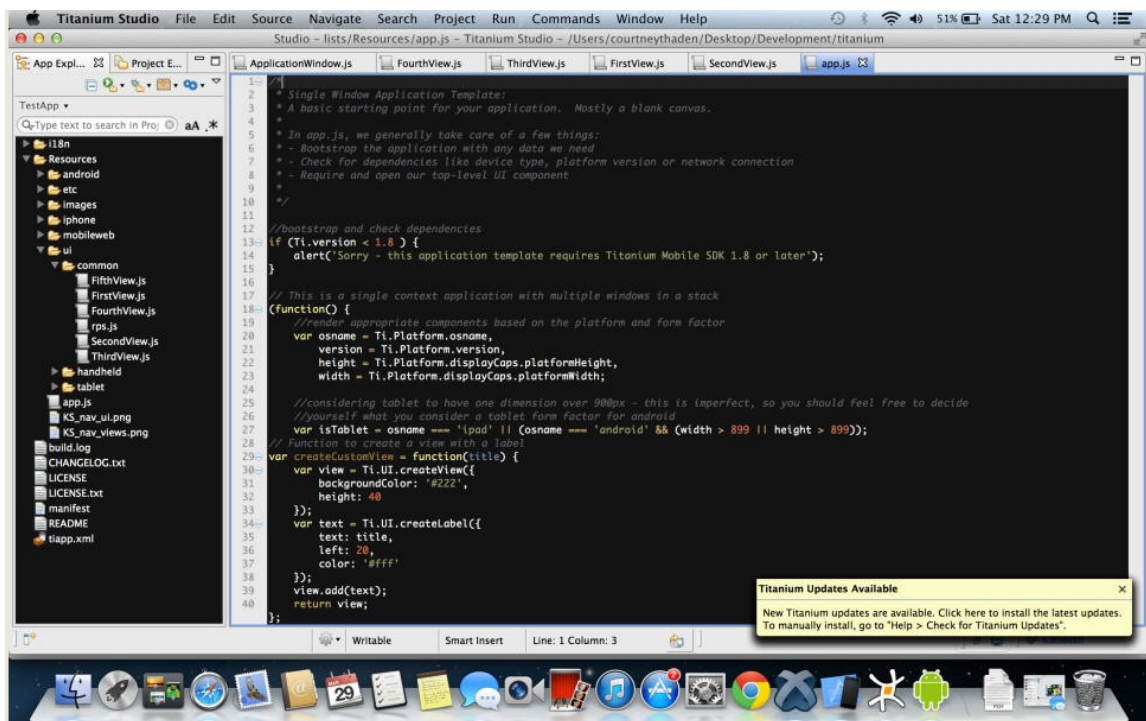
Figure 18. Titanium Studio GUI Screenshot.

Appcelerator had many Titanium resources on its website to enable a developer of any caliber to progress with building applications. There was a vast array of documentation on everything from the basics of getting started in Titanium, installation and configuration guides, API documentation with example code, GitHub repositories, and videos on best practices to more advanced topics. It was noted that Titanium did not have checkboxes but there was a work around. The work around included using a button with an event listener based on clicking that determined when to switch between displaying an unchecked box image and a checked box image. There were also many sample applications that could be imported into Titanium Studio. Developers could build off of these samples or use them to better understand how Titanium worked, although some samples did not use best coding practices according to one of the Titanium videos. Titanium also offered a variety of templates from which a project could be started.

Using Titanium Studio a test application called TestApp was created. The mobile project template chosen was a single window application with a single view. The deployment targets selected were iPhone and Android. Although Mobile Web was also a choice for a deployment target and would have created a web application, it was not selected as a deployment target. The single window application with a single view template without any code added did not install an application in an emulator/simulator. After the project was created, the base folders and files were created. See Figure 19 for the project structure. The TestApp application code that was changed or added is presented in Appendix E.
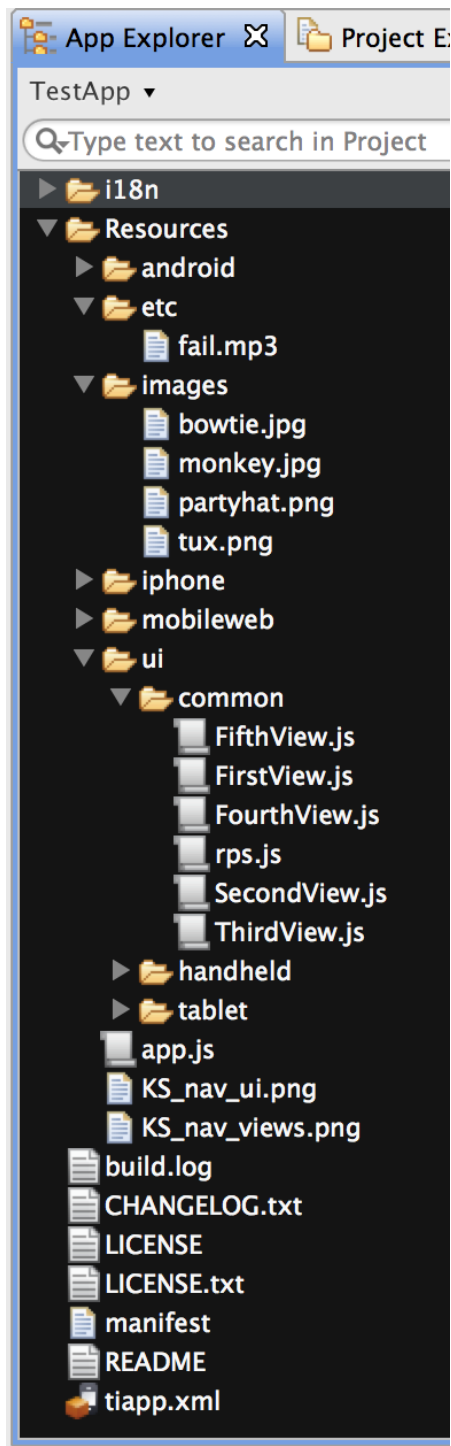
Figure 19. Titanium Project Structure Screenshot.

The i18n folder contained xml files for the languages the application should be

available in, such as English, Spanish, etc. The Resources folder was the main folder that

contained the application contents. Within the Resources folder there was a file called app.js, a folder for each deployment target, and a ui folder which was the user interface folder. The app.js file was a basic starting point for the application. It set up the application window. Within the ui folder there were three folders; common, handheld, and tablet. The common folder contained files for creating "Views." "Views" were similar to divisions in HTML and basically was a way to define a section of the application window. The template started with only the FirstView.js file as a starting point. Within the TestApp application project, the FirstView.js contained the code that created the change color button. SecondView.js contained the code that created the fail noise button. The rock, paper, scissors game code was contained within ThirdView.js, while the code to compare the user's selection to that of the computer was found in rps.js. The platform information code was contained in FourthView.js and the monkey outfit code was found in FifthView.js. Each view file contained a statement "module.exports = <filename>" at the bottom which allowed the code in each file to be pulled into the ApplicationWindow.js file found in the handheld folder.

The ApplicationWindow.js file defined the main application window; the order in which the view files were listed was the order in which they would be displayed in the application. There was also an ApplicationWindow.js file within the Android folder inside the handheld folder. This file was used to define anything specific to only Android for the application window. The tablet folder was the remaining folder found in the ui folder. It defined ApplicationWindow.js specific to tablets. The images folder was added to hold the image files such as the monkey and the parts of his outfit. The etc folder was also added and contained the mp3 file used for the fail noise button.

The only code language used in the Titanium native application by the developer was JavaScript. Through tutorial videos and testing, it was discovered that the iOS simulator took far less time to load applications than the Android emulator when using Titanium. Although the iOS simulator would not work consistently using MoSync, there were no problems running it using Titanium. The output for the TestApp application run on the iOS simulator as the iPhone (3.5-inch) can be seen in Figure 20. All functionality worked in the application using Titanium. Titanium applications do not default to being scrollable. They do, however, have code that can be added to employ the scroll ability. The TestApp application did not include this code.
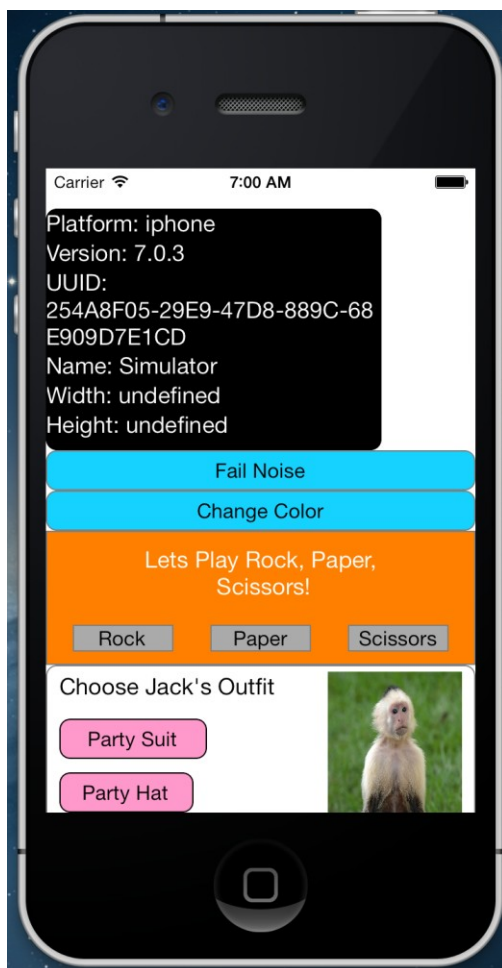
Figure 20. Titanium iOS TestApp Application View 1 Screenshot.

The output for the TestApp application run on the iOS simulator as the iPhone Retina (4.0-inch) can be seen in Figure 21. All of the application fit into this screen.
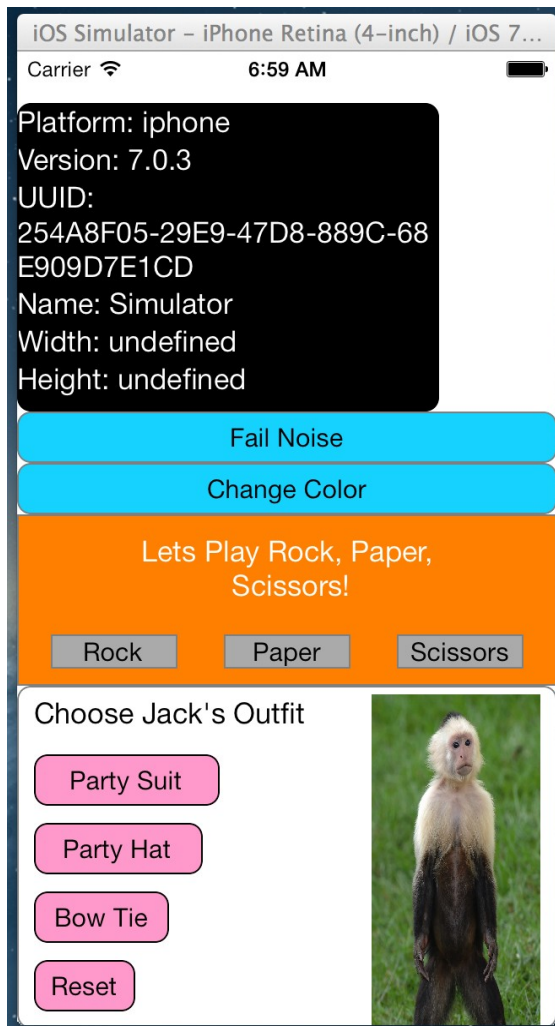
Figure 21. Titanium iOS TestApp Application View 2 Screenshot.

The result of the TestApp project can be seen in Figure 22 which was run on the Android emulator as an Android 4.4 (API level 19) device. Without looking at the code, one may think the application did not display correctly on Android. But there was a reason the Titanium project created a separate ApplicationWindow.js file in the Android folder within the handheld folder. In the TestApp project, the app.js file was changed to make the ApplicationWindow.js file that was used, to always be the one found in the

handheld folder. This was done purposely in order to show what would happen if a developer chose to not utilize the ApplicationWindow.js file in the android folder within the handheld folder. There were certain objects within Titanium's documentation that had properties that were defined differently for Android when compared to other platforms. In order to make the Android application appear similar to the iOS TestApp application, some of the files would have to be rewritten to have the Android specificities and saved under the Android folder. Then the ApplicationWindow.js file in the Android folder within the handheld folder would need to bring in those specific files. In making their multi-platform application in this manner, Titanium allowed the developer to still code in JavaScript, but use objects native to Android. The original app.js file that the project created before changes were made can be seen in Appendix F.

Appcelerator offered a specific set of documents geared towards maximum code reuse for the differences between Android and iOS by creating what they called "best of breed apps" (Appcelerator, n.d.b). In fact, this set of documents called "Cross-Platform Mobile Development in Titanium", claimed that it was not uncommon to reuse 80-100% of the developer's user interface code (Appcelerator, n.d.c).

Similar to MoSync IDE, Titanium Studio GUI offered a debugger that included a debug perspective, the ability to set manual and exception breakpoints, the ability to step through code, a variables view in real time, a console that output error messages, and a build log. The Titanium Studio GUI also needed to be changed to debug mode in order to be used.
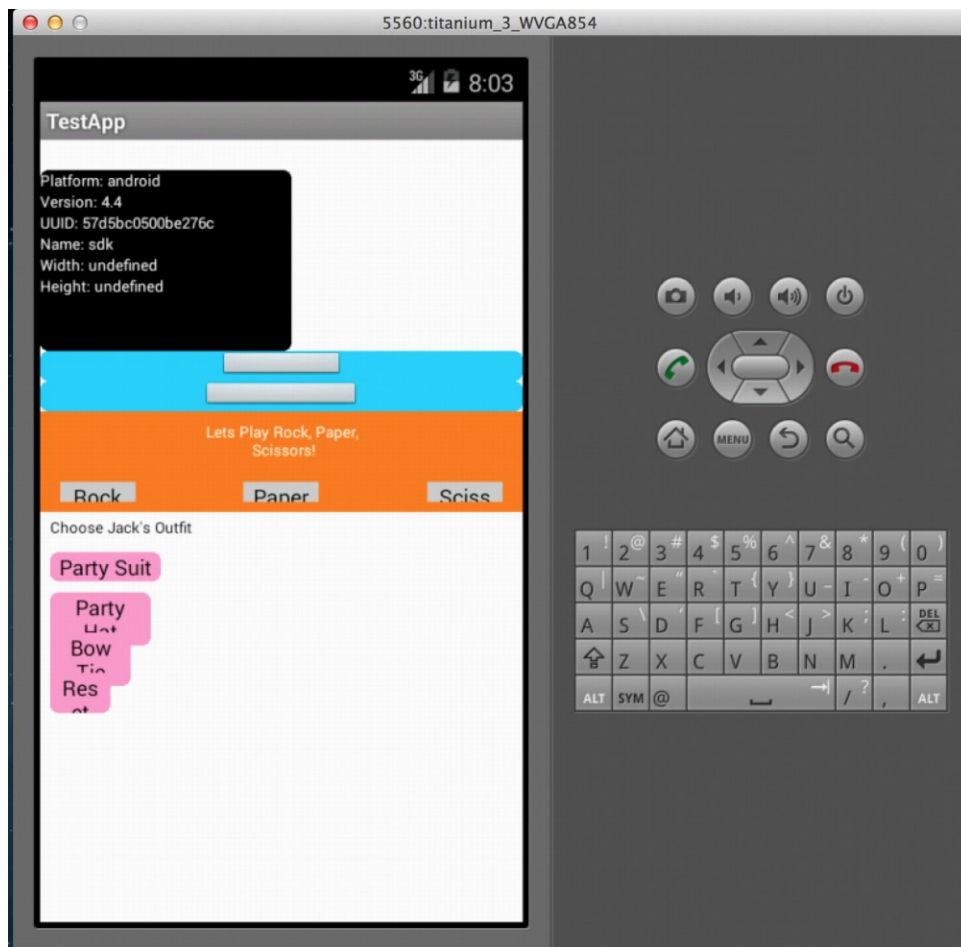
Figure 22. Titanium Android TestApp Application Screenshot.

PhoneGap

The third multi-platform application utilized was PhoneGap which was located at

www.phonegap.com. PhoneGap's website asserts, "PhoneGap is a free and open source

framework that allows you to create mobile apps using standardized web APIs for the

platforms you care about" (PhoneGap, 2014b). PhoneGap used "standard web

technologies such as HTML, CSS, and JavaScript" (PhoneGap, 2014c). Applications

built with PhoneGap are executed in wrappers and accessed device's sensors, data, and

network status through standardized API bindings (PhoneGap, 2014c). PhoneGap strictly

builds web applications. PhoneGap also offered PhoneGap Build, which was a web

50

service that compiled PhoneGap applications remotely for a developer (PhoneGap,

2014d). This option allowed a developer to not have to install and maintain several

mobile platform SDKs (PhoneGap, 2014d). PhoneGap Build was not used in this project.

PhoneGap did not include a GUI. It was strictly a command line interface. The

Terminal application on the MacBook was used for this purpose. The command "sudo

npm install –g phonegap" was used to install PhoneGap 3.1.0-0.15.0. Application files

for the example HelloWorld application were created inside a folder called hello using

the command "phonegap create hello com.example.hello HelloWorld" (PhoneGap,

2014a). The command used to create a project other than the example was "phonegap

build <folderName>" where folderName is the folder name of the project. The project

folder name for the TestApp application was TestApp. This TestApp folder contained the

same files that were in the HelloWorld application. The PhoneGap project structure is

located in Figure 23.The HelloWorld application had a logo image file in the img folder.

This was replaced with the fail.mp3 file for the TestApp application. The config.xml file

complied with the World Wide Web Consortium's (W3C) Packaged Web App, or widget

specification and provided application information and parameters (PhoneGap, 2014c).

The index.html file contained the HTML code for the application. The index.css file

contained the CSS style code properties for the HTML. API bindings were found in the

phonegap.js file. The index.js file contained the JavaScript code for the application. A

developer was required to make changes to the following files: the config.xml,

index.html, index.css, and index.js files. These files from the HelloWorld application are

presented in Appendix G. The TestApp application files were edited in a text editor and

are presented in Appendix H.
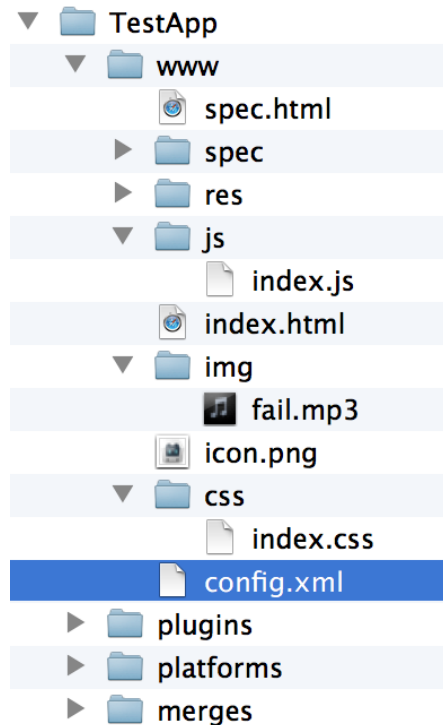
Figure 23. PhoneGap Project Structure Screenshot.

In order to build an application, the build command needed to be run while in the

project's directory (PhoneGap, 2014c). The build command was "phonegap build

<platform>" where platform was the targeted platform (PhoneGap, 2014c). The iOS

version of the HelloWorld application built with no errors. The Android version however

would not compile. The directions for building Android applications required the location

of the Android SDK platform-tools folder and tools folder to be added to the PATH

environment variable (PhoneGap, 2014a). This was done successfully. When the build

command was entered an error appeared that said the local copy of the Android SDK

could not be found. After much troubleshooting and searching PhoneGap documentation

as well as forums, it was determined that many other developers also had this problem.

The issue may have been a bug in the PhoneGap version used. The build command

worked for iOS and iOS ended up being the only platform used with PhoneGap. The

PhoneGap install command "phonegap install <platform>" where platform was the

targeted platform, was used to install the application on the emulator/simulator if a device

was not present (PhoneGap, 2014c). The build and install commands were implemented

in a row by using the PhoneGap run command "phonegap run <platform>" where

platform was target platform (PhoneGap, 2014c). The result of the HelloWorld example

application being run on the iOS simulator as an iPhone Retina (4.5-inch) can be seen in

Figure 24.
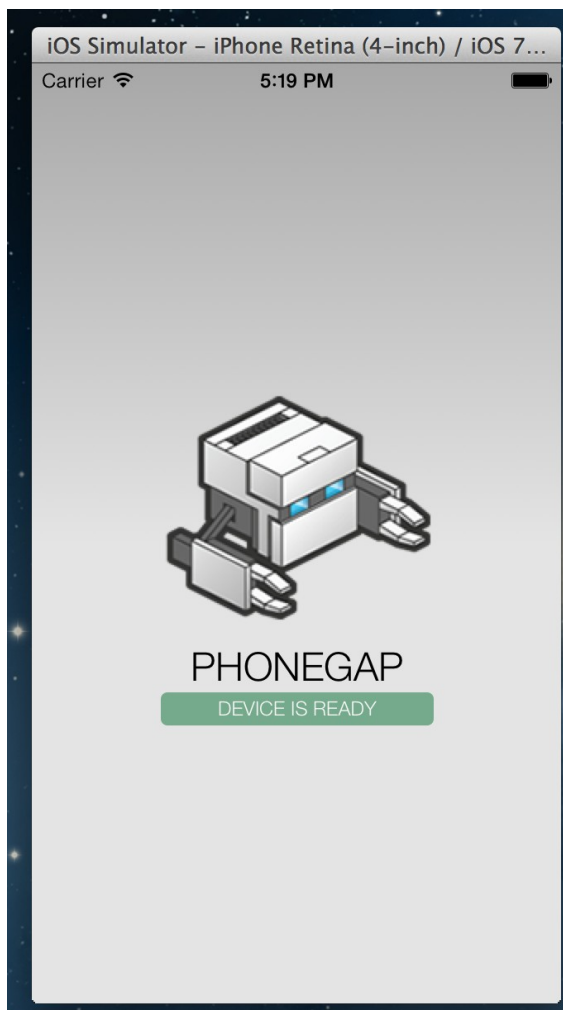
Figure 24. PhoneGap iOS HelloWorld Example Application Screenshot.

The TestApp application successfully built and installed on the iOS platform. The

Terminal output from using the run command can be seen in Figure 25.

```
Courtneys-MacBook-Pro:TestApp courtneythaden$ phonegap run ios
[phonegap] detecting iOS SDK environment...
[phonegap] using the local environment
[phonegap] compiling iOS...
[phonegap] successfully compiled iOS app
[phonegap] trying to install app onto device
[phonegap] no device was found
[phonegap] trying to install app onto emulator
[phonegap] successfully installed onto emulator
```

Figure 25. PhoneGap iOS Build Terminal Output Screenshot.

The PhoneGap projects defaulted to scrollable. The result of running the TestApp

application as it appears on the iPhone simulator as an iPhone Retina (4.5-inch) can be

seen in Figures 26 and 27.

PhoneGap was based on Apache's Cordova™ program (PhoneGap, 2014c) and

the documentation was confusing due to the fact that some of it still included references

only to Cordova and the rest contained a mixture of Cordova and PhoneGap. This issue

led to the PhoneGapTestApp application code being added to the project, to not include

PhoneGap or Cordova specific code. Anything added to the project included HTML5,

JavaScript, and CSS code found in some of the other multi-platform development

applications test applications. It was determined that most developers searching for a

multi-platform development application would not have wasted any more of their time

with PhoneGap, due to the major issues this development application created.
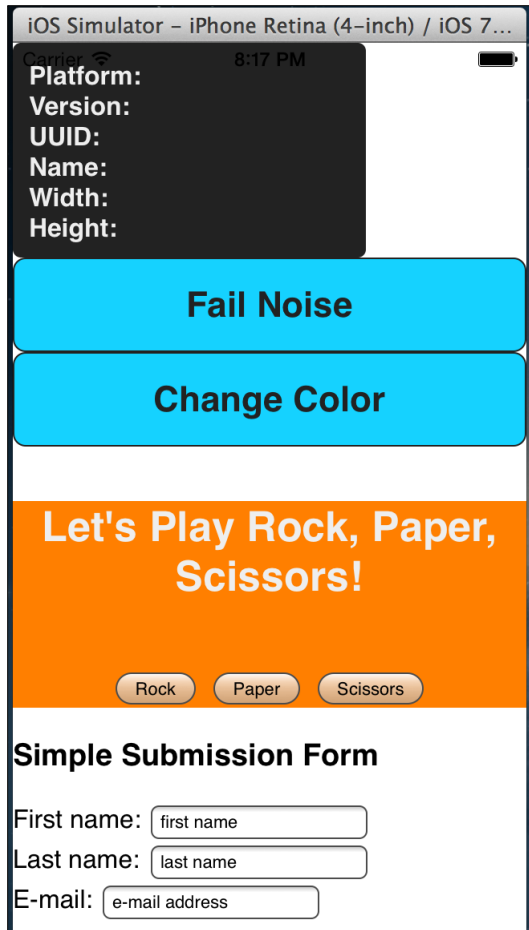
Figure 26. PhoneGap iOS TestApp Application View 1 Screenshot.

Figure 27. PhoneGap iOS TestApp Application View 2 Screenshot.

CHAPTER V

DISCUSSION

Summary

The purpose of this study was to investigate the multi-platform development applications, create a test application using each multi-platform development application, run the test application on the Android emulator and iOS simulator to determine performance, and determine which multi-platform application was best suited for allowing a developer to create a mobile application that could be utilized on a variety of platforms. Differences were investigated in regard to capabilities, features, ease of use, and functionality of each development application. The analysis of this study was determined using MoSync, Appcelerator, and PhoneGap.

MoSync

MoSync's SDK documentation with regard to the JavaScript side was lacking important details and therefore would cause a developer of any level to struggle. The lack of important details took away from the ability of a developer to completely utilize and customize the properties of components and objects within an application. MoSync SDK included an IDE that needed some updates to become more user-friendly and be of benefit to developers. Suggested updates could include but are not limited to code autocomplete and error detection within the code editor, error descriptions that appear on the emulator/simulator when there is an error running the application such as an

57

undefined variable, the ability to create web applications and native user interface applications within one project, and the launching of the iOS simulator working consistently. The IDE also included a built-in debugger. Parts of the native user interface test application did not work correctly which could be blamed on the lack of details in the documentation. The level of difficulty in using the MoSync SDK to build multi-platform applications using JavaScript was enough to make it not worth using unless the developer was very experienced. If improvements in documentation and the IDE were made, a novice developer could potentially create a native user interface multi-platform application with little to no problems. Although this study did not include using the MoSync SDK to create multi-platform applications using strictly C/C++, the level of complexity involved in C/C++ concepts would require the developer to be experienced in this language. MoSync also offered the MoSync Reload program, which had a smaller number of capabilities than MoSync SDK. Although this program was not used in this study, the documentation led me to believe it was geared toward novice developers.

<div align="center">

Appcelerator

</div>

Appcelerator Titanium had vast amounts of documentation for all levels of developers. Titanium included Titanium Studio which was the GUI and had the functionalities experienced developers appreciate and novice developers need. These functionalities included a built-in debugger, code autocomplete, error detection within the code editor, error descriptions that appear on the emulator/simulator when there was an error running the application, and native applications and web applications that could be built using one project. Titanium only used JavaScript and one single code base that still utilized platform differences and for these reasons a novice developer could easily use

Titanium. The test application worked as expected on both the iOS and Android

platforms. After initially setting up my user account within Appcelerator, I was leery

about using Titanium due to emails that included misspellings which when done in code

could cause annoying errors. After using Titanium, I would rank it number one among

the three multi-platform development applications and recommend it to any level of

developer.

PhoneGap

PhoneGap was based on Apache's Cordova™ program and the documentation

was confusing due to the fact that some of it still included references only to Cordova and

the rest contained a mixture of Cordova and PhoneGap. PhoneGap did not include a GUI

of any kind. It strictly used the command line interface. The documentation did include

PhoneGap specific commands and some basic commands. Minimal APIs were included

in PhoneGap but additional APIs needed for projects were included through plugins.

PhoneGap did include a debugger plugin, but encouraged developers to utilize debuggers

built in to the platform's native SDK. Web applications were the only type of application

that could be created, although PhoneGap offered plugins that could be used to store

information and files locally so the application could function even when a network

connection was not available. PhoneGap used HTML5, JavaScript, CSS, and XML. The

test application functionality on iOS did not work as expected although that was due to

PhoneGap specific code being omitted. PhoneGap did not recognize the Android SDK as

being installed even though all directions were followed to install the platform. This

resulted in the inability of running the test application in the Android emulator. Due to

the documentation problems, use of the command line interface, and problems installing

the Android platform a developer would need to be experienced to build applications using PhoneGap. However if the problems were fixed and the documentation included more of the basic commands used for the command line interface, a relatively inexperienced developer could potentially be successful creating a multi-platform web application.

Limitations

Limitations of this study included three free popular multi-platform development applications that used web development languages and the lack of publicly available standards for developing and testing mobile applications (Dye & Scarfone, 2014). This study did not encompass multi-platform development applications that were fee based; used languages other than web development languages or those less popular. The first publicly available development and testing mobile application standard was published by the United States Department of Defense in an attempt to reduce security vulnerabilities that could be found in mobile application code, input handling, initialization, termination, and external code (Dye & Scarfone, 2014).

Conclusions and Recommendations

Results of this study provided an understanding of how the standard web development languages HTML5, JavaScript, and CSS could be used in MoSync, Appcelerator, and PhoneGap to create multi-platform applications. The determination was made that Appcelerator's capabilities, features, ease of use, and functionalities outperformed MoSync and PhoneGap. Appcelerator would be the easiest multi-platform development application for a novice developer to use in the creation of a multi-platform

application and an experienced developer to gain the most out of their multi-platform application.

Open systems have caused problems for multi-platform application developers by allowing customization of any elements of a platform including but not limited to device sizes, objects, properties, native interfaces, and APIs (Abolfazil, Sanaei, Xia, &Yang, 2014). Multi-platform development applications cannot even come close to possibly including every implementation of an element within an open system. The discipline of technology in mobile device applications is growing steadily (Barmpatsalou, Damopoulos, Kambourakis, & Katos, 2013). Growing pains are being experienced in standardization efforts of mobile application terminology, development, and testing due to the field's relative infancy (Barmpatsalou et al., 2013). Laying the foundation for these standards is most important for the future of application and multi-platform application development, testing, and further research within the field of mobile technology.

APPENDICES

Appendix A
MoSync HTML5/JS WebUI Template Project

```html
<!DOCTYPE html>
<!--
* @file index.html
*
* Template application that shows examples of how to access
* device services from JavaScript using the Wormhole library.
-->
<html>
  <head>
    <meta name="viewport" content="width=320, user-scalable=no">
    <meta http-equiv="Content-type" content="text/html; charset=utf-8">
    <title>Wormhole Template App</title>
    <link rel="stylesheet" href="style.css" type="text/css" media="screen"
    title="no title" charset="utf-8">
    <script type="text/javascript" charset="utf-8" src="js/wormhole.js"></script>
    <script type="text/javascript">
      /**
       * Displays the device information on the screen.
       */
      function displayDeviceInfo()
      {
        document.getElementById("platform").innerHTML = device.platform;
        document.getElementById("version").innerHTML = device.version;
        document.getElementById("uuid").innerHTML = device.uuid;
        document.getElementById("name").innerHTML = device.name;
        document.getElementById("width").innerHTML = screen.width;
        document.getElementById("height").innerHTML = screen.height;
      }

      /**
       * Vibrate device.
       */
      function vibrate()
      {
        navigator.notification.vibrate(500);
      }

      /**
       * Play one beep sound.
       */
      function beep()
      {
        navigator.notification.beep(1);
      }

      /**
       * Change page background to a random color.
       */
      function changeColor()
      {
        var color = "#" +
          (Math.random() * 0xFFFFFF + 0x1000000)
            .toString(16).substr(1,6);
        document.documentElement.style.backgroundColor = color;
        document.body.style.backgroundColor = color;
      }

      // Register event listeners.

      // The "deviceready" event is sent when the system
      // has finished loading.
      document.addEventListener(
        "deviceready",
        displayDeviceInfo,
        true);
```

Page: 1

63

```
      // Close the application when the back key is pressed.
      document.addEventListener(
        "backbutton",
        function() { mosync.app.exit(); },
        true);
   </script>
 </head>
 <body>
   <div id="screen">
     <div class="pane" id="heading">Customized Wormhole Technology</div>
     <div class="pane" id="info">
       <div class="infoItem">Platform: <span id="platform"> </span></div>
       <div class="infoItem">Version: <span id="version"> </span></div>
       <div class="infoItem">UUID: <span id="uuid"> </span></div>
       <div class="infoItem">Name: <span id="name"> </span></div>
       <div class="infoItem">Width: <span id="width"> </span></div>
       <div class="infoItem">Height: <span id="height"> </span></div>
     </div>
     <div class="pane button" onclick="vibrate()">Vibrate</div>
     <div class="pane button" onclick="beep()">Beep</div>
     <div class="pane button" onclick="changeColor()">Change Color</div>
   </div>
 </body>
</html>
```

64

```cpp
/**
 * @file main.cpp
 *
 * This file contains the support code at the C++ level for
 * an HTML5/JS application that has access to device services.
 *
 * You don't need to change anything in this code file unless
 * you wish to add support for functions that are not available
 * out-of-the box in wormhole.js.
 */

#include <Wormhole/HybridMoblet.h>

#include "MAHeaders.h" // Defines BEEP_WAV

// Namespaces we want to access.
using namespace NativeUI; // WebView widget.
using namespace Wormhole; // Wormhole library.

/**
 * The application class.
 */
class MyMoblet : public HybridMoblet
{
public:
  MyMoblet()
  {
    // Show the start page.
    showPage("index.html");

    // Set the sound used by the PhoneGap beep notification API.
    // BEEP_WAV is defined in file Resources/Resources.lst.
    setBeepSound(BEEP_WAV);
  }
};

/**
 * Main function that is called when the program starts.
 * Here an instance of the MyMoblet class is created and
 * the program enters the main event loop.
 */
extern "C" int MAMain()
{
  (new MyMoblet())->enterEventLoop();
  return 0;
}
```

```css
/*
 * @file style.css
 */
html, body {
  margin: 0;
  padding: 0;
  width: 100%;
  height: 100%;
  background-color: #FFFFFF;
  font-family: sans-serif;
  text-decoration: none;

  /* Disable text selection. */
  -webkit-touch-callout: none;
  -webkit-user-select: none;
  -khtml-user-select: none;
  -moz-user-select: none;
  -ms-user-select: none;
  -o-user-select: none;
  user-select: none;
}

.pane {
  display: block;
  margin: 10px auto 10px auto;
  border: 0px solid #EEEEEE;
  -webkit-border-radius: 5px;
  border-radius: 5px;
  clear: both;
  width: 280px;
  padding: 10px 10px 10px 10px;
}

#heading {
  background: #FF7F00; /*#FF7744;*/
  color: #EEEEEE;
  font-size: 1.6em;
  font-weight: bold;
  text-align: center;
  margin-top: 15px;
}

#info {
  background: #222222;
  line-height: 1.5em;
}

.infoItem {
  font-size: 0.9em;
  font-weight: bold;
  color: #EEEEEE;
}

.button {
  border: 1px solid #222222;
  text-align: center;
  background: #15D2FF;
  color: #222222;
  font-size: 1.4em;
  font-weight: bold;
  -webkit-border-radius: 8px;
  border-radius: 8px;
  padding: 15px 10px 15px 10px;
  cursor: pointer;
}
```

66

```html
<!DOCTYPE html>
<!--
* file index.html
* Courtney Thaden
* 2013-2014
-->
<html>
  <head>
    <meta name="viewport" content="width=320, user-scalable=no">
    <meta http-equiv="Content-type" content="text/html; charset=utf-8">
    <title>Test App</title>
    <link rel="stylesheet" href="style.css" type="text/css" media="screen" title="no
title" charset="utf-8">
    <script type="text/javascript" charset="utf-8" src="js/wormhole.js"></script>
    <script type="text/javascript" src="js/script.js"></script>
    <script type="text/javascript">
      /**
       * Displays the device information on the screen.
       */
      function displayDeviceInfo()
      {
        document.getElementById("platform").innerHTML = device.platform;
        document.getElementById("version").innerHTML = device.version;
        document.getElementById("uuid").innerHTML = device.uuid;
        document.getElementById("name").innerHTML = device.name;
        document.getElementById("width").innerHTML = screen.width;
        document.getElementById("height").innerHTML = screen.height;
      }

      /**
       * Play one beep sound with beep sound being the fail.mp3.
       */
      function beep()
      {
        navigator.notification.beep(1);
      }

      /**
       * Change page background to a random color.
       */
      function changeColor()
      {
        var color = "#" +
          (Math.random() * 0xFFFFFF + 0x1000000)
            .toString(16).substr(1,6);
        document.documentElement.style.backgroundColor = color;
        document.body.style.backgroundColor = color;
      }



      // Register event listeners.

      // The "deviceready" event is sent when the system
      // has finished loading.
      document.addEventListener(
        "deviceready",
        displayDeviceInfo,
        true);

      // Close the application when the back key is pressed.
      document.addEventListener(
        "backbutton",
        function() { mosync.app.exit(); },
        true);
```

```
      </script>

  </head>
  <body>
    <div id="screen">

      <div class="pane" id="info">
        <div class="infoItem">Platform: <span id="platform"> </span></div>
        <div class="infoItem">Version: <span id="version"> </span></div>
        <div class="infoItem">UUID: <span id="uuid"> </span></div>
        <div class="infoItem">Name: <span id="name"> </span></div>
        <div class="infoItem">Width: <span id="width"> </span></div>
        <div class="infoItem">Height: <span id="height"> </span></div>
      </div>
      <div class="button" onclick="beep()">Fail Noise</div>
      <div class="button" onclick="changeColor()">Change Color</div>

          <div id="divider">

      <h4>Let's Play Rock, Paper, Scissors!</h4>
      <input type="button" onclick="compare(0)" value="Rock"/>
      <input type="button" onclick="compare(1)" value="Paper"/>
      <input type="button" onclick="compare(2)" value="Scissors"/>
      </div>


      <div id="header"><h3>Simple Submission Form</h3></div>
          <div id="inputs">
              <form name="input" action="#" method="post">
                First name: <input type="text" name="firstname" value="first
                name"><br/>
                Last name: <input type="text" name="lastname" value="last name"><br/>
                E-mail: <input type="email" name="email" value="e-mail
                address"><br/><br/>
              </form>
          </div>
          <div>
              <form id="options">
                  Sex:
                  <input type="radio" name="sex" value="male">Male
                  <input type="radio" name="sex" value="female">Female<br/><br/>
                  Major (check all that apply): <br>
                  <input type="checkbox" name="notify" value="ChemE">Chemical
                  Engineering<br/>
                  <input type="checkbox" name="notify" value="CE">Civil
                  Engineering<br/>
                  <input type="checkbox" name="notify" value="EE">Electrical
                  Engineering<br/>
                  <input type="checkbox" name="notify" value="ME">Mechanical
                  Engineering<br/>


                  Comments: <br/><textarea name="msg"></textarea><br/>
                  <input type="submit" value="Submit">
              </form>
          </div>
    </div>
  </body>
</html>
```

68

```
/**
 * file main.cpp
 * Courtney Thaden
 * 2013-2014
 *
 *
 * This file contains the support code at the C++ level for
 * an HTML5/JS application that has access to device services.
 *
 *
 */

#include <Wormhole/HybridMoblet.h>

#include "MAHeaders.h" // Defines BEEP_WAV

// Namespaces we want to access.
using namespace NativeUI; // WebView widget.
using namespace Wormhole; // Wormhole library.

/**
 * The application class.
 */
class MyMoblet : public HybridMoblet
{
public:
  MyMoblet()
  {
    // Show the start page.
    showPage("index.html");

    // Set the sound used by the PhoneGap beep notification API
    setBeepSound(FAIL_MP3);
  }
};

/**
 * Main function that is called when the program starts.
 * Here an instance of the MyMoblet class is created and
 * the program enters the main event loop.
 */
extern "C" int MAMain()
{
  (new MyMoblet())->enterEventLoop();
  return 0;
}
```

69

```
/*
 * file script.js
 * Courtney Thaden
 * 2013-2014
 *
 */

    function compare(index) {
      var rps = ["rock", "paper", "scissors"];

      var rules = {
          rock: {
            scissors: "breaks"
          },
          paper: {
            rock: "covers",
          },
          scissors: {
            paper: "cuts"
          }};
      var userChoice = rps[index];
          var computerChoice = rps[Math.floor(Math.random() * 5)];

          if (userChoice !== computerChoice){
            if (rules[userChoice].hasOwnProperty(computerChoice))
              alert("You win - " + userChoice + " " +
              rules[userChoice][computerChoice] +
                  " " + computerChoice + ".");
            else
              alert("You lose - " + computerChoice + " " +
              rules[computerChoice][userChoice] +
                  " " + userChoice + ".");
          }
          else
            alert("Draw. We both played " + userChoice + ".");
      };
```

70

```css
/*
 * file style.css
 * Courtney Thaden
 * 2013-2014
 */
html, body {
  margin: 0;
  padding: 0;
  width: 100%;
  height: 100%;
  background-color: #FFFFFF;
  font-family: sans-serif;
  text-decoration: none;

  /* Disable text selection. */
  -webkit-touch-callout: none;
  -webkit-user-select: none;
  -khtml-user-select: none;
  -moz-user-select: none;
  -ms-user-select: none;
  -o-user-select: none;
  user-select: none;
}
.pane {
  border: 0px solid #EEEEEE;
  -webkit-border-radius: 5px;
  border-radius: 5px;
  clear: both;
  width: 200px;
  padding: 10px;
}

#info {
  background: #222222;
  line-height: 1.2em;
}

.infoItem {
  font-size: 1em;
  font-weight: bold;
  color: #EEEEEE;
}

.button {
  border: 1px solid #222222;
  text-align: center;
  background: #15D2FF;
  color: #222222;
  font-size: 1.4em;
  font-weight: bold;
  -webkit-border-radius: 8px;
  border-radius: 8px;
  padding: 15px 10px 15px 10px;
  cursor: pointer;
  width: auto;
}

#divider {
    background: #FF7F00;
  color: #EEEEEE;
  font-size: 1.6em;
  font-weight: bold;
  text-align: center;
  margin-top: 10px;
}
```

```html
<!DOCTYPE html>
<!--
* @file index.html
*
* Template application that provides Native UI functionality in
* HTML5 and JavaScript.
-->
<html>
<head>
  <meta http-equiv="Content-type" content="text/html; charset=utf-8">
  <script type="text/javascript" charset="utf-8" src="js/wormhole.js"></script>
  <script type="text/javascript">
    /**
     * Called by the NativeUI library when the UI is ready to
     * be shown. Show the main screen here.
     *
     * Here we illustrate how to add events to widgets and how to
     * create widgets from JavaScript code.
     */
    mosync.nativeui.UIReady = function()
    {
      // First get the screen we want to show.
      var mainScreen = document.getNativeElementById("mainScreen");

      // Show the screen.
      mainScreen.show();

      // Get an instacne of the vibrate button created in the markup.
      var vibrateButton = document.getNativeElementById("vibrateButton");

      // Add an event listener to it. This is an alternative to
      // specify a function name in the markup.
      vibrateButton.addEventListener("Clicked", function()
      {
        navigator.notification.vibrate(1000);
      });

      // Create a button in JavaScript.
      var helloButton = mosync.nativeui.create("Button", "helloButton",
      {
        // Declarative way of setting properties.
        "width": "FILL_AVAILABLE_SPACE",
        "text": "Say Hello"
      });

      // Here is how to set properties in code.
      helloButton.setProperty("fontSize", "20");

      // Setting the clicked function.
      helloButton.addEventListener("Clicked", function()
      {
        helloButton.setProperty("text",  "Hello World!");
      });

      // Add button to layout.
      helloButton.addTo("mainLayout");
    }

    /**
     * Number of clicks on counterButton.
     */
    var clickCounter = 0;

    /**
     * Called when counterButton is clicked.
```

```
  */
  function counterButtonClicked()
  {
    var label = document.getNativeElementById("textLabel");
    ++clickCounter;
    label.setProperty("text", "No. of clicks: " + clickCounter);
  }

  // Register event listeners.

  // The "deviceready" event is sent when the system
  // has finished loading. Here we initialise the UI.
  document.addEventListener(
    "deviceready",
    function()
    {
      // Will call mosync.nativeui.UIReady when
      // all Native UI widgets have been created.
      mosync.nativeui.initUI();
    },
    true);

  // Close the application when the back key is pressed.
  document.addEventListener(
    "backbutton",
    function()
    {
      mosync.app.exit();
    },
    true);
</script>
</head>

<body>
  <!-- All of the mosync.nativeui widgets should be wrapped inside a
    tag with id="NativeUI" -->
  <div id="NativeUI">
    <!-- The element with id="mainScreen" contains the main UI screen.
      In this app the main screen has two tabs with screens. -->
    <div data-widgetType="TabScreen" id="mainScreen">
      <!-- First Screen -->
      <div data-widgetType="Screen"
        id="widgetScreen"
        data-title="Widgets"
        data-icon_android="img/Android_TabIconDevice.png"
        data-icon_iOS="img/IOS_TabIconDevice.png">
        <div data-widgetType="VerticalLayout"
          id="mainLayout"
          data-width="FILL_AVAILABLE_SPACE"
          data-height="FILL_AVAILABLE_SPACE">
          <div widgetType="Label"
            id="textLabel"
            data-width="FILL_AVAILABLE_SPACE"
            data-text="Native UI in JavaScript"
            data-fontSize="26">
          </div>
          <div data-widgetType="Button"
            id="counterButton"
            data-width="FILL_AVAILABLE_SPACE"
            data-text="Click Me!"
            data-fontSize="20"
            data-onevent="counterButtonClicked()">
          </div>
          <div data-widgetType="Button"
            id="vibrateButton"
```

73

```
                    data-width="FILL_AVAILABLE_SPACE"
                    data-text="Vibrate"
                    data-fontSize="20">
              </div>
          </div>
      </div>
      <!-- Second Screen -->
      <div data-widgetType="Screen"
          data-id="webScreen"
          data-title="Web"
          data-icon_android="img/Android_TabIconWebView.png"
          data-icon_iOS="img/IOS_TabIconWebView.png">
          <div data-widgetType="WebView"
              id="webBrowser"
              data-width="FILL_AVAILABLE_SPACE"
              data-height="FILL_AVAILABLE_SPACE"
              data-url="http://www.google.com">
          </div>
      </div>
    </div>
  </div>
</body>
</html>
```

74

```cpp
/**
 * @file main.cpp
 *
 * This file contains the support code at the C++ level for
 * an HTML5/JS application that uses NativeUI and has access
 * to device services.
 *
 * You don't need to change anything in this code file unless
 * you wish to add support for functions that are not available
 * out-of-the box in wormhole.js.
 */

#include <Wormhole/HybridMoblet.h>

#include "MAHeaders.h" // Defines BEEP_WAV

// Namespaces we want to access.
using namespace NativeUI; // WebView widget.
using namespace Wormhole; // Wormhole library.

/**
 * The application class.
 */
class MyMoblet : public HybridMoblet
{
public:
  MyMoblet()
  {
    // Show the page with NativeUI definitions.
    showNativeUI("index.html");

    // Set the sound used by the PhoneGap beep notification API.
    // BEEP_WAV is defined in file Resources/Resources.lst.
    setBeepSound(BEEP_WAV);
  }
};

/**
 * Main function that is called when the program starts.
 * Here an instance of the MyMoblet class is created and
 * the program enters the main event loop.
 */
extern "C" int MAMain()
{
  (new MyMoblet())->enterEventLoop();
  return 0;
}
```

```
<!DOCTYPE html>
<!--
* file index.html
* Courtney Thaden
* 2014
*
-->
<html>
<head>
  <meta http-equiv="Content-type" content="text/html; charset=utf-8">
  <script type="text/javascript" charset="utf-8" src="js/wormhole.js"></script>
  <script type="text/javascript" src="js/script.js"></script>
  <script type="text/javascript">
    /**
     * Called by the NativeUI library when the UI is ready to
     * be shown. Show the main screen here.
     *
     * Here we illustrate how to add events to widgets and how to
     * create widgets from JavaScript code.
     */
    mosync.nativeui.UIReady = function()
    {
      // First get the screen we want to show.
      var mainScreen = document.getNativeElementById("mainScreen");

      // Show the screen.
      mainScreen.show();

      showDeviceInfo();

    }



    /**
     * Displays the device information on the screen.
     */
    function showDeviceInfo()
    {
      var platform = document.getNativeElementById("PlatformLabel");
      platform.setProperty("text" , "Platform: " + device.platform);

      var version = document.getNativeElementById("VersionLabel");
      version.setProperty("text" , "Version: " + device.version);

      var uuid = document.getNativeElementById("UUIDLabel");
      uuid.setProperty("text" , "UUID: " + device.uuid);

      var deviceName = document.getNativeElementById("NameLabel");
      deviceName.setProperty("text" , "Name: " + device.name);

      var screenWidth = document.getNativeElementById("ScreenWidthLabel");
      screenWidth.setProperty("text" , "Screen Width: " + screen.width);

      var screenHeight = document.getNativeElementById("ScreenHeightLabel");
      screenHeight.setProperty("text" , "Screen Width: " + screen.height);
    }



    /**
     * Play one sound.
     */
    function noiseButtonClicked()
    {
```

76

```
  navigator.notification.beep(1);
}

/**
 * Change page background to a random color.
 */
function colorButtonClicked()
{
  var layout = document.getNativeElementById("mainLayout");
  var color = "#" +
    (Math.random() * 0xFFFFFF + 0x1000000)
      .toString(16).substr(1,6);
  layout.setProperty("backgroundColor", color);
}

function compare(index)
    {
  var rps = ["rock", "paper", "scissors"];
  var rules = {
      rock: {
        scissors: "breaks"
      },
      paper: {
        rock: "covers",
      },
      scissors: {
        paper: "cuts"
      }};
  var userChoice = rps[index];
        var computerChoice = rps[Math.floor(Math.random() * 5)];

        if (userChoice !== computerChoice){
          if (rules[userChoice].hasOwnProperty(computerChoice))
            alert("You win - " + userChoice + " " +
            rules[userChoice][computerChoice] +
                " " + computerChoice + ".");
          else
            alert("You lose - " + computerChoice + " " +
            rules[computerChoice][userChoice] +
                " " + userChoice + ".");
        }
        else
          alert("Draw. We both played " + userChoice + ".");
};


function rockButtonClicked()
{
  compare(0);
}

  function paperButtonClicked()
{
  compare(1);
}

function scissorsButtonClicked()
{
  compare(2);
}
```

```
/*    function resetButtonClicked()
    {
      var JackImage = mosync.nativeui.create("Image", "JackImage",
            {
                // Declarative way of setting properties.
                "width": "100",
                "height": "200",
                "top": "460",
                "left": "325",
                "image":"img/monkey.jpg",
                "scaleMode":"scalePreserveAspect"
            });
    }

*/
    // Register event listeners.
    // The "deviceready" event is sent when the system
    // has finished loading. Here we initialise the UI.
    document.addEventListener(
      "deviceready",
      function()
      {
        // Will call mosync.nativeui.UIReady when
        // all Native UI widgets have been created.
        mosync.nativeui.initUI();
      },
      true);



    // Close the application when the back key is pressed.
    document.addEventListener(
      "backbutton",
      function()
      {
        mosync.app.exit();
      },
      true);



  </script>
</head>

<body>

  <!-- All of the mosync.nativeui widgets should be wrapped inside a
    tag with id="NativeUI" -->
  <div id="NativeUI">
    <!-- The element with id="mainScreen" contains the main UI screen.
      In this app the main screen has two tabs with screens. -->
    <div data-widgetType="TabScreen" id="mainScreen">
      <!-- First Screen -->
      <div data-widgetType="Screen"
        id="widgetScreen"
        data-icon_android="img/Android_TabIconDevice.png"
        data-icon_iOS="img/IOS_TabIconDevice.png">
        <div data-widgetType="VerticalLayout"
          id="mainLayout"
          data-width="FILL_AVAILABLE_SPACE"
          data-height="FILL_AVAILABLE_SPACE"
          data-backgroundColor="#FFFFFF" >

            <div widgetType="Label"
```

78

```
      id="PlatformLabel"
      data-width="FILL_AVAILABLE_SPACE"
      data-fontSize="24"
      data-text="Platform:"
      data-fontColor="#FFFFFF"
      data-backgroundColor="#222222"
      data-textHorizontalAlignment="left">
  </div>
  <div data-widgetType="Label"
      id="VersionLabel"
      data-width="FILL_AVAILABLE_SPACE"
      data-fontSize="24"
      data-text="Version:"
      data-fontColor="#FFFFFF"
      data-backgroundColor="#222222"
      data-textHorizontalAlignment="left">
  </div>
  <div data-widgetType="Label"
      id="UUIDLabel"
      data-width="FILL_AVAILABLE_SPACE"
      data-fontSize="24"
      data-text="UUID:"
      data-fontColor="#FFFFFF"
      data-backgroundColor="#222222"
      data-textHorizontalAlignment="left">
  </div>
  <div data-widgetType="Label"
      id="NameLabel"
      data-width="FILL_AVAILABLE_SPACE"
      data-fontSize="24"
      data-text="Name:"
      data-fontColor="#FFFFFF"
      data-backgroundColor="#222222"
      data-textHorizontalAlignment="left">
  </div>
      <div data-widgetType="Label"
      id="ScreenWidthLabel"
      data-width="FILL_AVAILABLE_SPACE"
      data-fontSize="24"
      data-text=" Screen Width:"
      data-fontColor="#FFFFFF"
      data-backgroundColor="#222222"
      data-textHorizontalAlignment="left">
  </div>
  <div data-widgetType="Label"
      id="ScreenHeightLabel"
      data-width="FILL_AVAILABLE_SPACE"
      data-fontSize="24"
      data-text=" Screen Height:"
      data-fontColor="#FFFFFF"
      data-backgroundColor="#222222"
      data-textHorizontalAlignment="left">
  </div>

  <div data-widgetType="Button"
  id="noiseButton"
  data-width="FILL_AVAILABLE_SPACE"
  data-text="Fail Noise"
  data-fontSize="20"
  data-backgroundColor="#15D2FF"
              data-fontColor= "#222222"
  data-onevent="noiseButtonClicked()">
</div>
  <div data-widgetType="Button"
  id="colorButton"
```

```
        data-width="FILL_AVAILABLE_SPACE"
        data-text="Change Color"
        data-fontSize="20"
        data-backgroundColor="#15D2FF"
                    data-fontColor= "#222222"
        data-onevent="colorButtonClicked()">
    </div>
</div>
  <div data-widgetType="HorizontalLayout"
  id="RPSLayout"
  data-width="FILL_AVAILABLE_SPACE"
  data-height="150"
  data-top= "300"
  data-backgroundColor="#FF7F00" >
  </div>
      <div widgetType="Label"
      id= "RPSLabel"
      data-fontSize= "24"
          data-fontColor= "EEEEEE"
        data-top= "310"
        data-text= "Lets Play Rock, Paper, Scissors!"
        data-left= "50" >
      </div>
      <div data-widgetType="Button"
  id="rockButton"
  data-width="75"
  data-top="375"
  data-left="50"
  data-text="Rock"
  data-fontSize="20"
  data-backgroundColor="#808080"
                data-fontColor="#222222"
  data-onevent="rockButtonClicked()">
   </div>
      <div data-widgetType="Button"
  id="paperButton"
  data-width="90"
  data-top="375"
  data-left="180"
  data-text="Paper"
  data-fontSize="20"
  data-backgroundColor="#808080"
                data-fontColor="#222222"
  data-onevent="paperButtonClicked()">
   </div>

        <div data-widgetType="Button"
  id="scissorsButton"
  data-width="105"
  data-top="375"
  data-left="325"
  data-text="Scissors"
  data-fontSize="20"
  data-backgroundColor="#808080"
                data-fontColor="#222222"
  data-onevent="scissorsButtonClicked()">
   </div>

        <div widgetType="Label"
        id="JackLabel"
        data-width="FILL_AVAILABLE_SPACE"
        data-fontSize="24"
        data-text="Choose Jack's Outfit"
        data-fontColor="#222222"
        data-top="460"
```

80

```
            data-textHorizontalAlignment="left">
        </div>
        <div data-widgetType="Image"
    id="JackImage"
    data-width="100"
    data-height="200"
    data-top= "460"
    data-left= "325"
    data-image="/img/monkey.jpg"
    data-scaleMode="scalePreserveAspect">
    </div>
    <div data-widgetType="Image"
     id="tuxImage"
     data-width="100"
     data-height="120"
     data-top= "504"
     data-left= "323"
     data-image= "/img/tux.png"
     data-scaleMode="scalePreserveAspect">
    </div>

        <div data-widgetType="Image"
     id="hatImage"
     data-width="25"
     data-height="35"
     data-top= "446"
     data-left= "363"
     data-image="/img/partyhat.png"
     data-scaleMode="scalePreserveAspect">
    </div>

    <div data-widgetType="Image"
     id="tieImage"
     data-width="28"
     data-height="12"
     data-top= "510"
     data-left= "360"
     data-image="/img/bowtie.jpg"
     data-scaleMode="scaleXY">
    </div>

iv data-widgetType="Button"
    id="tuxButton"
    data-width="100"
    data-height="40"
    data-top= "500"
    data-left= "10"
    data-text="Party Suit"
    data-fontSize="16"
    data-backgroundColor="#FF99CC"
        data-fontColor= "#222222"
    data-onevent="tuxButtonClicked()">
      </div>
      <div data-widgetType="Button"
    id="hatButton"
    data-width="95"
    data-height="40"
    data-top= "550"
    data-left= "10"
    data-text="Party Hat"
    data-fontSize="16"
    data-backgroundColor="#FF99CC"
        data-fontColor= "#222222"
    data-onevent="hatButtonClicked()">
      </div>
```

81

```
        <div data-widgetType="Button"
    id="tieButton"
    data-width="90"
    data-height="40"
    data-top= "600"
    data-left= "10"
    data-text="Bow Tie"
    data-fontSize="16"
    data-backgroundColor="#FF99CC"
            data-fontColor= "#222222"
    data-onevent="tieButtonClicked()">
        </div>


        <div data-widgetType="Button"
    id="resetButton"
    data-width="70"
    data-height="40"
    data-top= "650"
    data-left= "10"
    data-text="Reset"
    data-fontSize="16"
    data-backgroundColor="#FF99CC"
                data-fontColor= "#222222"
    data-onevent="resetButtonClicked()">

        </div>
    </div>
    </div>
    </div>
</body>
</html>
```

82

```
/**
 * file main.cpp
 * Courtney Thaden
 * 2013-2014
 */

#include <Wormhole/HybridMoblet.h>

#include "MAHeaders.h" // Defines BEEP_WAV

// Namespaces we want to access.
using namespace NativeUI; // WebView widget.
using namespace Wormhole; // Wormhole library.

/**
 * The application class.
 */
class MyMoblet : public HybridMoblet
{
public:
  MyMoblet()
  {
    // Show the page with NativeUI definitions.
    showNativeUI("index.html");


    // FAIL_MP3 is defined in file Resources/Resources.lst.
    setBeepSound(FAIL_MP3);
  }
};

/**
 * Main function that is called when the program starts.
 * Here an instance of the MyMoblet class is created and
 * the program enters the main event loop.
 */
extern "C" int MAMain()
{
  (new MyMoblet())->enterEventLoop();
  return 0;
}
```

83

```
/*
 * file app.js
 * Courtney Thaden
 * 2013-2014
 */

//bootstrap and check dependencies
if (Ti.version < 1.8 ) {
  alert('Sorry - this application template requires Titanium Mobile SDK 1.8 or
  later');
}

// This is a single context application with multiple windows in a stack
(function() {
  //render appropriate components based on the platform and form factor
  var osname = Ti.Platform.osname,
    version = Ti.Platform.version,
    height = Ti.Platform.displayCaps.platformHeight,
    width = Ti.Platform.displayCaps.platformWidth;

  //considering tablet to have one dimension over 900px - this is imperfect, so you
  should feel free to decide
  //yourself what you consider a tablet form factor for android
  var isTablet = osname === 'ipad' || (osname === 'android' && (width > 899 || height
  > 899));

  var Window;
  if (isTablet) {
    Window = require('ui/tablet/ApplicationWindow');
  }
  else {
    // Android uses platform-specific properties to create windows.
    // All other platforms follow a similar UI pattern.
    if (osname === 'android') {
      Window = require('ui/handheld/ApplicationWindow');
    }
    else {
      Window = require('ui/handheld/ApplicationWindow');
    }
  }
  new Window().open();
})();
```

```
/*
 * file /handheld/android/ApplicationView.js
 * Courtney Thaden
 * 2013-2014
 */
function ApplicationWindow() {
  //load component dependencies
  var FirstView = require('ui/common/FirstView');

  //create component instance
  var self = Ti.UI.createWindow({
    backgroundColor:'#ffffff',
    navBarHidden:true,
    exitOnClose:true
  });

  //construct UI
  var firstView = new FirstView();
  self.add(firstView);

  return self;
}

//make constructor function the public component interface
module.exports = ApplicationWindow;
```

85

```
/*
 * file /handheld/ApplicationWindow.js
 * Courtney Thaden
 * 2013-2014
 */

function ApplicationWindow()
{
  //load component dependencies
  var FirstView = require('ui/common/FirstView');
  var SecondView = require('ui/common/SecondView');
  var ThirdView = require('ui/common/ThirdView');
  var FourthView = require('ui/common/FourthView');
  var FifthView = require('ui/common/FifthView');




// Create main application window
    var testapp = Ti.UI.createWindow(
  {
    backgroundColor: '#FFF',
    layout:'vertical',
  });


// Platform info
  var fourthView = FourthView();
  testapp.add(fourthView);

// Play sound
  var secondView = new SecondView();
  testapp.add(secondView);

// Change color
  var firstView = new FirstView();
  testapp.add(firstView);

    Ti.App.addEventListener('changeBG', function(e)
    {
     testapp.backgroundColor = e.newcolor;
    });

// Rock, Paper, Scissors Game
  var thirdView = new ThirdView();
  testapp.add(thirdView);

// Dress Jack the monkey
  var fifthView = new FifthView();
  testapp.add(fifthView);


  return testapp;
}

//make constructor function the public component interface
module.exports = ApplicationWindow;
```

86

```
/*
 * file FifthView.js
 * Courtney Thaden
 * 2013-2014
 */

function FifthView() {

  var self= Ti.UI.createView({
    height: 200,
    borderColor: 'gray',
      borderRadius: 8
  });



    var accTitle = Ti.UI.createLabel({
    text:"Choose Jack's Outfit",
    fontSize:12,
    color:'black',
    left:10,
    top:5,
    });
    self.add(accTitle);



    var apeImage = Ti.UI.createLabel({
    width:100,
    height:200,
    backgroundImage:'images/monkey.jpg',
    right:10,
    top:5,
    });
    self.add(apeImage);



  var tuxHolder = Ti.UI.createLabel({
      right:35,
    width:52,
    height:101,
      top:51
  });
  self.add(tuxHolder);

  var hatHolder = Ti.UI.createLabel({
      right:50,
    width:15,
    height:25,
      top:2
  });
  self.add(hatHolder);

  var tieHolder = Ti.UI.createLabel({
      right:50,
    width:20,
    height:5,
      top:55
  });
  self.add(tieHolder);

    var tuxButton = Titanium.UI.createButton({
      title: "Party Suit",
      color: 'black',
```

```
        left: 10,
        top:40,
        width: 110,
        fontSize:10,
        borderColor:'black',
        borderRadius:8,
        backgroundColor:'#FF99CC'
});
tuxButton.addEventListener('click', function()
{
    tuxHolder.setBackgroundImage('images/tux.png');
});
self.add(tuxButton);

    var hatButton = Titanium.UI.createButton({
        title: "Party Hat",
        color: 'black',
        left: 10,
        top: 80,
        width: 100,
        fontSize:10,
        borderColor:'black',
        borderRadius: 8,
        backgroundColor:'#FF99CC'
});
hatButton.addEventListener('click', function()
{
    hatHolder.setBackgroundImage('images/partyhat.png');
});
self.add(hatButton);

    var tieButton = Titanium.UI.createButton({
        title: "Bow Tie",
        color: 'black',
        left: 10,
        top: 120,
        width: 80,
        fontSize:10,
        borderColor:'black',
        borderRadius: 8,
        backgroundColor:'#FF99CC'
});
tieButton.addEventListener('click', function()
{
    tieHolder.setBackgroundImage('images/bowtie.jpg');
});
self.add(tieButton);

 var resetButton = Titanium.UI.createButton({
        title: "Reset",
        color: 'black',
        left: 10,
        top: 160,
        width: 60,
        fontSize:10,
        borderColor:'black',
        borderRadius: 8,
        backgroundColor:'#FF99CC'
});
resetButton.addEventListener('click', function()
{
    tuxHolder.setBackgroundImage('');
    hatHolder.setBackgroundImage('');
    tieHolder.setBackgroundImage('');
```

88

```
  });
  self.add(resetButton);

  return self;
}


module.exports = FifthView;
```

```
/*
 * file FirstView.js
 * Courtney Thaden
 * 2013-2014
 */
function FirstView() {


  var self= Ti.UI.createView({
    backgroundColor:'#15D2FF',
    height: 30,
    layout:'vertical',
    //top: 260,
    borderColor: 'gray',
      borderRadius: 8
  });


  var btn = Titanium.UI.createButton({
      title: "Change Color",
      color: 'black',
      touchEnabled: true,
      whichObj: 'Button'
  });


    btn.addEventListener('click', function(){
  Ti.API.info('Button');
  Ti.App.fireEvent('changeBG', {newcolor: Titanium.UI.setBackgroundColor('#' +
  (Math.random() * 0xFFFFFF + 0x1000000).toString(16).substr(1,6))});
});

self.add(btn);


  return self;
}




module.exports = FirstView;
```

```
/*
 * file FourthView.js
 * Courtney Thaden
 * 2013-2014
 */

function FourthView() {

    var self= Ti.UI.createView({
        backgroundColor:'black',
        height:180,
        layout: 'vertical',
        top: 30,
        left: 0,
        width: 250,
        borderColor: 'black',
          borderRadius: 8
    });


    var platLabel = Titanium.UI.createLabel({
        text:('Platform: ' + Titanium.Platform.osname) ,
        color:'white',
        left: 0,
        fontSize: 10,
        //top: 25
        //layout:'vertical'
    });
        self.add(platLabel);


        var versLabel = Titanium.UI.createLabel({
        text:('Version: ' + Titanium.Platform.version),
        color:'white',
        left: 0,
        fontSize: 10,
        //top: 50
        //layout:'vertical'
    });
        self.add(versLabel);

        var uuidLabel = Titanium.UI.createLabel({
        text:('UUID: ' + Titanium.Platform.id),
        color:'white',
        left: 0,
        fontSize: 10,
        //top: 75
        //layout:"vertical"
    });
        self.add(uuidLabel);

        var namLabel = Titanium.UI.createLabel({
        text:('Name: ' + Titanium.Platform.model),
        color:'white',
        left: 0,
        fontSize: 10,
        //top: 125
        //layout:'vertical'
    });
        self.add(namLabel);


        var widLabel = Titanium.UI.createLabel({
        text:('Width: ' + Titanium.UI.width),
```

```
        color:'white',
        left: 0,
        fontSize: 10,
        //top: 150
        //layout:'vertical'
    });
        self.add(widLabel);


        var heiLabel = Titanium.UI.createLabel({
        text:('Height: ' + Titanium.UI.height),
        color:'white',
        left: 0,
        fontSize: 10,
        //top: 175
        //layout:'vertical'
    });
        self.add(heiLabel);
    return self;
}


module.exports = FourthView;
```

```
/*
 * file rps.js
 * Courtney Thaden
 * 2013-2014
 */
var  compare =     function (index) {
     var rps = ["rock", "paper", "scissors"];

     var rules = {
         rock: {
           scissors: "breaks"
         },
         paper: {
           rock: "covers",
         },
         scissors: {
           paper: "cuts"
         }};
     var userChoice = rps[index];
         var computerChoice = rps[Math.floor(Math.random() * 5)];

         if (userChoice !== computerChoice){
           if (rules[userChoice].hasOwnProperty(computerChoice))
             alert("You win - " + userChoice + " " +
             rules[userChoice][computerChoice] +
                 " " + computerChoice + ".");
           else
             alert("You lose - " + computerChoice + " " +
             rules[computerChoice][userChoice] +
                 " " + userChoice + ".");
         }
         else
           alert("Draw. We both played " + userChoice + ".");
     };


   exports.compare = compare;
```

93

```
/*
 * file SecondView.js
 * Courtney Thaden
 * 2013-2014
 */
function SecondView() {

  var self= Ti.UI.createView({
    backgroundColor:'#15D2FF',
    height: 30,
    layout: 'vertical',
    //top: 200,
    borderColor: 'gray',
      borderRadius: 8
  });



    var sound = Titanium.Media.createSound();
      sound.url='/etc/fail.mp3';

  //
  // PLAY
  //
  var failnoise = Titanium.UI.createButton({
      title: "Fail Noise",
      color: 'black'
  });
  failnoise.addEventListener('click', function()
  {
    sound.play();
  });
  self.add(failnoise);



  return self;
}


module.exports = SecondView;
```

```
/*
 * file ThirdView.js
 * Courtney Thaden
 * 2013-2014
 */
function ThirdView() {

    var rps = require('ui/common/rps');

  var self= Ti.UI.createView({
    backgroundColor:'#FF7F00',
    height: 100,
    //layout:'vertical',
    //top: 350,
    borderColor: 'gray'
  });


  var aLabel = Titanium.UI.createLabel({
    text:"Lets Play Rock, Paper,\n Scissors!",
    color:'white',
    width:'auto',
    textAlign:'center',
    fontSize: 48,
    top: 10
    //layout:'vertical'
  });
    self.add(aLabel);

//Rock
  var rock = Titanium.UI.createButton({
    title: "Rock",
    height:20,
    width:75,
    left: 20,
    bottom: 10,
    backgroundColor: 'lightgray',
    borderColor: 'gray',
    color: 'black'
    });

  rock.addEventListener('click', function()
  {
    rps.compare(0);
  });
  self.add(rock);

//Paper
  var paper = Titanium.UI.createButton({
    title: "Paper",
    height:20,
    width:75,
    center: 0,
    bottom: 10,
    backgroundColor: 'lightgray',
    borderColor: 'gray',
    color: 'black'
  });

  paper.addEventListener('click', function()
  {
    rps.compare(1);
  });
  self.add(paper);
```

95

```
//Scissors
  var scissors = Titanium.UI.createButton({
      title: "Scissors",
      height:20,
      width:75,
      right: 20,
      bottom: 10,
      backgroundColor: 'lightgray',
      borderColor: 'gray',
      color: 'black'
  });

  scissors.addEventListener('click', function()
  {
    rps.compare(2);
  });
  self.add(scissors);

  return self;
}


module.exports = ThirdView;
```

96

Appendix F
Titanium Original app.js File

```
/*
 * Single Window Application Template:
 * A basic starting point for your application.  Mostly a blank canvas.
 *
 * In app.js, we generally take care of a few things:
 * - Bootstrap the application with any data we need
 * - Check for dependencies like device type, platform version or network connection
 * - Require and open our top-level UI component
 *
 */

//bootstrap and check dependencies
if (Ti.version < 1.8 ) {
  alert('Sorry - this application template requires Titanium Mobile SDK 1.8 or
  later');
}

// This is a single context application with multiple windows in a stack
(function() {
  //render appropriate components based on the platform and form factor
  var osname = Ti.Platform.osname,
    version = Ti.Platform.version,
    height = Ti.Platform.displayCaps.platformHeight,
    width = Ti.Platform.displayCaps.platformWidth;

  //considering tablet to have one dimension over 900px - this is imperfect, so you
  should feel free to decide
  //yourself what you consider a tablet form factor for android
  var isTablet = osname === 'ipad' || (osname === 'android' && (width > 899 || height
  > 899));

  var Window;
  if (isTablet) {
    Window = require('ui/tablet/ApplicationWindow');
  }
  else {
    // Android uses platform-specific properties to create windows.
    // All other platforms follow a similar UI pattern.
    if (osname === 'android') {
      Window = require('ui/handheld/android/ApplicationWindow');
    }
    else {
      Window = require('ui/handheld/ApplicationWindow');
    }
  }
  new Window().open();
})();
```

```
<!--
file config.xml
-->
<?xml version='1.0' encoding='utf-8'?>
<widget id="com.example.hello" version="1.0.0" xmlns="http://www.w3.org/ns/widgets"
xmlns:gap="http://phonegap.com/ns/1.0">
    <name>HelloWorld</name>
    <description>
        Hello World sample application that responds to the deviceready event.
    </description>
    <author email="support@phonegap.com" href="http://phonegap.com">
        PhoneGap Team
    </author>
    <feature name="http://api.phonegap.com/1.0/device" />
    <preference name="permissions" value="none" />
    <preference name="orientation" value="default" />
    <preference name="target-device" value="universal" />
    <preference name="fullscreen" value="true" />
    <preference name="webviewbounce" value="true" />
    <preference name="prerendered-icon" value="true" />
    <preference name="stay-in-webview" value="false" />
    <preference name="ios-statusbarstyle" value="black-opaque" />
    <preference name="detect-data-types" value="true" />
    <preference name="exit-on-suspend" value="false" />
    <preference name="show-splash-screen-spinner" value="true" />
    <preference name="auto-hide-splash-screen" value="true" />
    <preference name="disable-cursor" value="false" />
    <preference name="android-minSdkVersion" value="7" />
    <preference name="android-installLocation" value="auto" />
    <icon src="icon.png" />
    <icon gap:density="ldpi" gap:platform="android"
src="res/icon/android/icon-36-ldpi.png" />
    <icon gap:density="mdpi" gap:platform="android"
src="res/icon/android/icon-48-mdpi.png" />
    <icon gap:density="hdpi" gap:platform="android"
src="res/icon/android/icon-72-hdpi.png" />
    <icon gap:density="xhdpi" gap:platform="android"
src="res/icon/android/icon-96-xhdpi.png" />
    <icon gap:platform="blackberry" src="res/icon/blackberry/icon-80.png" />
    <icon gap:platform="blackberry" gap:state="hover"
src="res/icon/blackberry/icon-80.png" />
    <icon gap:platform="ios" height="57" src="res/icon/ios/icon-57.png" width="57" />
    <icon gap:platform="ios" height="72" src="res/icon/ios/icon-72.png" width="72" />
    <icon gap:platform="ios" height="114" src="res/icon/ios/icon-57-2x.png"
width="114" />
    <icon gap:platform="ios" height="144" src="res/icon/ios/icon-72-2x.png"
width="144" />
    <icon gap:platform="webos" src="res/icon/webos/icon-64.png" />
    <icon gap:platform="winphone" src="res/icon/windows-phone/icon-48.png" />
    <icon gap:platform="winphone" gap:role="background"
src="res/icon/windows-phone/icon-173.png" />
    <gap:splash gap:density="ldpi" gap:platform="android"
src="res/screen/android/screen-ldpi-portrait.png" />
    <gap:splash gap:density="mdpi" gap:platform="android"
src="res/screen/android/screen-mdpi-portrait.png" />
    <gap:splash gap:density="hdpi" gap:platform="android"
src="res/screen/android/screen-hdpi-portrait.png" />
    <gap:splash gap:density="xhdpi" gap:platform="android"
src="res/screen/android/screen-xhdpi-portrait.png" />
    <gap:splash gap:platform="blackberry" src="res/screen/blackberry/screen-225.png"
/>
    <gap:splash gap:platform="ios" height="480"
src="res/screen/ios/screen-iphone-portrait.png" width="320" />
    <gap:splash gap:platform="ios" height="960"
src="res/screen/ios/screen-iphone-portrait-2x.png" width="640" />
```

```
        <gap:splash gap:platform="ios" height="1024"
        src="res/screen/ios/screen-ipad-portrait.png" width="768" />
        <gap:splash gap:platform="ios" height="768"
        src="res/screen/ios/screen-ipad-landscape.png" width="1024" />
        <gap:splash gap:platform="winphone"
        src="res/screen/windows-phone/screen-portrait.jpg" />
        <access origin="http://127.0.0.1*" />
        <content src="index.html" />
</widget>
```

```
/*
*file index.css
*
*
* Licensed to the Apache Software Foundation (ASF) under one
* or more contributor license agreements.  See the NOTICE file
* distributed with this work for additional information
* regarding copyright ownership.  The ASF licenses this file
* to you under the Apache License, Version 2.0 (the
* "License"); you may not use this file except in compliance
* with the License.  You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing,
* software distributed under the License is distributed on an
* "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
* KIND, either express or implied.  See the License for the
* specific language governing permissions and limitations
* under the License.
*/
* {
    -webkit-tap-highlight-color: rgba(0,0,0,0); /* make transparent link selection,
    adjust last value opacity 0 to 1.0 */
}

body {
    -webkit-touch-callout: none;                /* prevent callout to copy image, etc
    when tap to hold */
    -webkit-text-size-adjust: none;             /* prevent webkit from resizing text
    to fit */
    -webkit-user-select: none;                  /* prevent copy paste, to allow,
    change 'none' to 'text' */
    background-color:#E4E4E4;
    background-image:linear-gradient(top, #A7A7A7 0%, #E4E4E4 51%);
    background-image:-webkit-linear-gradient(top, #A7A7A7 0%, #E4E4E4 51%);
    background-image:-ms-linear-gradient(top, #A7A7A7 0%, #E4E4E4 51%);
    background-image:-webkit-gradient(
        linear,
        left top,
        left bottom,
        color-stop(0, #A7A7A7),
        color-stop(0.51, #E4E4E4)
    );
    background-attachment:fixed;
    font-family:'HelveticaNeue-Light', 'HelveticaNeue', Helvetica, Arial, sans-serif;
    font-size:12px;
    height:100%;
    margin:0px;
    padding:0px;
    text-transform:uppercase;
    width:100%;
}

/* Portrait layout (default) */
.app {
    background:url(../img/logo.png) no-repeat center top; /* 170px x 200px */
    position:absolute;                  /* position in the center of the screen */
    left:50%;
    top:50%;
    height:50px;                        /* text area height */
    width:225px;                        /* text area width */
    text-align:center;
    padding:180px 0px 0px 0px;          /* image height is 200px (bottom 20px are
    overlapped with text) */
```

100

```
        margin:-115px 0px 0px -112px;   /* offset vertical: half of image height and text
        area height */
                                         /* offset horizontal: half of text area width */
}

/* Landscape layout (with min-width) */
@media screen and (min-aspect-ratio: 1/1) and (min-width:400px) {
    .app {
        background-position:left center;
        padding:75px 0px 75px 170px;   /* padding-top + padding-bottom + text area =
        image height */
        margin:-90px 0px 0px -198px;   /* offset vertical: half of image height */
                                       /* offset horizontal: half of image width and
                                       text area width */
    }
}

h1 {
    font-size:24px;
    font-weight:normal;
    margin:0px;
    overflow:visible;
    padding:0px;
    text-align:center;
}

.event {
    border-radius:4px;
    -webkit-border-radius:4px;
    color:#FFFFFF;
    font-size:12px;
    margin:0px 30px;
    padding:2px 0px;
}

.event.listening {
    background-color:#333333;
    display:block;
}

.event.received {
    background-color:#4B946A;
    display:none;
}

@keyframes fade {
    from { opacity: 1.0; }
    50% { opacity: 0.4; }
    to { opacity: 1.0; }
}

@-webkit-keyframes fade {
    from { opacity: 1.0; }
    50% { opacity: 0.4; }
    to { opacity: 1.0; }
}

.blink {
    animation:fade 3000ms infinite;
    -webkit-animation:fade 3000ms infinite;
}
```

101

```html
<!DOCTYPE html>
<!--

    file index.html

    Licensed to the Apache Software Foundation (ASF) under one
    or more contributor license agreements.  See the NOTICE file
    distributed with this work for additional information
    regarding copyright ownership.  The ASF licenses this file
    to you under the Apache License, Version 2.0 (the
    "License"); you may not use this file except in compliance
    with the License.  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

    Unless required by applicable law or agreed to in writing,
    software distributed under the License is distributed on an
    "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
     KIND, either express or implied.  See the License for the
    specific language governing permissions and limitations
    under the License.
-->
<html>
    <head>
        <meta charset="utf-8" />
        <meta name="format-detection" content="telephone=no" />
        <meta name="viewport" content="user-scalable=no, initial-scale=1,
        maximum-scale=1, minimum-scale=1, width=device-width, height=device-height,
        target-densitydpi=device-dpi" />
        <link rel="stylesheet" type="text/css" href="css/index.css" />
        <title>Hello World</title>
    </head>
    <body>
        <div class="app">
            <h1>PhoneGap</h1>
            <div id="deviceready" class="blink">
                <p class="event listening">Connecting to Device</p>
                <p class="event received">Device is Ready</p>
            </div>
        </div>
        <script type="text/javascript" src="phonegap.js"></script>
        <script type="text/javascript" src="js/index.js"></script>
        <script type="text/javascript">
            app.initialize();
        </script>
    </body>
</html>
```

```
/*
 *
 *file index.js
 *
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements.  See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership.  The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
 * with the License.  You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 * software distributed under the License is distributed on an
 * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 * KIND, either express or implied.  See the License for the
 * specific language governing permissions and limitations
 * under the License.
 */
var app = {
    // Application Constructor
    initialize: function() {
        this.bindEvents();
    },
    // Bind Event Listeners
    //
    // Bind any events that are required on startup. Common events are:
    // 'load', 'deviceready', 'offline', and 'online'.
    bindEvents: function() {
        document.addEventListener('deviceready', this.onDeviceReady, false);
    },
    // deviceready Event Handler
    //
    // The scope of 'this' is the event. In order to call the 'receivedEvent'
    // function, we must explicity call 'app.receivedEvent(...);'
    onDeviceReady: function() {
        app.receivedEvent('deviceready');
    },
    // Update DOM on a Received Event
    receivedEvent: function(id) {
        var parentElement = document.getElementById(id);
        var listeningElement = parentElement.querySelector('.listening');
        var receivedElement = parentElement.querySelector('.received');

        listeningElement.setAttribute('style', 'display:none;');
        receivedElement.setAttribute('style', 'display:block;');

        console.log('Received Event: ' + id);
    }
};
```

Appendix H
PhoneGap TestApp Application

```
<!--
file config.xml
Courtney Thaden
2013-2014
-->
<?xml version='1.0' encoding='utf-8'?>
<widget id="com.courtneythaden.TestApp" version="1.0.0"
xmlns="http://www.w3.org/ns/widgets" xmlns:gap="http://phonegap.com/ns/1.0">
    <name>TestApp</name>
    <description>
        Test application
    </description>
    <author>
        Courtney Thaden using the PhoneGap HelloWorld Template
    </author>
    <feature name="http://api.phonegap.com/1.0/device" />
    <preference name="permissions" value="none" />
    <preference name="orientation" value="default" />
    <preference name="target-device" value="universal" />
    <preference name="fullscreen" value="true" />
    <preference name="webviewbounce" value="true" />
    <preference name="prerendered-icon" value="true" />
    <preference name="stay-in-webview" value="false" />
    <preference name="ios-statusbarstyle" value="black-opaque" />
    <preference name="detect-data-types" value="true" />
    <preference name="exit-on-suspend" value="false" />
    <preference name="show-splash-screen-spinner" value="true" />
    <preference name="auto-hide-splash-screen" value="true" />
    <preference name="disable-cursor" value="false" />
    <preference name="android-minSdkVersion" value="7" />
    <preference name="android-installLocation" value="auto" />
    <icon src="icon.png" />
    <icon gap:density="ldpi" gap:platform="android"
src="res/icon/android/icon-36-ldpi.png" />
    <icon gap:density="mdpi" gap:platform="android"
src="res/icon/android/icon-48-mdpi.png" />
    <icon gap:density="hdpi" gap:platform="android"
src="res/icon/android/icon-72-hdpi.png" />
    <icon gap:density="xhdpi" gap:platform="android"
src="res/icon/android/icon-96-xhdpi.png" />
    <icon gap:platform="blackberry" src="res/icon/blackberry/icon-80.png" />
    <icon gap:platform="blackberry" gap:state="hover"
src="res/icon/blackberry/icon-80.png" />f
    <icon gap:platform="ios" height="57" src="res/icon/ios/icon-57.png" width="57" />
    <icon gap:platform="ios" height="72" src="res/icon/ios/icon-72.png" width="72" />
    <icon gap:platform="ios" height="114" src="res/icon/ios/icon-57-2x.png"
width="114" />
    <icon gap:platform="ios" height="144" src="res/icon/ios/icon-72-2x.png"
width="144" />
    <icon gap:platform="webos" src="res/icon/webos/icon-64.png" />
    <icon gap:platform="winphone" src="res/icon/windows-phone/icon-48.png" />
    <icon gap:platform="winphone" gap:role="background"
src="res/icon/windows-phone/icon-173.png" />
    <gap:splash gap:density="ldpi" gap:platform="android"
src="res/screen/android/screen-ldpi-portrait.png" />
    <gap:splash gap:density="mdpi" gap:platform="android"
src="res/screen/android/screen-mdpi-portrait.png" />
    <gap:splash gap:density="hdpi" gap:platform="android"
src="res/screen/android/screen-hdpi-portrait.png" />
    <gap:splash gap:density="xhdpi" gap:platform="android"
src="res/screen/android/screen-xhdpi-portrait.png" />
    <gap:splash gap:platform="blackberry" src="res/screen/blackberry/screen-225.png"
/>
    <gap:splash gap:platform="ios" height="480"
src="res/screen/ios/screen-iphone-portrait.png" width="320" />
```

104

```
        <gap:splash gap:platform="ios" height="960"
        src="res/screen/ios/screen-iphone-portrait-2x.png" width="640" />
        <gap:splash gap:platform="ios" height="1024"
        src="res/screen/ios/screen-ipad-portrait.png" width="768" />
        <gap:splash gap:platform="ios" height="768"
        src="res/screen/ios/screen-ipad-landscape.png" width="1024" />
        <gap:splash gap:platform="winphone"
        src="res/screen/windows-phone/screen-portrait.jpg" />
        <access origin="http://127.0.0.1*" />
        <content src="index.html" />
</widget>
```

105

```css
/*
 *
 *file index.css
 *Courtney Thaden
 *2013-2014
 *
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements.  See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership.  The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
 * with the License.  You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 * software distributed under the License is distributed on an
 * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 * KIND, either express or implied.  See the License for the
 * specific language governing permissions and limitations
 * under the License.
 */
* {
    -webkit-tap-highlight-color: rgba(0,0,0,0); /* make transparent link selection,
    adjust last value opacity 0 to 1.0 */
}

html, body {

  margin: 0;
  padding: 0;
  width: 100%;
  height: 100%;
  background-color: #FFFFFF;
  font-family: sans-serif;
  text-decoration: none;

  /* Disable text selection. */
  -webkit-text-size-adjust: none;             /* prevent webkit from resizing text to
  fit */
  -webkit-touch-callout: none;
  -webkit-user-select: none;
  -khtml-user-select: none;
  -moz-user-select: none;
  -ms-user-select: none;
  -o-user-select: none;
  user-select: none;
}

.pane {
/*  display: block;
  margin: 5px;*/
  border: 0px solid #EEEEEE;
  -webkit-border-radius: 5px;
  border-radius: 5px;
  clear: both;
  width: 200px;
  padding: 10px;
}

#info {
  background: #222222;
  line-height: 1.2em;
}
```

```css
.infoItem {
  font-size: 1em;
  font-weight: bold;
  color: #EEEEEE;
}

.button {
  border: 1px solid #222222;
  text-align: center;
  background: #15D2FF;
  color: #222222;
  font-size: 1.4em;
  font-weight: bold;
  -webkit-border-radius: 8px;
  border-radius: 8px;
  padding: 15px 10px 15px 10px;
  cursor: pointer;
  width: auto;
}

#divider {
    background: #FF7F00;
  color: #EEEEEE;
  font-size: 1.6em;
  font-weight: bold;
  text-align: center;
  margin-top: 10px;
}
```

```html
<!DOCTYPE html>
<!--
    file index.html
    Courtney Thaden
    2013-2014

    Licensed to the Apache Software Foundation (ASF) under one
    or more contributor license agreements.  See the NOTICE file
    distributed with this work for additional information
    regarding copyright ownership.  The ASF licenses this file
    to you under the Apache License, Version 2.0 (the
    "License"); you may not use this file except in compliance
    with the License.  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

    Unless required by applicable law or agreed to in writing,
    software distributed under the License is distributed on an
    "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
     KIND, either express or implied.  See the License for the
    specific language governing permissions and limitations
    under the License.
-->
<html>
    <head>
        <meta charset="utf-8" />
        <meta http-equiv="Content-type" content="text/html"/>
        <meta name="format-detection" content="telephone=no" />
        <meta name="viewport" content="user-scalable=no, initial-scale=1,
        maximum-scale=1, minimum-scale=1, width=device-width, height=device-height,
        target-densitydpi=device-dpi" />
        <link rel="stylesheet" type="text/css" href="css/index.css" />
        <title>TestApp</title>
          <script type="text/javascript" src="phonegap.js"></script>
          <script type="text/javascript" src="js/index.js"></script>
          <script type="text/javascript">
             app.initialize();
          </script>


    </head>
    <body>
      <div id="screen">

        <div class="pane" id="info">
          <div class="infoItem">Platform: <span id="platform"> </span></div>
          <div class="infoItem">Version: <span id="version"> </span></div>
          <div class="infoItem">UUID: <span id="uuid"> </span></div>
          <div class="infoItem">Name: <span id="name"> </span></div>
          <div class="infoItem">Width: <span id="width"> </span></div>
          <div class="infoItem">Height: <span id="height"> </span></div>
        </div>
        <div class="button" onclick="beep()">Fail Noise</div>
        <div class="button" onclick="changeColor()">Change Color</div>

            <div id="divider">

        <h4>Let's Play Rock, Paper, Scissors!</h4>
        <input type="button" onclick="compare(0)" value="Rock"/>
        <input type="button" onclick="compare(1)" value="Paper"/>
        <input type="button" onclick="compare(2)" value="Scissors"/>
        </div>
```

108

```html
<div id="header"><h3>Simple Submission Form</h3></div>
    <div id="inputs">
        <form name="input" action="#" method="post">
         First name: <input type="text" name="firstname" value="first
         name"><br/>
         Last name: <input type="text" name="lastname" value="last
         name"><br/>
         E-mail: <input type="email" name="email" value="e-mail
         address"><br/><br/>
        </form>
    </div>
    <div>
        <form id="options">
            Sex:
            <input type="radio" name="sex" value="male">Male
            <input type="radio" name="sex"
            value="female">Female<br/><br/>
            Major (check all that apply): <br>
            <input type="checkbox" name="notify" value="ChemE">Chemical
            Engineering<br/>
            <input type="checkbox" name="notify" value="CE">Civil
            Engineering<br/>
            <input type="checkbox" name="notify" value="EE">Electrical
            Engineering<br/>
            <input type="checkbox" name="notify" value="ME">Mechanical
            Engineering<br/>


            Comments: <br/><textarea name="msg"></textarea><br/>
            <input type="submit" value="Submit">
        </form>
    </div>
    </div>
  </body>
</html>
```

```
/*
 *
 *file index.js
 *Courtney Thaden
 *2013-2014
 *
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements.  See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership.  The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
 * with the License.  You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 * software distributed under the License is distributed on an
 * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 * KIND, either express or implied.  See the License for the
 * specific language governing permissions and limitations
 * under the License.
 */


/**
 * Displays the device information on the screen.
 */
function showDeviceInfo()
{
  var platform = document.getNativeElementById("PlatformLabel");
  platform.setProperty("text" , "Platform: " + device.platform);

  var version = document.getNativeElementById("VersionLabel");
  version.setProperty("text" , "Version: " + device.version);

  var uuid = document.getNativeElementById("UUIDLabel");
  uuid.setProperty("text" , "UUID: " + device.uuid);

  var deviceName = document.getNativeElementById("NameLabel");
  deviceName.setProperty("text" , "Name: " + device.name);

  var screenWidth = document.getNativeElementById("ScreenWidthLabel");
  screenWidth.setProperty("text" , "Screen Width: " + screen.width);

  var screenHeight = document.getNativeElementById("ScreenHeightLabel");
  screenHeight.setProperty("text" , "Screen Width: " + screen.height);
}



    /**
     * Play one sound.
     */
    function noiseButtonClicked()
    {
      navigator.notification.beep(1);
    }

    /**
     * Change page background to a random color.
     */
    function colorButtonClicked()
    {
```

```
    var layout = document.getNativeElementById("mainLayout");
    var color = "#" +
      (Math.random() * 0xFFFFFF + 0x1000000)
        .toString(16).substr(1,6);
    layout.setProperty("backgroundColor", color);
}

function compare(index)
    {
    var rps = ["rock", "paper", "scissors"];
    var rules = {
        rock: {
          scissors: "breaks"
        },
        paper: {
          rock: "covers",
        },
        scissors: {
          paper: "cuts"
        }};
    var userChoice = rps[index];
        var computerChoice = rps[Math.floor(Math.random() * 5)];

        if (userChoice !== computerChoice){
          if (rules[userChoice].hasOwnProperty(computerChoice))
            alert("You win - " + userChoice + " " +
            rules[userChoice][computerChoice] +
                " " + computerChoice + ".");
          else
            alert("You lose - " + computerChoice + " " +
            rules[computerChoice][userChoice] +
                " " + userChoice + ".");
        }
        else
          alert("Draw. We both played " + userChoice + ".");
};


function rockButtonClicked()
{
  compare(0);
}

  function paperButtonClicked()
{
  compare(1);
}

function scissorsButtonClicked()
{
  compare(2);
}




var app = {
    // Application Constructor
    initialize: function() {
        this.bindEvents();
    },
    // Bind Event Listeners
    //
```

```
// Bind any events that are required on startup. Common events are:
// 'load', 'deviceready', 'offline', and 'online'.
bindEvents: function() {
    document.addEventListener('deviceready', this.onDeviceReady, false);
},
bindEvents: funtion(){
document.addEventListener('deviceready', showDeviceInfo, true);
},


// deviceready Event Handler
//
// The scope of 'this' is the event. In order to call the 'receivedEvent'
// function, we must explicity call 'app.receivedEvent(...);'
onDeviceReady: function() {
    app.receivedEvent('deviceready');
},
// Update DOM on a Received Event
receivedEvent: function(id) {
    var parentElement = document.getElementById(id);
    var listeningElement = parentElement.querySelector('.listening');
    var receivedElement = parentElement.querySelector('.received');

    listeningElement.setAttribute('style', 'display:none;');
    receivedElement.setAttribute('style', 'display:block;');

    console.log('Received Event: ' + id);
}
```

Appendix I
Corral Correspondence: Permission to Use

c thaden <courtney.thaden@gmail.com>   12:24 PM (10 hours ago)
to luis.corral

Potential advantages and disadvantages of multiplatform development frameworks - A vision on mobile environments
Luis Corral / *Procedia Computer Science, 10,* 1202-1207

I was wondering if I could use Figure 1 a and b and Tables 1-3 from your article in my Master's Thesis in order to better explain the differences in the traditional and multiplatform development models and potential advantages and disadvantages of both. My thesis is an analysis of multiplatform development applications.
Please let me know at your earliest convenience.
Thank you,
Courtney Thaden

Corral Luis (unibz) <luis.corral@unibz.it>   3:50 PM (6 hours ago)
to me

Hi Courtney

of course you can use the figure and tables, I am glad you find them useful. The only thing I would ask you is to add a reference to the original work, something like:

Corral, L.; Janes, A.; & Remencius, T.; Potential Advantages and Disadvantages of
Multiplatform Development Frameworks-A Vision on Mobile Environments, Procedia Computer
Science, Volume 10, 2012, Pages 1202-1207, ISSN 1877-0509, http://dx.doi.org/10.1016/j.
procs.2012.06.173.

Also, I also suggest you to check the following articles, you may find them useful:

http://scholar.google.it/citations?view_op=view_citation&hl=en&user=ywf9nzcAAAAJ&citation_for_
view=ywf9nzcAAAAJ:d1gkVwhDpl0C

http://scholar.google.it/citations?view_op=view_citation&hl=en&user=ywf9nzcAAAAJ&citation_for_
view=ywf9nzcAAAAJ:u-x6o8ySG0sC

Please let me know if it suits your needs! Good work and if there is anything else I can do, just drop me an email.

...

Luis

Appendix K
ACM: License to Use

**ASSOCIATION FOR COMPUTING MACHINERY, INC. LICENSE
TERMS AND CONDITIONS**

Apr 15, 2014

This is a License Agreement between Courtney Thaden ("You") and Association for Computing Machinery, Inc. ("Association for Computing Machinery, Inc.") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by Association for Computing Machinery, Inc., and the payment terms and conditions.

License Number	3370290214461
License date	Apr 15, 2014
Licensed content publisher	Association for Computing Machinery, Inc.
Licensed content publication	Communications of the ACM
Licensed content title	Mobile application development: web vs. native
Licensed content author	Andre Charland, et al
Licensed content date	May 1, 2011
Volume number	54
Issue number	5
Type of Use	Thesis/Dissertation
Requestor type	Academic
Format	Print and electronic
Portion	figure/table
Number of figures/tables	1
Will you be translating?	No
Order reference number	
Title of your thesis/dissertation	Analysis of Multi-Platform Mobile Application Development
Expected completion date	May 2014
Estimated size (pages)	130
Billing Type	Credit Card
Credit card info	Visa ending in 3092
Credit card expiration	12/2016
Total	8.00 USD
Terms and Conditions	

Rightslink Terms and Conditions for ACM Material

1. The publisher of this copyrighted material is Association for Computing Machinery, Inc. (ACM). By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the Billing and Payment terms and conditions established by Copyright Clearance Center, Inc. ("CCC"), at the time that you opened your Rightslink account and that are available at any time at).

2. ACM reserves all rights not specifically granted in the combination of (i) the license details provided by you and accepted in the course of this licensing transaction, (ii) these terms and conditions and (iii) CCC's Billing and Payment terms and conditions.

3. ACM hereby grants to licensee a non-exclusive license to use or republish this ACM-copyrighted material* in secondary works (especially for commercial distribution) with the stipulation that consent of the lead author has been obtained independently. Unless otherwise stipulated in a license, grants are for one-time use in a single edition of the work, only with a maximum distribution equal to the number that you identified in the licensing process. Any additional form of republication must be specified according to the terms included at the time of licensing.

*Please note that ACM cannot grant republication or distribution licenses for embedded third-party material. You must confirm the ownership of figures, drawings and artwork prior to use.

4. Any form of republication or redistribution must be used within 180 days from the date stated on the license and any electronic posting is limited to a period of six months unless an extended term is selected during the licensing process. Separate subsidiary and subsequent republication licenses must be purchased to redistribute copyrighted material on an extranet. These licenses may be exercised anywhere in the world.

5. Licensee may not alter or modify the material in any manner (except that you may use, within the scope of the license granted, one or more excerpts from the copyrighted material, provided that the process of excerpting does not alter the meaning of the material or in any way reflect negatively on the publisher or any writer of the material).

6. Licensee must include the following copyright and permission notice in connection with any reproduction of the licensed material: "[Citation] © YEAR Association for Computing Machinery, Inc. Reprinted by permission." Include the article DOI as a link to the definitive version in the ACM Digital Library. Example: Charles, L. "How to Improve Digital Rights Management," Communications of the ACM, Vol. 51:12, © 2008 ACM, Inc. http://doi.acm.org/10.1145/nnnnnn.nnnnnn (where nnnnnn.nnnnnn is replaced by the actual number).

7. Translation of the material in any language requires an explicit license identified during the licensing process. Due to the error-prone nature of language translations, Licensee must include the following copyright and permission notice and disclaimer in connection with any reproduction of the licensed material in translation: "This translation is a derivative of ACM-copyrighted material. ACM did not prepare this translation and does not guarantee that it is an accurate copy of the originally published work. The original intellectual property contained in this work remains the property of ACM."

8. You may exercise the rights licensed immediately upon issuance of the license at the end of the licensing transaction, provided that you have disclosed complete and accurate details of your proposed use. No license is finally effective unless and until full payment is received from you (either by CCC or ACM) as provided in CCC's Billing and Payment terms and conditions.

9. If full payment is not received within 90 days from the grant of license transaction, then any license preliminarily granted shall be deemed automatically revoked and shall be void as

116

if never granted. Further, in the event that you breach any of these terms and conditions or any of CCC's Billing and Payment terms and conditions, the license is automatically revoked and shall be void as if never granted.

10. Use of materials as described in a revoked license, as well as any use of the materials beyond the scope of an unrevoked license, may constitute copyright infringement and publisher reserves the right to take any and all action to protect its copyright in the materials.

11. ACM makes no representations or warranties with respect to the licensed material and adopts on its own behalf the limitations and disclaimers established by CCC on its behalf in its Billing and Payment terms and conditions for this licensing transaction.

12. You hereby indemnify and agree to hold harmless ACM and CCC, and their respective officers, directors, employees and agents, from and against any and all claims arising out of your use of the licensed material other than as specifically authorized pursuant to this license.

13. This license is personal to the requestor and may not be sublicensed, assigned, or transferred by you to any other person without publisher's written permission.

14. This license may not be amended except in a writing signed by both parties (or, in the case of ACM, by CCC on its behalf).

15. ACM hereby objects to any terms contained in any purchase order, acknowledgment, check endorsement or other writing prepared by you, which terms are inconsistent with these terms and conditions or CCC's Billing and Payment terms and conditions. These terms and conditions, together with CCC's Billing and Payment terms and conditions (which are incorporated herein), comprise the entire agreement between you and ACM (and CCC) concerning this licensing transaction. In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall control.

16. This license transaction shall be governed by and construed in accordance with the laws of New York State. You hereby agree to submit to the jurisdiction of the federal and state courts located in New York for purposes of resolving any disputes that may arise in connection with this licensing transaction.

17. There are additional terms and conditions, established by Copyright Clearance Center, Inc. ("CCC") as the administrator of this licensing service that relate to billing and payment for licenses provided through this service. Those terms and conditions apply to each transaction as if they were restated here. As a user of this service, you agreed to those terms and conditions at the time that you established your account, and you may see them again at any time at http://myaccount.copyright.com

18. Thesis/Dissertation: This type of use requires only the minimum administrative fee. It is not a fee for permission. Further reuse of ACM content, by ProQuest/UMI or other document delivery providers, or in republication requires a separate permission license and fee. Commercial resellers of your dissertation containing this article must acquire a separate license.

Special Terms:

If you would like to pay for this license now, please remit this license along with your payment made payable to "COPYRIGHT CLEARANCE CENTER" otherwise you will be invoiced within 48 hours of the license date. Payment should be in the form of a check or money order referencing your account number and this invoice number RLNK501278761.
Once you receive your invoice for this order, you may pay your invoice by credit card. Please follow instructions provided at that time.

Make Payment To:
Copyright Clearance Center
Dept 001
P.O. Box 843006
Boston, MA 02284-3006

For suggestions or comments regarding this order, contact RightsLink Customer Support: customercare@copyright.com or +1-877-622-5543 (toll free in the US) or +1-978-646-2777.

Gratis licenses (referencing $0 in the Total field) are free. Please retain this printable license for your reference. No payment is required.

REFERENCES

Abolfazil, S., Sanaei, Z., Gani, A., Xia, F., & Yang, L. T. (2014). Rich mobile
applications: genesis, taxonomy, and open issues. *Journal of Network and
Computer Applications, 40,* 345-362.

Appcelerator. (n.d.a). Appcelerator Platform. *In Appcelerator online.* Retrieved from
http://www.appcelerator.com/platform/appcelerator-platform/

Appcelerator. (n.d.b). Cross-platform mobile development in titanium. *In Appcelerator
online.* Retrieved from http://docs.appcelerator.com/titanium/3.0/#!/guide/Cross-
Platform_Mobile_Development_In_Titanium

Appcelerator. (n.d.c). Supporting multiple platforms in a single codebase. *In
Appcelerator online.* Retrieved from
http://docs.appcelerator.com/titanium/latest/#!/guide/Supporting_Multiple_Platfor
ms_in_a_Single_Codebase

Appcelerator. (n.d.d). Titanium. *In Appcelerator online.* Retrieved from
http://www.appcelerator.com/titanium/

Appcelerator. (n.d.e). Titanium SDK. *In Appcelerator online.* Retrieved from
http://www.appcelerator.com/titanium/titanium-sdk/

Application. (2014). In *Tech Terms Computer Dictionary online.* Retrieved from
http://www.techterms.com/definition/application

Application Programming Interface. (2014). In *Tech Terms Computer Dictionary online.*

    Retrieved from http://www.techterms.com/definition/api

Barmpatsalou, K., Damopoulos, D., Kambourakis, G., & Katos, V. (2013). A critical

    review of 7 years of mobile device forensics. *Digital Investigation, 10,* 323-349.

Blom, S., Book, M., Hrushchak, R., & Köhler, A. (2008). Write once, run anywhere a

    survey of mobile runtime environments. *Proc. 3rd Int'l Conf. Grid and Pervasive*

    Computing (GPC 08), IEEE CS Press, 2008, pp. 132–137.

Boardman, B. (2012). No app for that? Write one! *Industrial Engineer: IE*, *44*(3), 44-48.

Charland, A. & Leroux, B. (2011). Mobile application development: web vs. native.

    Communications of the ACM, 54(5), 49-53. doi :10.1145/1941487.1941504

Closed System. (2014). In *Wikipedia, The Free Encyclopedia online.* Retrieved from

    http://en.wikipedia.org/wiki/Closed_source_software

Corral, L., Janes, A., & Remencius, T. (2012a). Potential advantages and disadvantages

    of multiplatform development frameworks–a vision on mobile environments.

    Procedia Computer Science, 10, 1202-1207. doi.org/10.1016/j.bbr.2011.03.031

Corral, L., Sillitti, A., & Succi, G. (2012b). Mobile multiplatform development: An

    experiment for performance analysis. *Procedia Computer Science, 10*, 736-743.

Debug. (2014). In *Tech Terms Computer Dictionary online.* Retrieved from

    http://www.techterms.com/definition/debug

Developer. (n.d.) . In *PCMag online.* Retrieved from

    http://www.pcmag.com/encyclopedia/term/41187/developer

Dupont, B. (2012, July 1). Develop once, run everywhere? *Information Week, 2*.

Dye, S. M. & Scarfone, K. (2014). A standard for developing secure mobile applications. *Computer Standards & Interfaces, 36,* 534-530.

Emmanouilidis, C., Koutsiamanis, R. A., & Tasidou, A. (2013). Mobile guides: Taxonomy of architectures, context awareness, technologies and applications. *Journal of Network and Computer Applications, 36,* 103-125.

Event Listener. (2003). In *W3C online.* Retrieved from http://www.w3.org/2003/01/dom2-javadoc/org/w3c/dom/events/EventListener.html

Extensible Markup Language. (2014). In *Tech Terms Computer Dictionary online.* Retrieved from http://www.techterms.com/definition/xml

Finnie, S. (2013, January 14). 5 tips for developing successful mobile apps. *Computerworld*, *47*(1), 40.

Gandhewar, N. & Sheikh, R. (2011, February). Google android: An emerging software platform for mobile devices. *International Journal on Computer Science and Engineering,* 12-17.

Gavalas, D. & Economou, D. (2011). Development platforms for mobile applications: Status and trends. *IEEE Software, 28*(1), 77-86.

Graphical User Interface. (2014). In *Tech Terms Computer Dictionary online.* Retrieved from http://www.techterms.com/definition/gui

Holzer, A. & Ondrus, J. (2011). Mobile application market: A developer's perspective. *Telmatics and Informatics, 28*, 22-31.

Hybrid App. (n.d.). In *Appcelerator online.* Retrieved from

http://docs.appcelerator.com/titanium/latest/#!/guide/Mobile_Web_Platform_Ove

rview

Interface. (2014). In *Tech Terms Computer Dictionary online.* Retrieved from

http://www.techterms.com/definition/user_interface

Internet. (2014). In *Terms Computer Dictionary online.* Retrieved from

http://www.techterms.com/definition/internet

iOS. (2014). iOS: a virtual history. In *The Verge online*. Retrieved from

http://www.theverge.com/2011/12/13/2612736/ios-history-iphone-ipad

Model-View-Controller. (2014). In *Wikipedia, The Free Encyclopedia online.* Retrieved

from

http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

MoSync. (2013a). Create your first app. In *MoSync online.* Retrieved from

http://www.mosync.com/docs/index.html

MoSync. (2013b). In *MoSync online.* Retrieved from

http://www.mosync.com/docs/sdk/js/guides/quick-start/how-create-html5-project-

mosync/index.html

MoSync. (2013c). MoSync reload versus mosync SDK. In *MoSync online.* Retrieved

from http://www.mosync.com/docs/reload/guides/quick-start/mosync-reload-vs-

sdk/index.html

Multi-Platform Application. (2014). In *Tech Terms Computer Dictionary online.*

Retrieved from http://www.techterms.com/definition/multiplatform

122

Native Application. (n.d.). In *Appcelerator online.* Retrieved from

http://docs.appcelerator.com/titanium/latest/#!/guide/Mobile_Web_Platform_Ove

rview

Open System. (2014). In *Wikipedia, The Free Encyclopedia online.* Retrieved from

http://en.wikipedia.org/wiki/Open-source_software

Operating System. (2014). In *Tech Terms Computer Dictionary online.* Retrieved from

http://www.techterms.com/definition/operating_system

PhoneGap. (2014a). Android Platform Guide . *In PhoneGap online.* Retrieved from

http://docs.phonegap.com/en/edge/guide_platforms_android_index.md.html#Andr

oid%20Platform%20Guide

PhoneGap. (2014b). *In PhoneGap online.* Retrieved from http://phonegap.com/

PhoneGap. (2014c). Overview. *In PhoneGap online.* Retrieved from

http://docs.phonegap.com/en/edge/guide_overview_index.md.html#Overview

PhoneGap. (2014d). PhoneGap Documentation: Introducing PhoneGap Build. *In*

PhoneGap online. Retrieved from

http://docs.phonegap.com/en/edge/guide_phonegapbuild_index.md.html#Introduc

ing%20PhoneGap%20Build

Platform. (2014). In *Tech Terms Computer Dictionary online.*

Retrieved from http://www.techterms.com/definition/platform

Portable Operating System Interface. (2014). In *Wikipedia, The Free Encyclopedia*

online. Retrieved from http://en.wikipedia.org/wiki/POSIX

Proffitt, B. (2011). Tools & toys: Open android-for better and for worse. *IEEE Spectrum,*

48(5), 22-25.

Sharma, K. (2011). Android in opposition to iphone. *International Journal on Computer*

    Science and Engineering, 3(5), 1965-1969.

Smartphone. (2014). In *Tech Terms Computer Dictionary online.* Retrieved from

    http://www.techterms.com/definition/smartphone

SMS. (2014). In *Tech Terms Computer Dictionary online.* Retrieved from

    http://www.techterms.com/definition/sms

Software Development Kit. (2014). In *Tech Terms Computer Dictionary online.*

    Retrieved from http://www.techterms.com/definition/sdk

Tablet. (2014). In *Tech Terms Computer Dictionary online.* Retrieved from

    http://www.techterms.com/definition/tablet

Web App. (n.d.). In *Appcelerator online.* Retrieved from

    http://docs.appcelerator.com/titanium/latest/#!/guide/Mobile_Web_Platfor

    m_Overview

Wong, C. Y., Khong, C. W., & Chu, K. (2012). Interface design practice and education

    towards mobile apps development. *Procedia – Social and Behavioral Sciences,*

    51, 698-702.

World Wide Web Consortium. (2012). In *W3C online.* Retrieved from

    http://www.w3.org/Consortium/

Zakas, N.C. (2013). The evolution of web development for mobile devices.

    Communications of the ACM, 56(4), 42-48. doi:10.1145/2436256.2436269