Cloud Application Development Continual Assessment (2023)

An Evaluation of Real Device Testing on Cloud for Cross-Platform Mobile Web Applications

Yit How Wong

Munster Technological University, Tralee, Kerry, V92 HD4V, Ireland

Abstract

In recent years, the demands for software applications to run on multiple different devices has been soaring. The pressure of delivery market demands in a short amount of time has caused companies to often neglect software testing. Testing mobile applications also has an additional layer of difficulty due to the difference in hardware from developing and running environment. With the annual release of new mobile devices and operating systems, it is unfeasible for company to own and maintain a large pool of mobile devices, just for the purpose of testing.

This paper investigates the implementation of cloud tool AWS Device Farm to ease the process of testing for cross-platform mobile web applications. The goal is to access cloud tools that could ease the long process of testing, using a combination of manual and automated testing.

*Keywords:* Mobile Web Applications Testing; AWS Device Farm; Automated Testing; Cloud Real Device Testing

1. Introduction

The market of consumer hardware has been saturated with various devices. It is no longer safe to assume that a consumer will be using traditional desktop or laptop computer to browse a company’s website or application. This increases the demand for developer to develop both desktop and mobile devices compatible applications as in theory, it is more efficient to build one application that serves the same functionality. However, the trend of companies releasing new devices annually or even shorter period time, has increased the limited resource on these devices, but raised concerns of compatibility in cross-platform environment. Software developers are also pressured to continuously produce new features and ready-to-ship software increments in short time, resulting the neglect of application testing by developers. Thus, an automation support for cross-platform web application development, quality assurance, automated testing and deployment process are needed.

The study will investigate the tools offered by AWS Cloud services that can be utilized for an ease implementation of real device testing for cross-platform mobile web applications. The proposed application will incorporate a static JavaScript website where a user can play the Pong game on the desktop and mobile devices browser. The control scheme of the Pong game will change accordingly to the browser environment, where the desktop user controls the paddle with keyboard, and touch control scheme for the mobile device user.

Overall, this study aims to seek the implementation of real device testing on cloud and discuss its benefits and cons.

1. Literature Review
   1. Introduction to Cross platform applications

Cross platform application is a computer software that is designed to work in several computing platforms. These applications are compatible with multiple operating systems and can, therefore run on any designated device. The idea of this development is to create and maintain a single application for all the platforms (Rahul Raj C.P, 2012). Cross platform application is usually developed using a portable framework language and tools, such as Java, Unity, React Native, Xamarin, Node.js, Flutter, PhoneGap and etc (Klubnikin, 2017).

It is important to note that the term “cross platform application” does not exclusively refer to cross platform mobile application. This is because the most common type of cross platform development now is cross platform mobile application development (Samsukha, 2022). Another example of cross platform application development is video game that can be deployed across multiple platforms. Section below discusses the three types of mobile applications that are usually referred to cross platform application and their differences.

* 1. Mobile application development

Cross-platform application development is an approach that allows a company to build single mobile applications that runs smoothly on several operating systems (Kotlin, 2023). It has gained significant attention in recent years due to the increasing demand for applications to run for multiple platform such as iOS, Android, Windows and others. According to the Fortune Business Insights (Fortune Business Insight, 2022), the global market share is projected grow 10.5% annually, further emphasizes the need for it. One recent statistic also revealed that 60.67% of all web traffic came through mobile phones, where in 2012, the figure was only at 10.88% (Orbelo, 2023) (GlobalStats StatCounter, 2023). A mobile application can be separated into three types: Native Application, Hybrid Application and Web Application.

Native applications indicate the software program built for use on particular platform or device. On mobile device, they are installed through an application store such as Google Play or Apple’s App store. Since they are developed specifically for one platform, they can take full advantage of the device features, as like the camera, GPS, accelerometer, the list of contact and so on (Budiu, 2013). The development of native applications requires a specialized knowledge in language for certain device. For example, Java and Kotlin are generally used for developing native Android mobile application (Geeks for Geeks, 2022), even more commonly, still requires specific training for mobile development.

Hybrid applications are referred to native applications with a web browser embedded inside them. Like native applications, a user can download and install hybrid application from application stores. The internal structure of hybrid applications is different than a native application, in which the formal use HTML5, CSS, JavaScript and native shell to behave more like web applications (aws amazon, n.d.). Once the users download the application, the shell connects to designated web server through a browser that is embedded in the application. The browser on the backend is usually invisible to the end user. Companies frequently create hybrid application as wrapper their existing website in an effort to establish a presence on the app store, without expending significant effort creating a new native app. Since the same HTML code components can be reused across various mobile operating systems, hybrid applications allow good efficiency in cross-platform development and thereby reducing development cost significantly (Budiu, 2013).

Web applications are referred to as an application that can be accessed from the browser of a desktop or mobile device. These applications do no need to be downloaded as they are usually stored on a remote server and delivered over internet through the browser interface. They are similar to a website, but not entirely. Web application is intended to provide interaction between the user and the application. They can be easy as displaying latest news articles or weather updates, to more complex tasks such as online shopping, social networking (Tuama, 2022). An example would be browsing social media platform such as Instagram, where the user could access the application through mobile device browser, and perform interaction activities such as reading posts, reels or sending messages.

In summary, companies have three options to develop a mobile application or use a combination of all. To reach the widest possible base within limiting resources, companies must evaluate factors such as time to market, customer requirements, marketing strategy, device features, offline functionality, performance, complexity, and maintainability before venturing into development (aws amazon, n.d.) (Budiu, 2013).

Diagram

Description automatically generated

Fig. 1. Figure of cross platform capability for different mobile applications (Paulo R. M. de Andrade, 2015).

Table 1. Summary of differences: Native application, Hybrid Application, Web Application (aws amazon, n.d.)

|  |  |  |  |
| --- | --- | --- | --- |
| Characteristics | Native App | Hybrid App | Web App |
|  |  |  |  |
| Usage | Required to be installed on device | Required to be installed on device | Can access directly from web browser |
| Internal working | Client code written in technology and language specific to the device or platform | Client and browser code wrapped in a native shell or container | Client code in the browser communicates |
| Native device features | Accessible | Accessible | Limited access to certain features and depends on browser capability (Stangarone, 2012) (Ionic, 2023) |
| User experience |  |  |  |
| Access | One-step access with offline features | One-step access with offline features | Limited by browser capability and network connectivity |
| Performance | Performance can be optimized for device or platform | Faster, but may consume more battery power | Slowest and least responsive |
| Development | Most expensive and h |  |  |
|  |  |  |  |
|  |  |  |  |

These applications usually utilize network connection to access remote computing resources such as data accessing with API (aws, 2023). For mobile web applications, these applications are mainly developed using the same tools and language as regular web pages, but with touch-friendly user interface. This has made it possible for developers to develop apps for various platforms quickly.

* 1. End to end testing

End-to-end (E2E) testing is essential for the seamless functioning of mobile applications. According to a blog post by Shreya Bose (Bose, 2023), End-to-end testing is a technique that verifies the entire software application from start to finish, including the systems, components, and integrations involved in the application’s workflow. During E2E testing, the software is tested from the end user’s perspective, in which test cases that simulate real user scenario are designed, including the user interface, backend services, databases and network communications, to validate the application’s overall behavior. The goal is to simulate a real user scenario to test the functionality and performance under

* 1. Testing Mobile Applications

Most mobile web applications are developed with a combination of HTML5, JavaScript, and CSS (Jobe, 203). Therefore, regular application testing types and techniques can also be applied to mobile web applications. These include a combination of manual and automated testing that covers functionality, usability, interface, database, performance, security, regression and compatibility (Unadkat, 2023). Fortunately, web application testing framework has matured that allows automated test to be performed. Selenium is well known example of open-source testing framework that enables testers to write automated test in various programming language for functionality testing (Selenium , 2023). However, testing mobile web applications has an additional layer of difficulties as opposed to the traditional, due to the wide variety of devices, operating systems, and browsers that exist. There are ways QA can test their mobile application discussed as below:

**Virtual device**: A common approach is to use emulators or simulators. These tools allow testers to replicate different mobile devices and operating systems in a virtual environment. For example, Android Studio has built in simulator allows creating instances of Android virtual devices that mimic Android version and hardware characteristic of the simulated device (Android for Developers, 2023). However, it is impossible for these tools to perform the same as real devices (McPeak, 2017). They also have a reputation for producing false negative and positive results (Indium, 2017) (Kam, 2023). For this reason, testing on real devices is essential to replicate highly accurate representation.

**Real mobile device**: Testing mobile web applications on an actual device provides an edge compared to a virtual device. It gives real time technical performance such as load times, response times, and overall responsiveness. Test network connectivity, battery life, and power consumption are also important metrics for resource intensive web applications. Most importantly, real devices allow for more accurate user experience testing, as users interact with the application in a way that emulators and simulators cannot replicate. With that said, real device testing is best reserved for later in the development cycle, especially entering the integration phase., where integration, functional and non-functional testing are needed (Kinsbruner, 2020).

**Real Device Testing Cloud**: The recent emerging trends and technologies in cloud computing has enabled the option of cloud based mobile application testing, which is built on top of the mobile device infrastructure owned by the cloud provider (Chuanqi Tao, 2017). This practice of running mobile application tests in the cloud instead of an on-premises environment, has allowed developers or companies to run their software test on various mobile devices without the need for physical hardware. Therefore, this greatly reduces the initial cost of device testing in a scalable environment. Additionally, cloud hosted testing occurs in real time, so results are immediately available.

1. Diagram

   Description automatically generatedMethodology

Fig. 2. Figure of Architecture Diagram

The design philosophy for this study is to create a cross platform mobile web application that integrates several cloud tools that were reviewed above, that supports a combination of manual and automating testing during the software development lifecycle. The web application is a Pong game written in the language of HTML, CSS, and JavaScript. The application is capable of performing browser identification based on the user agent of the browser and changing the control scheme for the paddle accordingly. On the desktop browser environment, the paddle will be controlled by using the UP and DOWN, or W and S for upward and downward movement. On a mobile device, the paddle will be controlled using the touch screen function where the paddle direction movement is inverted after a touch screen action. The code repository is hosted and managed on GitHub. To deploy the web application onto cloud environment, an automate continuous delivery pipeline tool, AWS CodePipeline, is used to automate the software release process. An AWS CodePipeline service role is automatically established, or can be created manually, during the pipeline setup stage. By default, the service role provides full access to many of the AWS services such as the compute services: EC2, Elastic BeanStalk, CloudFormation; storage services: S3; and monitoring services: CloudWatch. Then, the GitHub authentication is performed to establish a connection between CodePipeline, managed by AWS CodeStar, to retrieve the code artifacts. Lastly, the deployment destination is set to AWS S3 bucket as the Pong game is a static website, that does not rely on server-side processing used commonly in dynamic website. Public access policy is set to publicly accessible on the S3 Bucket policy permissions, for public accessing on the Internet.

AWS Device Farm is selected for the automated and manual testing tool for this study. It allows seamless connection to multiple mobile devices, instead of creating complex testing infrastructure. With AWS free tier account, it provides 1000 minutes of automated testing and remote access. To request remote access on device farm, one simply has to choose a device from a huge selection of device pool. Once the session connection is established, the Pong game website is accessible via inputting the index.html object URL stated in the S3 bucket. This has allowed functional testing to be carried out.

To integrate automated testing in AWS Device Farm, a simple unit test case using Mocha JS library is created and specified locally in the Node.js project. Then, the Node.js project is packaged into a tarball, which contains all the code and dependencies, and uploaded to the Device Farm test run. The automated testing is configured to Appium Node.js testing framework, which is an open-source test automation framework for mobile applications. A pre-written Appium test script has been used for test environment setup configuration which include:

* Installation of stated version of Node, Appium, and WebDriverAgent which is an iOS remote server test automation framework adopted by Appium (Documentation Eggplant Software, 2022).
* Unpackage and install the uploaded node modules.
* Test environment setup for Appium
* Test suite execution

The unit test is specified in the test suite execution section for automated testing. The automated test is then run on two dynamic device pools with three random Android and iOS mobile devices respectively.

1. Results/Data/Findings

The process of establishing the AWS CodePipeline was straightforward. AWS CodePipeline is able to detect any changes made to the GitHub code artifacts within minutes and deploy the application to S3 bucket. The static website is accessible via entering the static website’s object URL. Using a desktop browser, the Pong game is playable with the control scheme using keyboard.

The manual testing of web application is performed successfully on multiple mobile devices with a variety version of Android and iOS. 150 minutes was allocated for each session of remote access, which provided sufficient testing time. The Pong game is accessible via the mobile device browser by entering the website’s object URL. Next, the game functionality test passed successfully, by being able to control the paddle using the touch control scheme. Lastly, each of remote access session provided a log that details debug information during environment setup, a MP4 video files that was recorded during the session and a TCP log that details the network traffic.

For automated testing, all testing including the unit testing has passed on iOS devices. Unfortunately, unit testing is unsuccessful to be executed on all tested Android devices.

Text

Description automatically generated

Fig. 3. Figure of error log produced during the attempt of automating unit test on Android devices.

Overall, cross platform mobile web application testing is simplistic to setup with AWS Device farm in the cloud.

1. Discussion

The outcome of this study suggested that the development of cross platform web applications can be supported on AWS cloud infrastructure. Using AWS CodePipeline, a continuous delivery pipeline for fast and reliable updates can be established easily. AWS CodePipeline provides seamless integration with other AWS services, which could be extended for future work, this includes but not limited to:

* AWS CloudFormation for cloud infrastructure provisioning.
* AWS Elastic Beanstalk for deploying and scaling web applications and service, with integrated capacity provisioning, load balancing and health monitoring.
* AWS Device Farm for configuring automated test actions to run tests on multiple devices within the continuous integration flow.
* AWS Lambda to execute customized lambda function within the pipeline flow.

With AWS Device Farm, real device testing can be performed parallelly across multiple platforms without having to purchase and maintain the infrastructure. Therefore, greatly reducing the cost of device testing. Since the testing are performed on actual physical device, an accurate representation of the user interaction can be represented, supported by log data such as battery usage, network traffic, CPU usage. With remote access, testers can reproduce issues manually on the customer reported device. Videos, logs, and performance data are generated which allows the developers to debug their application more easily and deploy with the said pipeline. With automated testing, the testers can customize the provided test suite, and allows unit testing to run on the cloud. Using the Appium framework integrated with AWS Device Farm, automation test written in Appium to simulate real user scenario can be run parallel to fully integrate End-to-end testing in the software lifecycle.

Unfortunately, the reason for the failed automated unit test on Android device cannot be specified after an investigation into the test specification log produced.

1. Conclusion

This study presented an approach of real device testing using the toolkits within the AWS cloud environment. Testing should be carried out in every phase during the software lifecycle to ensure software is of high quality and meets requirements. As we all know, testing early during the lifecycle helps to identify defects and issues early in the development. Therefore, it would be wise to implement them within the development pipeline and automate them as much as possible. Although automated mobile testing can be performed on emulators or real devices, doing this on cloud premises offers the benefits that cover cost, scalability, flexibility, and faster testing. For further study, a monitoring tool could be implemented within the pipeline, to monitor the health of the web application. An understanding of Appium testing framework is also valuable for creating automated test suite that can replicate use case scenario, to fully integrate the principles of End-to-end testing.

1. Acknowledgements

I would like to thank Andrew Shields and Therese Enright whose assistance and guidance was appreciated during this study.

# References

Android for Developers, 2023. *Run apps on the Android Emulator.* [Online]   
Available at: https://developer.android.com/studio/run/emulator  
[Accessed April 2023].

aws amazon, n.d. *What's The Difference Between Web Apps, Native Apps, And Hybrid Apps?.* [Online]   
Available at: What's The Difference Between Web Apps, Native Apps, And Hybrid Apps?  
[Accessed April 2023].

aws, 2023. *What is Mobile Application Development?.* [Online]   
Available at: https://aws.amazon.com/mobile/mobile-application-development/  
[Accessed 2023].

Bose, S., 2023. *What is End To End Testing?.* [Online]   
Available at: https://www.browserstack.com/guide/end-to-end-testing  
[Accessed April 2023].

Budiu, R., 2013. *Mobile: Native Apps, Web Apps, and Hybrid Apps.* [Online]   
Available at: https://www.nngroup.com/articles/mobile-native-apps/  
[Accessed April 2023].

Chuanqi Tao, J. G., 2017. On building a cloud-based mobile testing infrastructure service system. *The Journal of Systems and Software,* Volume 124, pp. 39-55.

Documentation Eggplant Software, 2022. *WebDriverAgent vs Appium.* [Online]   
Available at: https://docs.eggplantsoftware.com/mobilegateway/webdriver-vs-appium/#:~:text=Appium%20adopted%20WebDriverAgent%20to%20implement,the%20original%20version%20from%20Facebook.  
[Accessed April 2023].

Fortune Business Insight, 2022. *Smartphone Market Size, Share & Covid 19 Impact Analysis.* [Online]   
Available at: https://www.fortunebusinessinsights.com/industry-reports/smartphone-market-100308  
[Accessed April 2023].

Geeks for Geeks, 2022. *Top Programming Languages for Android App Development.* [Online]   
Available at: https://www.geeksforgeeks.org/top-programming-languages-for-android-app-development/  
[Accessed April 2023].

GlobalStats StatCounter, 2023. *Desktop vs Mobile Market Share Worldwide.* [Online]   
Available at: https://gs.statcounter.com/platform-market-share/desktop-mobile/worldwide/#yearly-2011-2022  
[Accessed April 2023].

Indium, 2017. *Real Devices Vs Emulator/Simulator in Mobile Testing: Which is Best?.* [Online]   
Available at: https://huddle.eurostarsoftwaretesting.com/mobile-testing-with-simulatorsemulators-vs-real-devices/  
[Accessed April 2023].

Ionic, 2023. *Accessing Native Features.* [Online]   
Available at: https://ionic.io/enterprise-guide/native  
[Accessed April 2023].

Jobe, W., 203. Native Apps Vs. Mobile Web Apps. *International Journal of Interactive Mobile Technologies (iJIM),* pp. 27-32.

Kam, K., 2023. *Mobile Emulators vs. Real Devices.* [Online]   
Available at: https://qualitestgroup.com/insights/white-paper/mobile-emulators-vs-real-devices/  
[Accessed April 2023].

Kinsbruner, E., 2020. *When to Test On Mobile Devices: Real vs. Virtual.* [Online]   
Available at: https://www.perfecto.io/blog/test-on-mobile-devices#:~:text=When%20to%20Test%20On%20Real,part%20of%20the%20CI%20phase.  
[Accessed April 2023].

Klubnikin, A., 2017. *Cross-platform vs Native Mobile App Development: Choosing the Right Development Tools for Your Project.* [Online]   
Available at: https://andrei-klubnikin.medium.com/cross-platform-vs-native-mobile-app-development-choosing-the-right-dev-tools-for-your-app-project-47d0abafee81#:~:text=A%20cross%2Dplatform%20application%20is,%2C%20smartwatch%2C%20and%20connected%20TV.  
[Accessed April 2023].

Kotlin, 2023. *What is cross-platform mobile development?.* [Online]   
Available at: https://kotlinlang.org/docs/cross-platform-mobile-development.html#different-approaches-to-mobile-app-development  
[Accessed April 2023].

McPeak, A., 2017. *Choosing Between Emulator vs Simulator vs Real Devices for Testing.* [Online]   
Available at: https://smartbear.com/blog/2017/emulators-simulators-real-devices-testing/?lang=de-de#:~:text=While%20simulators%20and%20emulators%20will,performed%20on%20a%20real%20device.  
[Accessed April 2023].

Orbelo, 2023. *WHAT PERCENTAGE OF INTERNET TRAFFIC IS MOBILE?.* [Online]   
Available at: https://www.oberlo.com/statistics/mobile-internet-traffic#:~:text=As%20of%20February%202022%2C%2050.59,49.41%20percent%20coming%20from%20desktops.  
[Accessed April 2023].

Paulo R. M. de Andrade, A. B. A., 2015. Cross Platform App A Comparitve Study. *International Journal of Computer Science & Information Technology (IJCSIT) ,* 7(1), pp. 33-40.

Rahul Raj C.P, S. B. T., 2012. A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach. *2012 Annual IEEE India Conference (INDICON),* Issue A study on approaches to build cross-platform, pp. 625-629.

Samsukha, A., 2022. *Best Cross-Platform App Development Frameworks in 2023.* [Online]   
Available at: https://www.emizentech.com/blog/cross-platform-app-development-frameworks.html  
[Accessed April 2023].

Selenium , 2023. *Selenium automates browsers..* [Online]   
Available at: https://www.selenium.dev/documentation/  
[Accessed April 2023].

Stangarone, J., 2012. *6 “native” features you can use with mobile web apps.* [Online]   
Available at: https://www.mrc-productivity.com/blog/2012/01/6-%E2%80%9Cnative%E2%80%9D-features-you-can-use-with-mobile-web-apps/  
[Accessed April 2023].

Tuama, D. Ó., 2022. *What is the Difference Between Web App & Mobile App?.* [Online]   
Available at: https://codeinstitute.net/ie/blog/web-app-vs-mobile-app/  
[Accessed April 2023].

Unadkat, J., 2023. *7-Step QA Checklist for Website Testing.* [Online]   
Available at: https://www.browserstack.com/guide/how-to-perform-website-qa-testing  
[Accessed April 2023].