CS CAPSTONE TECH REVIEW

DECEMBER 6, 2018

APP TO SUPPORT FIELD DIAGNOSTICS IN VETERINARY MEDICINE

PREPARED FOR

OREGON VETERINARY DIAGNOSTIC LABORATORY

DR. CHRISTIANE LOEHR _____ _____
Signature *Date*

PREPARED BY

GROUP 72

MALSANO

BRANDON JOLLY _____ _____
Signature *Date*

KATHERINE JEFFREY _____ _____
Signature *Date*

BRADFORD WONG _____ _____
Signature *Date*

Abstract

Currently, there are many difficulties for veterinary pathologists trying to perform remote diagnostics. There are not any effective ways for people out in the field collecting samples to communicate with specialized experts located in laboratories. As a result, this project will involve creating an android mobile application that will be used as a bridge to connect the field personnel with the veterinary pathologists in laboratories. With this mobile application, the field personnel will be able to take pictures of the individual that is being analyzed and then send the pictures along with other data such as the patient, location, and time to a pathologist. The pathologist will then be able to use the provided information to perform a necropsy and send feedback to the field personnel. This project is intended to support remote field diagnostics in veterinary medicine.

CONTENTS

1 Introduction 2

1.1 Roles . . . 2

1.2 What the team is trying to accomplish . . . 2

2 Pieces of Technology 2

2.1 Mobile Application Frameworks . . . 2

2.1.1 React Native . . . 2

2.1.2 Flutter . . . 2

2.1.3 Native Android . . . 3

2.1.4 Overall Recommendation and Conclusion . . . 3

2.2 Automated Testing . . . 3

2.2.1 Robotium . . . 3

2.2.2 Appium . . . 4

2.2.3 UI Automator . . . 4

2.2.4 Overall Recommendation and Conclusion . . . 4

2.3 Languages . . . 4

2.3.1 Java . . . 4

2.3.2 Kotlin . . . 5

2.3.3 LUA . . . 5

2.3.4 Overall Recommendation and Conclusion . . . 5

2.4 Conclusion . . . 6

References 7

1 Introduction

1.1 Roles

Bradford will be doing native android development for this project. In this tech review, he will be writing about mobile application frameworks, frameworks for automated testing, and languages for mobile applications.

1.2 What the team is trying to accomplish

The team is attempting to create a mobile Android application to improve pathologists' ability to perform remote diagnostics. This application will allow pathologists in a lab to be able to communicate with personnel out in the field. The field personnel will be able to take pictures and write text to create a report. They will then be able to send the report to the personnel in the laboratories, who can then send feedback to the field personnel.

2 Pieces of Technology

2.1 Mobile Application Frameworks

There are a variety of choices when it comes to using a framework to develop this mobile application. The team will most likely use native Android because that is what the client specified, but it is still useful to look at the other frameworks. This paper will look at the following criteria: how many platforms can take advantage of the framework, the development and compilation speed, how popular the framework is, and any extra features in order to evaluate the frameworks.

2.1.1 React Native

A framework that can be used to develop mobile apps is React Native, which is a framework that only uses JavaScript. Native Android, native iOS, and React Native use all of the same fundamental UI building blocks. JavaScript and React are built by putting these blocks together. Additionally, React Native builds faster than native Android because the app can be reloaded instantly after changes instead of having to recompile all the files again. Many mobile apps use React Native including Facebook, Instagram, and Pinterest [13]. The main benefits of React Native are that an application using this framework will work on both iOS and Android, it is faster to build using this framework, and a developer can make changes to the code while the app continues to run and automatically reloads after changes. The drawbacks are that there are less tools to create navigation components to enable seamless UX for users, it lacks some custom modules, and some components such as cameras and sensors still need native developers and knowledge. Lastly, Facebook owns React Native, which can be problem if they decide to stop supporting this framework [14].

2.1.2 Flutter

Google also created a mobile application SDK (software development kit) called Flutter, which creates native interfaces on both iOS and Android quickly. It is free and open source. Flutter also has fast development because like React Native, it has Hot Reload. Hot Reload allows the emulator that the developer is using to update automatically and quickly after code changes without needing to recompile every file [6]. Like React Native, Flutter enables cross-platform development. It also provides its own widgets, which are created with a high-performance rendering engine. The main benefits of Flutter are that there is Hot Reload, the code runs on both Android and iOS platforms, and Flutter apps look the same on older operating systems. The problems with Flutter are that it is still in beta, the libraries and support still

aren't as rich as native Android, and it still isn't widely supported by Continuous Integration services such as Jenkins [5]. Continuous Integration services are services that developers use to merge code into a single repository and verify that the applications build correctly. Some apps that use Flutter are Alibaba, Google Ads, and Hamilton Musical [15].

2.1.3 Native Android

Native applications are platform-specific apps that are built using a programming language that is specific to that platform. Since native applications are developed for a particular platform, they are fully able to access features specific to that device such as the camera, GPS, and Bluetooth. Native applications work on the devices operating system, meaning that they require total access to all of that devices hardware and functionality. Native android provides developers with a standardized software development kit with tools, libraries, and documentation. The benefits of using native Android is that the app will perform faster, work offline, and have a recognizable look and feel for users. The drawbacks are that there is no flexibility or cross-platform development and requires frequent app updates [8]. Since there is no cross-platform development, the app will only run on Android devices and will not work on iOS devices. This is the option that the team will take because it was specified by the client and because it is the most stable between the three options.

2.1.4 Overall Recommendation and Conclusion

It is recommended to use native Android for this project because of a variety of reasons. The largest reason is that the client specified that they wanted a native Android application. React Native is a solid option because developing with it would be fast, but it is risky because Facebook could decide at any time to stop supporting the framework. Flutter is very risky because it doesn't have as large of a community as native Android. Due to the smaller community, if the group runs into a problem and needs help, there will be less resources that the team can reference. Furthermore, Flutter is currently only in beta, meaning that it does not have as many libraries as native Android. Overall, native Android is the best and recommended option because of its stability, large community and vast resources, and the fact that the client specified that they wanted a native Android application.

2.2 Automated Testing

Automated testing is useful for ensuring the quality of the application. In order to evaluate the quality of an automated testing framework, this paper will look at criteria such as how fast the framework is, how easy it is to use, and how much it can test the app.

2.2.1 Robotium

Robotium is an open-source framework that is used for automated testing with Android applications. Developers can use Robotium in order to test various scenarios for Android activities. Robotium enables developers to provide automated testing for UI test cases with Android applications. The benefits of Robotium are that it is easy to write code and it isn't necessary to spend a lot of time to create solid test cases. Additionally, test cases can be written without needing extensive knowledge of the application. Robotium automatically finds views, has automatic timing and delays, is able to make its own decisions (such as when to scroll the application), and integrates smoothly with Maven. The drawbacks are that it can only handle one application at a time and cannot simulate clicking on the software keyboard, meaning that that developers will have to write code to enter the text. Robotium also can't interact with status bar notifications

and can be relatively slow, especially on older devices [11]. Robotium seems like the best testing framework because it can test so many Android features.

2.2.2 Appium

Another automated testing framework is Appium. Appium is useful because it supports cross-platform test automation, meaning that it can test both iOS and Android applications. Additionally, it supports tests on any framework using any language. The developers behind Appium believe that testers shouldn't have to recompile the app or modify it to test it, testers shouldn't be locked into specific languages or frameworks in order to write and run tests, automation frameworks shouldn't reinvent the wheel, and automation should be open source. Appium uses a standard API across all platforms, meaning that testers won't have to modify code or recompile the app [1]. Additionally, Appium is free, open source, has a well supported and active group, and should be relatively future proof. The drawbacks are that it doesn't support any intelligent waiting and there is limited support for gestures and lacks support for older Android versions [10].

2.2.3 UI Automator

UI Automator is a UI testing framework that is used to test the UI in mobile applications. The framework includes a viewer that allows the tester to inspect the layout hierarchy. It also has an API (application programming interface) to retrieve information regarding the current state of the device and perform operations on it. Additionally, the framework has APIs that support cross-app UI testing [17]. UI Automator is also supported by Google. The problem with this framework is that it only supports native Android apps [9]. Furthermore, UI Automator only works on some Android devices and doesn't support webview, meaning that it isn't possible to easily access Android objects [3].

2.2.4 Overall Recommendation and Conclusion

This paper recommends Robotium for a variety of reasons. Robotium is easy to use and will not require a significant amount of time for the team to use. Also, the framework should be able to test every functionality that the team will want to verify. While writing tests with Appium seems fast because the app does not have to be compiled, it is not recommended because it doesn't support many gestures that the team might want to test. Additionally, it lacks support for older Android devices, which is a problem because the team will want to test as many devices as possible. UI Automator also seems like a solid option, but it only support Android devices with API level 16 or above and doesn't support webview. This framework lacks support for a decent amount of Android devices and objects. Overall, Robotium is the recommended framework because it is easy to use and can test the most features on a wide range of Android devices.

2.3 Languages

When it comes to languages, there are several different options. This paper will look at the ease of use, functionality, and the popularity of each language in order to determine what language will be used.

2.3.1 Java

Java is an object oriented programming language with many benefits. There are many Java libraries in the Android SDK, and apps built using Java build relatively fast and are fairly light [7]. There is also a very large community of developers who use Java when developing Android apps, which means that it is easier to find resources for help [12].

However, there are some cons with Java. Java has limitations such as that it requires a larger amount of memory. It is also a verbose language, meaning that it requires more code. Having more code can potentially lead to a higher amount of bugs [7]. Java seems like the best option because of its wide popularity and the fact that there will be a lot of resources that the team can turn to for help.

2.3.2 Kotlin

Kotlin was initially created by the developers from JetBrains with the purpose of adding more modern features to Java that will be useful for mobile development. The benefits of using Kotlin are that it is easy to use and is fairly concise, which means that there are less opportunities for bugs in the code [7]. Kotlin is much more concise than a language like Java because many methods such as getters, setters, and toString() don't need to be specified because Kotlin generates them automatically. Java is currently the most used language for Android development, but this may change since Kotlin was declared by Google to be Android's official language [12]. Additionally, Kotlin works with Java, meaning that developers can create new modules using Kotlin and those modules will be able to work alongside the already existing Java code. Kotlin is compatible with every Java library and framework. The drawbacks of using Kotlin are that it has a relatively steep learning curve due to the nature of the concise syntax and it is usually slower to compile than Java. Furthermore, there is a smaller community of developers who use Kotlin, meaning that there are less resources for help and it is more difficult to find answers. Additionally, Android Studios compilation typically runs slower when there are Kotlin modules mixed in with Java code as opposed to just Java [7].

2.3.3 LUA

LUA is an open source scripting language. It is lightweight, fast, and powerful. It is used in applications such as Warcraft and Angry Birds. Developers who code using LUA also use the Corona framework which provides some advantages as well. It is a cross-platform framework meaning that the code in LUA will work for many different platforms. Additionally, it is possible to call native libraries such as Java [4]. LUA is also able to run on an emulator without needing to be compiled first, limiting the down time in development. However, LUA isn't supported by Android Studio, meaning that developers will need to find a different text editor. LUA is also fairly limited when it comes to Android app functionality. As a result, LUA is more ideal for mobile apps that are fairly simple, which likely isn't the case for this project [16].

2.3.4 Overall Recommendation and Conclusion

Java is the recommended language for this project. There is a large community of developers who use this language, which means that there will be a lot of resources that the team will be able to reference. Furthermore, Java apps are fairly fast and light. Kotlin also seems like a great option because it is easy to use and concise. However, Kotlin has a small community, which means that the team might not be able to find help for certain problems that they may face. Also, Kotlin has a steep learning curve and is slower to compile. LUA is a powerful language, but it doesn't seem appropriate for this application. This language is limited when it comes to Android functionality and is typically used for simple mobile apps and games. Since this project won't be simple and is not a game, LUA doesn't seem like an appropriate language for the team. Overall, Java is the recommended language because of its large community and the fact that Java apps are fairly fast.

2.4 Conclusion

There are a lot of different options when it comes to mobile application frameworks, automated testing, and languages. For this project, it is recommended to use native Android, Robotium, and Java. Native Android is recommended because it is stable, has a large community, and the client specified that they wanted a native Android application. Robotium is the recommended automated testing framework because is easy to use and should be able to test all the functionality that the team will want to verify. Lastly, Java is the recommended language because it has a large community, Java apps are fairly fast and light, and it seems the most appropriate for this type of application.

REFERENCES

[1] An Introduction to Appium: The Best Open Source Mobile App Automation Tool , *Hacker Noon*, 11-Nov-2017. [Online]. Available: https://hackernoon.com/an-introduction-to-appium-the-best-open-source-mobile-app-automation-tool-940f5eaabae9. [Accessed: 02-Nov-2018].

[2] Appium, *Appium*.[Online]. Available: http://appium.io/. [Accessed: 02-Nov-2018].

[3] J. Chakraborty, Top 5 Android Testing Frameworks, *Linkedin*, 20-Aug-2015. [Online]. Available: https://www.linkedin.com/pulse/top-5-android-testing-frameworks-jaybrata-chakraborty/. [Accessed: 02-Nov-2018].

[4] Corona, *Corona Labs*.[Online]. Available: https://coronalabs.com/. [Accessed: 02-Nov-2018].

[5] A. Czapla, Flutter in Mobile App Development Pros Risks for App Owners, *Droid On Roids*.24-Apr-2018. [Online]. Available: https://www.thedroidsonroids.com/blog/flutter-in-mobile-app-development-pros-and-risks-for-app-owners [Accessed: 02-Nov-2018].

[6] Flutter, *Flutter*.[Online]. Available: https://flutter.io/. [Accessed: 02-Nov-2018].

[7] K. Jackowski, Kotlin vs. Java: Which One You Should Choose for Your Next Android App, *Netguru*, [Online]. [Online]. Available: https://www.netguru.co/blog/kotlin-java-which-one-you-should-choose-for-your-next-android-app. [Accessed: 02-Nov-2018].

[8] U. Khan, The Pros and Cons of Native Apps, *Clutch*, 12-Jun-2018. [Online]. Available: https://clutch.co/app-developers/resources/pros-cons-native-apps. [Accessed: 02-Nov-2018].

[9] N. Minaev, The Top 5 Android UI Frameworks for Automated Testing, *Sauce Labs*, 13-Sep-2017. [Online]. Available: https://saucelabs.com/blog/the-top-5-android-ui-frameworks-for-automated-testing. [Accessed: 02-Nov-2018].

[10] Mobile Testing Appium Framework *TutorialsPoint*. [Online]. Available: https://www.tutorialspoint.com/mobile_testing/mobile_testing_appium_framew [Accessed: 02-Nov-2018].

[11] Mobile Testing Robotium Framework, *TutorialsPoint*.[Online]. Available: https://www.tutorialspoint.com/mobile_testing/mobile_testing_robotium_fram [Accessed: 02-Nov-2018].

[12] J. Paul, Java vs. Kotlin - Which Language Should Android Developers Start With, *DZone*. [Online]. Available: https://dzone.com/articles/java-vs-kotlin-which-language-android-developer-sh. [Accessed: 02-Nov-2018].

[13] React Native, *Facebook*. [Online]. Available: https://facebook.github.io/react-native/. [Accessed: 02-Nov-2018].

[14] React Native - Pros and Cons Of Facebook's Framework, *Netguru* , 22-Aug-2017. [Online]. Available: https://www.netguru.co/blog/react-native-pros-and-cons. [Accessed: 02-Nov-2018].

[15] Showcase, *Flutter*. [Online]. Available: https://flutter.io/showcase/. [Accessed: 02-Nov-2018].

[16] A. Sinicki, I want to develop Android Apps - What languages should I learn?, *Android Authority*, 24-Dec-2017. [Online]. Available: https://www.androidauthority.com/develop-android-apps-languages-learn-391008/. [Accessed: 02-Nov-2018].

[17] UI Automator, *Android Developers*. [Online]. Available: https://developer.android.com/training/testing/ui-automator. [Accessed: 02-Nov-2018].