CS Capstone Technology Review

November 21, 2017

eBay iOS eSports Application

CS461 Senior Software Engineering Project I

Fall 2017

Prepared for

eBay

Luther Boorn _____ _____
Signature *Date*

Prepared by

Group 17

Skill Capped IRL

Meagan Olsen _____ _____
Signature *Date*

CONTENTS

1 Piece 1:Platform 2

1.1 Overview . . . 2
1.2 Criteria . . . 2
1.3 Android . . . 2
1.4 Universal Windows Platform . . . 2
1.5 iOS . . . 3
1.6 Conclusion . . . 3

2 Piece 2:Framework 3

2.1 Overview . . . 3
2.2 Criteria . . . 4
2.3 Objective C . . . 4
2.4 Xarmin . . . 4
2.5 Swift . . . 4
2.6 Discussion . . . 5
2.7 Conclusion . . . 5

3 Piece 3:Navigation 5

3.1 Overview . . . 5
3.2 Criteria . . . 5
3.3 Hamburger menu . . . 5
3.4 Tab Bars . . . 6
3.5 Tabs . . . 6
3.6 Discussion . . . 6
3.7 Conclusion . . . 6

References 6

1 PIECE 1:PLATFORM

1.1 Overview

Deciding which platform to use is a fundamental design decision when developing a mobile application, and it is is one that must be decided before almost anything else because it has a fundamental impact on several other decisions such as what kinds of tools are going to be used, what libraries are available to the developer, and much more.

1.2 Criteria

For our application,there was several criteria that we had to look at when choosing which platform to develop our mobile application on. Perhaps the most important criteria for selecting a platform was consistency with the corporation standards that our client is associated with. Another important criterion that we had to consider was the type of application that we wanted our end product to be. We want an iOS application, so our platform needs to support iOS development. We also had to consider security and what kind of features it had for UI development. The last thing to consider was performance. We want strong performance for a mobile application. In general, native applications tend to have many optimizations at the compiler level.The three platform choices that we explored were iOS, Android, and Windows.

1.3 Android

Android is an open source Linux based platform which is used to build applications which run on many different types of devices including phones, televisions, and tablets. [1] Being Linux based allows the platform to leverage some of the added features of the Linux operating system and creates a common ground for the creation of new hardware devices. For example, the Android Runtime(ART) is able to rely on the Linux Kernel for implementation of features such as threading and memory management. Android's implementation of the Linux user based model also ensures that each application runs as its own process with restricted permissions. Furthermore, each process also has its own virtual machine to ensure isolation from other applications. Each application also has its own Android Runtime(ART), which optimizes memory usage through optimization of bytecode. Some distinct features of the ART are its support for Ahead of Time(AOT) and Just in Time(JIT) compilation, its optimized implementation of garbage collection, and added debugging support .Android also provides support for the Java Runtime Libraries, allowing for a convenient, easy to use,Java development environment. Android also allows developers to take advantage of native c/c++ libraries through a Java API. The Java API framework also allows for code reuse ad developers are able to have access to all the framework API that Android itself uses . Android also provides some built in applications that developers can use such as applications for email, SMS messaging, calendars, Internet browsing, and contacts. Another advantage of Android is its device configuration support built into the applications themselves. This allows for an application to be configured to run on devices of all different screen sizes.

1.4 Universal Windows Platform

The Universal Windows Platform runs on Windows 10 and was built as an extension to the Windows Runtime Platform(Win Rt) which came out with Windows 8. [2]An advantage of the Universal Windows Platform is its support for languages such as C sharp and XAML.A major advantage of the Universal Windows platform is its device support across all devices running Windows. Windows also provides customizable software development kits which allow

applications to be configured to evoke special features on devices that have support for them. In addition, all Windows Applications are packaged allowing for an easy installation. Windows applications allow developers to provide real time data to help keep the application relevant and exciting. Windows Universal platform also provides a scaling mechanism for UI elements to keep them a consistent size across all platforms.

1.5 iOS

The iOS platform is an Apple owned Mach based Unix/OSX platform that specifically runs on Apple owned devices such as iPad, iPhone, and iPod touch devices. [3] Since both hardware and software components are owned by Apple, there is a tight integration between the devices and the software that runs them. It also has a unified interface because of this.The iOS platform also includes a variety of system applications including Phone, Mail, and Safari.iOS comes with an iOS software development kit that allows for running, testing, and installing native applications.One advantage to the iOS framework is that the iOS resources and shared libraries are packaged into frameworks that can be imported through the x Code IDE. Another advantage from the developer prospective is that iOS provides a developer library, which provides several resources for the developer such as API reference, programming guides, release notes, tech notes, sample code.This reference is also accessible with the Apple IDE,X code as well as through the Apple developer website.iOS has a layered architecture, where the OS core features are at the bottom and the higher level services are found at the top.This makes it simpler and easier to write applications for the developer.Another advantage that iOS has is its access to the Cryptography library, which allows for symmetric encryption, hash-based message authentication codes (HMACs), and digests. It also has APIS for performing various types of encryption and pseudo random number generation.[3] subsectionDiscussion One similarity between the Android and Windows platform is that they both provide a common interface for development on multiple different devices whereas iOS only has support for Apple devices.A major difference between these frameworks is that they are all based on a different operating system. While Android is completely open source, Apple is only partially open sourced. A common feature between Android and Windows is that they both have support for development on multiple screen sizes.Android and iOS both have built in applications that are accompany the operating system.Windows and iOS both come with a development kit for ease of testing.Whereas Android is Java based, iOS runs on Objective C.A similarity between Android and iOS is that they are both a layered architecture.

1.6 Conclusion

In conclusion, we chose to use the iOS platform.The reason that we chose this platform is because our goal was to develop for an iOS device. Because it is based on the objective C framework, it served as a good introduction to mobile application development given our backgrounds in C. The corporation that we are developing for also has a large development department that develops specifically for this platform.

2 Piece 2:Framework

2.1 Overview

Framework is an important decision to make when developing a mobile application because it sets the stage for what features the developer will have available from UI design to user interaction to development tools support. This is because the amount of development tools that a developer has is based on which framework is used. For example, certain IDEs have integration capabilities with a particular framework.

2.2 Criteria

For our application, we had to abide by the standards used by our client. We wanted to develop with a particular IDE, so that contributed to choosing our framework. We also did not have a need for cross platform support, so that also contributed to our framework decision. The three frameworks that we looked at were Objective C, Xarmin, and Swift.

2.3 Objective C

Objective C is the development framework for iOS.[4].In the past, it served as the primary programming language for writing OSX and IOS software.[5]It was initially built as an extension to the ANSI C programming language for the purpose of allowing for complex object oriented programming without the complex syntax of C++. Its simple syntax makes it easy to learn while still not restricting the power of an object oriented language. It includes dynamic runtime capabilities, making it a flexible language for design as users have much more freedom to specify things at runtime. Because of this, this framework is known to be a good choice for development and design of front end GUIS.One advantage of this technology is that all of Apple's documentation is based off of it, so it has a wide range of support and documentation.

2.4 Xarmin

Xarmin is a cross platform tool used for mobile development built on the C sharp programming language.[6]Its C sharp interface has a 75 percent code sharing capability.Xarmin also has its own development environment. And advantage of this technology is its support for multiple development environments including MacOSX and Microsoft Visual Studio.Another advantage of this platform is that, by being C sharp based, a developer is able to write code in c sharp which can be executed on Android, oSX, or Windows platforms.This is advantageous for developers who have more of a Windows background since C sharp is a Windows language.Xarmin allows for integration with Microsoft's Mono.Net framework. For iOS, Xarmin supports Ahead of Time Compiling(AOT) to ARM assembly language. For Android development, it performs just in time(JIT) compilation to optimize performance.One advantage of Xarmin is that it has built in tools for platform specific hardware acceleration which cannot be achieved in frameworks like objective C.Another advantage that Xarmin has over objective C and swift is its support for asynchronous programming. It also simplifies lambdas and supports objective C code.Xarmin also supports Java code from Android.Another thing that makes this platform extremely versatile is that it allows access to iOS, Windows, and Android software development kits.It also allows developers to use design tools for Windows, iOS, and Android for UI design.

2.5 Swift

Swift is the newer, more modern, evolved version of Objective C and is now starting to replace it in many applications for all OsX devices.[7]This framework can be used for desktops, servers, mobile devices, and many other things.In addition to Apple contributions, Swift also incorporates modifications from the open source community.Swift has lightweight syntax that is optimized for developers while still allowing for complex logic.It has a playground which allows for easy testing and exploration of code.An advantage that is has for previous iOS developers is that it is syntactically similar to Objective C syntax.It also has advanced types and a type interface. This helps prevent type errors and also favors fewer type declarations.Swift is compatible with all current iOS tools.Another advantage is that it allows integration with objective C.

2.6 Discussion

Perhaps the most obvious differences between these frameworks is the type of applications they support. While Xarmin allows for both native and cross platform development, Objective C and Swift are limited to native development.In contrast to Objective C and Swift, which is made for the Xcode development environment, Xarmin has its own development environment.Xarmin has built in tools for platform specific hardware acceleration which cannot be achieved in frameworks like objective C.Xarmin also has the capabilities for Asynchronous programming, which requires callbacks in objective C and Swift. in Xarmin, syntax is more difficult to implement for lambdas.An advantage that Xarmin has over objective C is that it actually supports objective C including code, frameworks, and custom controls.One difference between Objective C and Swift is that . One difference between Swift and Objective C is its lighter weight syntax.Another difference between Swift and objective C is Swifts support for tuples.It also replaces nil pointers in Objective C, which makes it much safer and less error prone.Objective C does also not have a type interface.Swift also enhances objective c with the features of cocoa touch.

2.7 Conclusion

In conclusion, we chose to use Swift. Our biggest reasoning for this was that this is what our client requested us to use. While Xarmin might have been beneficial if we wished to develop cross platform, this was not our goal.Instead, we wanted to focus more on enhancing our iOS application rather than aim for cross platform capability.

3 Piece 3:Navigation

3.1 Overview

How a user will navigate through an application is an important design decision.This decision can depend on the type of application. For example, iOS, Android, and Windows all have different navigation styles that they implement a little differently.While some of the same types of navigation do show up in Android, iOS, and Windows, navigation style choice really depends on what content is being displayed. Some navigation design patterns are better for different amounts of content. Certain navigational patterns als might be better on different devices such as desktops,laptops, mobile, etc.

3.2 Criteria

We want our application to have a simple, easy to use interface. We also don't have many pages, so styles that display larger amounts of content wouldn't really be appropriate for our application.We are developing on a mobile platform, so it is important that our design pattern scales well for mobile. We are also developing for iOS, so it is important that we choose styles which are widely used in iOS applications since we would like to publish to the Apple store if possible.The three navigational choices that we explored are hamburger menu, tab bars, and tabs.

3.3 Hamburger menu

The hamburger menu is a commonly used navigational method for both mobile and web applications.[8] It is also commonly called a side navigation drawer. It is usually implemented by a carousel with bars somewhere near the top of a page which, when clicked, pulls out a menu of some kind. While these were initially not part of the apple API, they have become a popular design pattern on Android devices and are implemented natively on that platform.One advantage

to this style of navigation is that it allows more space for emphasis of the main content. This can be particularly advantageous on a mobile platform where space is limited. Another advantage that this navigation style ads is the ability to display multiple navigational links that might clutter up the screen on a mobile device.

3.4 Tab Bars

Tab bars are a design pattern that show up frequently in both Android and iOS mobile applications. [9]In iOS, they are more commonly known as a navigation bar whereas Android refers to them as tool bars.In iOS, this navigation style usually contains the title of the current page that the user is on, some sort of arrow to navigate to a previous page, and some sort of filter to control the content of the current page.This design choice is useful in situations where there are fewer navigation choices. It is often implemented when the application uses a hierarchical structure. It is often implemented on applications that use a hierarchical structure.

3.5 Tabs

Tabs are a design pattern that also shows up quite frequently in both iOS and Android. In iOS, tabs usually appear at the bottom of the screen whereas in Android, they are usually found towards the top unless the primary navigation shows up at the bottom of the application.This design pattern is often a good choice for simplicity. Taken from the desktop model, it provides direct access to many links and leaves no ambiguity about what page the end user is on.This is often useful with an application that performs several distinct things because it provides the user a clear indication of where they are on an application. Another advantage that these tabs have is persistence.They always stay on the page.

3.6 Discussion

All of these options are valid in different use cases based on the strengths above.Unlike the hamburger menu which has the capacity to hold many links, tabs are limited in the amount of elements that they can display. Having too many tabs clutters up the screen too much. A difference between hamburger menus and Tabs is the persistence. Whereas hamburger menus are able to be hidden, tabs stay on the page.

3.7 Conclusion

We chose the tabs design pattern as our primary navigation mainly because we have two main functionalities:display products and display events. These two pieces are distinct actions, so this seemed like a solid design choice. We also have a home page for a total of three tabs.Our application does implement tab bars as well on each of our individual pages.Since our application does not present many navigation options, this provides a sound,easy to use navigational style.While our client did suggest that we use this, he also has given us the freedom to customize UI elements such as these.

REFERENCES

[1] A. Developer, “Platform architecture,” https://developer.android.com/guide/platform/index.html#system-apps(2017/11/13).

[2] Microsoft, “Intro to the universal windows platform,” https://docs.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide(2017/11/13).

[3] Apple, “ios technology overview,” https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html#//apple_ref/doc/uid/TP40007898-CH1-SW1(2017/11/13).

[4] ——, “Cocoa core competencies,” https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/ObjectiveC.html(2017/11/13).

[5] Upwork, “Swift vs. objective-c: A look at ios programming languages,” https://abiosgaming.com/esports-data-api(2017/11/13).

[6] X. Inc, “Xarmin products,” https://www.xamarin.com/platform(2017/11/13).

[7] Apple, “The swift programming language (swift 4),” https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/index.html(2017/11/13).

[8] N. Babich, “Basic patterns for mobile navigation,” http://babich.biz/basic-patterns-for-mobile-navigation/(2017/11/13).

[9] J. Rjaud, “Developing for android vs. ios: Navigation patterns,” https://medium.com/@jrejaud/developing-for-android-vs-ios-navigation-patterns-c0e11286562c(2017/11/13).