



Faculty of Engineering and Technology
Master in Software Engineering Program

Master Thesis

Cloud Testing Frameworks for Mobile Apps: A Comparative Study

اختبار برمجيات المحمول في السحاب: دراسة مقارنة

Supervisor: Dr. Samer Zein

Student Names: Ruba Esawi

Student ID: 1185294

"This Thesis was submitted in partial fulfillment of the requirements for the Master Degree in Software Engineering from the Faculty of Engineering and Technology at Birzeit University, Palestine"

May 5, 2021



Cloud Testing Frameworks for Mobile Apps: A Comparative Study

Thesis

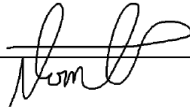
Author : Ruba Esawi

Approved by the thesis committee:

Dr. Samer Zein, Birzeit University
(Chairman of the Committee)



Dr. Mamoun Nawahdah, Birzeit University
(Member)



Dr. Radi Jarar: Birzeit University
(Member)



Date approved:

Aug 8, 2021

Abstract

In recent years, mobile devices have been ubiquitous, and mobile apps development has become a considerably popular field. In addition, mobile apps target various contexts and domains including critical ones. Because of this popularity, software testing techniques are important to ensure the high quality and robustness of mobile apps. One of the main challenges that face test engineers when approaching mobile app testing is the large fragmentation in screen sizes, underlying hardware, and operating system APIs. To overcome these issues, many mobile app vendors turned to cloud testing that provides the benefits of deploying and testing their apps on a large set of real mobile devices. Nowadays, the number of cloud testing frameworks for mobile apps is growing. However, little research that compares cloud testing frameworks is available. This research aims to conduct a systematic comparative study for the most popular mobile cloud testing frameworks to help practitioners choose between them. To achieve this goal of the research , we have selected the two most popular mobile cloud testing frameworks for Android (Perfecto and Xamarin). The comparison is based on the technical requirement, performance, and usability parameters. To achieve a good result, the same mobile app was used. The experiment conducted as part of this research revealed that Perfecto outperformed Xamarin in terms of performance and usability.

الملخص

في السنوات الأخيرة، انتشرت الأجهزة المحمولة في كل مكان، وأصبح تطوير تطبيقات الأجهزة المحمولة مجالاً شائعاً إلى حد كبير. تستهدف تطبيقات الأجهزة المحمولة سياقات ومجالات مختلفة بما في ذلك السياقات والمجالات الهامة. نظراً لهذه الأهمية، تعد تقنيات اختبار البرامج مهمة لضمان الجودة العالية والمتانة لتطبيقات الأجهزة المحمولة. اختبار البرمجيات يعتبر أحد الخطوات الرئيسية في تطوير البرمجيات، وهو عبارة عن عملية فحص للتأكد من متطلبات المستخدم والتحقق منها.

أحد التحديات الرئيسية التي تواجه مهندسي الاختبار عند الاقتراب من اختبار تطبيقات الأجهزة المحمولة هو التجزئة الكبيرة في أحجام الشاشة، والأجهزة الأساسية، وواجهات برمجة تطبيقات نظام التشغيل. للتغلب على هذه المشكلات، تحول العديد من بائعي تطبيقات الأجهزة المحمولة إلى اختبار السحابة الذي يوفر فوائد نشر واختبار تطبيقاتهم على مجموعة كبيرة من الأجهزة المحمولة الحقيقية.

يتزايد عدد أطر اختبار السحابة لتطبيقات الأجهزة المحمولة. ولكن اختيار الأداة الأكثر ملاءمة لخدمة تطبيق الأجهزة المحمولة يعد عملية صعبة حيث لا يوجد دليل أو بحث تساعد المستخدم على تحديد الخيار الأفضل من بين الأدوات المتاحة.

يهدف هذا البحث إلى إجراء دراسة مقارنة منهجية لأطر اختبار السحابة المتنقلة الأكثر شيوعاً لمساعدة الممارسين على الاختيار فيما بينها. لتحقيق هذا الهدف من الاقتراح، اخترنا في البداية اثنين من أكثر أطر اختبار السحابة المحمولة شيوعاً لنظام **Android (Perfecto and Xamarin)**. تعتمد المقارنة على معايير إعادة الطاقة الفنية والأداء وقابلية الاستخدام. لتحقيق نتيجة جيدة، تم استخدام نفس تطبيق الهاتف المحمول. كشفت التجربة التي أجريت كجزء من هذا البحث أن **Perfecto** تفوقت على **Xamarin** من حيث الأفضلية وسهولة الاستخدام.

Table of Contents

ABSTRACT	II
الملخص	IV
LIST OF FIGURE	VII
LIST OF TABLE	IX
ABBREVIATIONS	X
ACKNOWLEDGMENT	XI
CHAPTER 1 INTRODUCTION	1
1.1 INTRODUCTION	1
1.2 MOTIVATION	4
1.3 RESEARCH PROBLEM	4
1.4 RESEARCH QUESTIONS	5
1.5 REPORT STRUCTURE	5
CHAPTER 2 BACKGROUND	7
2.1 ANDROID PLATFORM	7
2.1 HIGH FRAGMENTATION OF ANDROID DEVICES	9
2.3 AUTOMATED TESTING IN THE CLOUD	11
CHAPTER 3 LITERATURE REVIEW	14
3.1 INTRODUCTION	14
3.2 SEARCH METHOD	15
3.2.1 Method	15
3.2.2 Source Databases	15
3.2.3 Search Strings:	16
3.2.4 Inclusion, Exclusion Criteria and selection process	16
3.3 RELATED WORK	19
3.3.1 Mobile apps testing method	19
3.3.2 Evaluation Methods testing in the cloud	21
3.3.3 Contribution Facts for testing in the cloud	23
3.4 SUMMARY	27
CHAPTER 4: RESEARCH METHODOLOGY	29
4.1 SELECTING FRAMEWORKS	31
4.2 SELECTING APPLICATION	33
4.3 IDENTIFYING CRITERIA	34
4.4 DATA COLLECTION	35
4.5 DATA ANALYSIS	36
4.6 EXPERIMENT DESIGN	37
4.6.1 Experiment Hypothesis	37
4.6.2 Experiment Procedure	40
CHAPTER 5 EXPERIMENT EVALUATION	42
5.1 EXPERIMENT	42
5.2 USABILITY EXPERIENCE	42
5.2.1 Selection of Participants	43
5.2.2 Scenarios based exercise	44
5.2.3. Test Location	44

5.2.4 Usability Test Equipment and Material	44
5.2.5.Experiment Information	45
5.2.6 Questionnaire for Usability Evaluation (Ease of learn)	45
5.2.7 Retention Rate Evaluation.....	46
5.3 TECHNICAL REQUIREMENTS	46
5.4 PERFORMANCE TESTING.....	47
CHAPTER 6 RESULT AND DISCUSSIONS	48
6.1 USABILITY EXPERIENCE RESULT	48
6.1.1 Questioner result.....	48
6.2 TECHNICAL REQUIREMENT RESULT.....	60
6.3 PERFORMANCE CAPABILITIES RESULT.....	65
6.4 DISCUSSION	70
6.5 THREATS TO VALIDITY	73
6.5.1 External validity.....	73
6.5.2 Internal validity.....	74
6.5.3 conclusion validity	74
6.5.4 construct validity.....	75
CHAPTER 7 CONCLUSION	76
CHAPTER 8 FUTURE WORK	78
REFERENCES.....	79
APPENDIX	87
TASK	87
QUESTIONNAIRE	89
PERFECTO AND XAMARIN DEVICES	90
CODE EXAMPLE FOR PERFECTO AND XAMARIN FRAMEWORK	92
ADDITIONAL DETAILS	94

List of Figure

Figure 1 Android platform architecture (Lettner et al., 2011)	8
Figure 2 Android Fragmentation	10
Figure 3 Automation Testing Method.....	13
Figure 4 Paper selection Process.....	17
Figure 5 AM-TaaS Architecture (Rojas et al.,2016).....	24
Figure 6 Cloud based infrastructure of MTaaS.....	25
Figure 7 Research Methodology.....	30
Figure 8 Droid Weight health application	33
Figure 9 Likert Rating Scale	46
Figure 10 Excel Analysis Snapshot	49
Figure 11 t-test two sample snapshot.....	49
Figure 12 Example for t-test	50
Figure 13 Perfecto Ease of use.....	53
Figure 14 Xamarin Ease of use	54
Figure 15 Sufficiencyof learning material Result	56
Figure 16 Programming skills material.....	57
Figure 17 Retention Rate for Perfecto and Xamarin	58
Figure 18 Fault Detection Graph	60
Figure 19 Perfecto Vs Xamarin test result report	62
Figure 20 (A) Perfecto pass result report. (B) Perfecto failed result report.....	63
Figure 21 All perfect test report.....	63
Figure 22 All test in Xamarin Framework	64
Figure 23 Xamarin test report.....	64
Figure 24 Samsung Galaxy S9.....	66
Figure 25 Perfecto Cloud Testing Time Execution	67
Figure 26 Perfecto Cloud Testing Time Execution	67
Figure 27 Xamarin cloud testing Time Execution.....	68
Figure 28 Perfecto Execution time	69
Figure 29 Xamarin Execution time.....	69
Figure 30 (A) Xamarin Rating, (B) Perfecto Rating.....	72
Figure 31 Perfecto and Xamarin Review	73
Figure 32 Perfecto Devices.....	90
Figure 33 Xamarin Devices	91

Figure 34 Perfecto Code Example	92
Figure 35 Xamarin Code Example	93
Figure 36 My test Perfecto Framework	93
Figure 37 Device Capabilities (Perfecto Framework)	94
Figure 38 My Device (Perfecto Framework).....	94

List of Table

Table 1 Included studies of related work.....	18
Table 2 Evaluation criteria for mobile cloud test frameworks	34
Table 3 Data Collection for evaluation criteria.....	35
Table 4 Usability Experience Category	36
Table 5 Technical Requirement Category	37
Table 6 Performance Capabilities Category	37
Table 7 Group Design.....	40
Table 8 Group Description.....	43
Table 9 Set of question	45
Table 10 Mean and SD for question two	50
Table 11 Mean and SD for question three	51
Table 12 Mean and SD for question six.....	51
Table 13 Mean and SD for question eight	52
Table 14 Mean and SD for question nine	52
Table 15 Mean and SD for question one	54
Table 16 Mean and SD for question seven.....	55
Table 17 Mean and SD for question five	56
Table 18 Mean and SD for question four.....	59
Table 19 Mean and SD for question eight	61
Table 20 Scaling Rate	89
Table 21 Questionnaire.....	89

Abbreviations

<i>SaaS</i>	<i>Software as a Service</i>
<i>IaaS</i>	<i>Infrastructure as a Service</i>
<i>PaaS</i>	<i>Platform as a Service</i>
<i>TaaS</i>	<i>Testing as a Service</i>
<i>OS</i>	<i>Operating System</i>
<i>API</i>	<i>Application Programming Interface</i>
<i>ADT</i>	<i>Application Development Trends</i>
<i>DDT</i>	<i>Data-Driven Testing</i>
<i>APK</i>	<i>Android Package Kit</i>
<i>SDK</i>	<i>Software Development Kit</i>
<i>MIaaS</i>	<i>Mobile Infrastructure-as- a service</i>
<i>CTP</i>	<i>Cloud Testing Platform</i>
<i>AM-TaaS</i>	<i>Automated Mobile Testing as a service</i>
<i>MTaaS</i>	<i>Mobile Testing as a Service</i>
<i>CSTS</i>	<i>Cloud Software Test data generation Service</i>

Acknowledgment

First and foremost, I want to thank ALLAH for his guidance, without which I would not have been able to accomplish what I have.

I'd like to express my gratitude to Dr. Samer Zein, my thesis advisor at Birzeit University, for his tireless efforts. He also always pointed me in the right direction when he thought I needed it.

I'd also like to thank Dr. Mamoun Nawahdah and Dr. Radi Jarar at Birzeit University for serving as second readers and dedicated members of this thesis committee.

Finally, I'd like to express my heartfelt gratitude to my parents for their unwavering support throughout my years of study. This accomplishment would not have been possible without their assistance. Thank you very much, Mom and Dad.

Chapter 1 Introduction

In this chapter, we will describe a general overview of mobile applications based on cloud testing. A brief introduction on cloud testing and the motivation for doing this research will be discussed at first. The research problem will then be summarized, thus creating a research gap. After that, the research questions will be identified and then the research outlines will be completed.

1.1 Introduction

Nowadays, smartphones are prevalent and play an important role in different modern life aspects. Mobile applications are evolving rapidly and many people are using them to manage their daily activities. In addition, mobile apps are integrated with different sectors such as m-banking and health, payments just to mention a few. Moreover, the different platforms of iPhone and Android make the developer face problems to provide an application that is suitable for all platforms. With high competition in the mobile apps marketplace, immense pressure is laid on mobile app developers to deliver high quality mobile user experience in a short time. Also, if a small error occurs, it may lead to poor review in users, so we need mobile application testing (Malini et al., 2014).

The aim of mobile application testing includes verification, validation, quality assurance, and reliability estimation. It is a major activity of any software quality assurance (Girardon et al., 2020). Mobile app testing follows two approaches: manual and automated. In real time projects, manual testing might not always be effective since it consumes a considerable amount of time, effort, and cost. On the other hand, automated testing can be very efficient when compared with manual testing. Automated testing is usually implemented by recording the interaction with the system into test cases to be tested under many configurations (Zein et al., 2015).

In mobile app development, developers and test engineers face the problem of large fragmentation in mobile platforms, hardware, and operating systems versions. Consequently, testing mobile apps on various devices is complex and needs more resources. In other words, when a mobile app runs on various mobile devices, it is needed to connect these mobile devices with that app to examine if it is working as well or not. This is where testing in the cloud comes in handy (Malini et al., 2014).

Cloud-based software services rely on the basic cloud infrastructure, which includes the operating system (OS), networking, storage capacity, and networking. Software services are expected to scale up and down based on user demand using the virtual infrastructure provided by cloud services. As an example, software services as a service (SaaS) are on-demand apps that follow a pay-as-you-go model. It is also supported by the provision of infrastructure as a service (IaaS) and platform as a service (PaaS). Finally, the number of cloud-based apps is rapidly increasing as a result of increased availability as well as the efficiency of cloud computing (Armbrust et al., 2010).

Cloud testing is a challenging activity for a software engineering project that is rapidly developing in software testing research (Hosseini et al., 2015). Testing mobile applications on the cloud offer the best advantages compared to traditional methods because resource required for mobile testing is available in the cloud such as hardware, software, and infrastructure; so many companies are moving to cloud service (Kaur et al., 2016) Liviu Ciordea et al. introduce cloud9 which is the first parallel symbolic execution engine. It is also a cloud-based testing service that makes testing quick, inexpensive, and high-quality. They also created an initial cloud-based test experiment (Ciordea et al., 2011)

The most notable fact is that cloud testing makes use of cloud infrastructure for testing, but users are not required to be aware of the technology or infrastructure that is being used. When cloud testing is used, the time it takes to set up the environment is reduced, and services can be accessed from anywhere while utilizing existing resources (Temkar et al.,2015). The goals of cloud testing are to provide high-quality testing while avoiding data loss both outside and inside the data center (Mittal et al.,2017). Platform testing can be done in a variety of cloud models (private, public, hybrid, and community) (Riungu et al., 2010).

Cloud testing frameworks are being used to facilitate mobile testing. The majority of cloud testing frameworks for mobile testing services are referred to as a "cloud of devices." In other words, it offers remote access to real-world devices in the cloud as well as device hosting. This service assists developers and test engineer in using remote smartphones (real devices) for automation testing on various platforms (IOS, Android), as well as manual testing and script recording (Fazzini et al.,20120). The benefits of cloud testing for mobile applications can be summarized as follows (Gao et al.,2011):

- The mobile app can be tested on various platforms (Android, IOS).
- Allow for a quick time to market due to the virtual test environment's quick availability.
- The test environment can be scaled up and down as needed.
- Testing can be done at an independent location, allowing us to test anywhere.
- A large number of smartphones (real devices) are ready for testing.
- Cloud tests can be run on multiple devices at the same time.

Nowadays, many frameworks are available on the cloud to test mobile applications by using real devices. Despite the fact that there are numerous studies that focus on presenting new approaches for these frameworks, to the best of our knowledge, there are few studies that investigate comparing these frameworks. Cloud testing service frameworks such as SauceLabs, SOASTA, Firebase7, Amazon Web Services AWS Device Farm6, TestDroid, Perfecto Mobile, and Xamarin have been proposed as alternatives frameworks to provide testing environments in recent studies. When compared to other frameworks, this can be more cost effective for testing and more effective at supporting a wider range of devices to be tested. Also, cloud testing is becoming a great topic of research that is being used in different software (Rojas et al.,2016).

According to the findings of cloud testing research, there is a lack of research that investigates comparing cloud testing frameworks based on the mobile application. In this research , two frameworks are researched and compared using various criteria to determine which one is the best.

1.2 Motivation

Cloud testing has recently become the most popular method of software development, with widespread success when compared to traditional methods. However, quality assurance and testing engineers face some challenges in cloud testing applications, such as building on-demand test environments and testing security and measurement in clouds. Furthermore, while many papers have been published discussing cloud architecture, models, and design, cloud testing is still a relatively new topic in software testing (Gao et al.,2011).

To test mobile apps, various frameworks are available on the cloud. Although many studies focus on listing those frameworks, few studies focus on comparing them. During our literature search, we discovered only one comparison study in the field of cloud testing frameworks based on mobile. Nachiyappan et al. (2016) present a comparative study that focuses solely on the benefits and drawbacks of some cloud testing frameworks. Our research , on the other hand, is comprehensive because it compares the two most popular mobile cloud testing frameworks based on identified comparison criteria. Also, it helps practitioners to choose the best frameworks in the cloud to test mobile apps.

Finally, I'm writing this research to learn more about cloud testing. In addition, to gain experience with testing on real-world devices. As a result, this motivates me to write a research on cloud testing and, possibly, work as a Quality Assurance Engineer in the future.

1.3 Research Problem

Selecting the cloud testing frameworks to test mobile apps is a critical decision that teams should take. Several studies have been performed in the cloud testing frameworks. However, these studies focused on explaining cloud testing frameworks. Based on the literature review there is no study

focused on comparing cloud testing frameworks except one study that focused on analyzing advantages and disadvantages for some cloud testing frameworks. Consequently, this research aims to make a comparison between the widest cloud testing frameworks based on the mobile application. In our research , we examined the details about selected frameworks and we chose an Android app. Because of Android's widespread in the market.

1.4 Research Questions

- **RQ1:** How cloud testing frameworks for mobile apps are compared in terms of usability experience (ease of learning and retention rate) ?
- **RQ2:** How cloud testing frameworks for mobile apps are compared in terms of technical requirements (test result report)?
- **RQ3:** How cloud testing frameworks for mobile apps are compared in terms of performance capabilities (execution speed of scripts)?
- **RQ4:** How cloud testing frameworks for mobile apps are compared in terms of fault detection?

1.5 Report Structure

The rest of this research was structured as follows:

- Chapter 1 describes an introduction to cloud testing, the motivation for the research , research problems, and research questions.
- Chapter 2 describes backgrounds and introduces all related concepts to provide reader with a suitable background.
- In chapter 3 presents a literature review and document prior works as set of groups.
- chapter 4 shows the research methodology in order to show the data collection and analyze it. Also, It provides a general idea about research experiments.

- Chapter 5 presents the evaluation of the experiment to explain how it is comprehensive and satisfies the research objective.
- Chapter 6 analyzes and discusses the results of the experiments. Furthermore, hypotheses were tested and threats to validity mitigated.
- Chapter 7 and 8 summarize the findings and make recommendations for future research.

Chapter 2 Background

This chapter provides an overview of cloud testing. It was set up as follows: Section 1.2 provides a high-level overview of the Android platform. The high fragmentation of Android devices is depicted in Section 2.2. Section 2.3 of the final section discusses cloud-based automated testing.

2.1 Android Platform

The number of mobile applications developed on the Android operating systems is growing due to the diversity of mobile platforms. Android is a mobile operating system that offers a set of features to support mobile applications. It is also an open-source mobile software environment that was developed and designed for mobile devices in 2007 by the Open Handset Alliance and is based on the Linux kernel (Grgurina et al., 2011). Java language codes are used to create Android mobile applications, and it allows developers to write their code in Java. It is a flexible OS environment to develop mobile apps as the developers aren't only using Android Java libraries but can use normal Java IDE (Holla et al., 2012).

The Android platform has two major advantages over other development platforms: it is free to download and use because it is open-source, and it is built in Java, the most popular programming language. As a result, it is a popular choice among smartphone manufacturers and has been used in a wide range of smartphone devices. Furthermore, because it is the primary target choice for scientific research, it has contributed to identifying the development and design issues of tools to assist developers (Ghanem et al., 2020).

As mentioned above, the Android applications are written in the Java language and then compiled into Android packages that are called (APK) Android Package Kit file by using (SDK) Android Software Development Kit tool that developed over Google. Also, SDK provides the testing

environment to allow developers to test applications. Moreover, when users install an app, the device will install the APK file (Moonsamy et al., 2014).

In Android, the Linux kernel interacts with the device hardware, while application APIs run after the kernel. Figure 1 depicts the Android platform architecture, which is divided into five layers: native libraries, Linux kernel, framework layer, Android Runtime, and top application. The Linux kernel includes models for security, networks, and memory management. Furthermore, some libraries, such as Webkit, SSL, and SQLite, provide system functionality (Lettner et al., 2011).

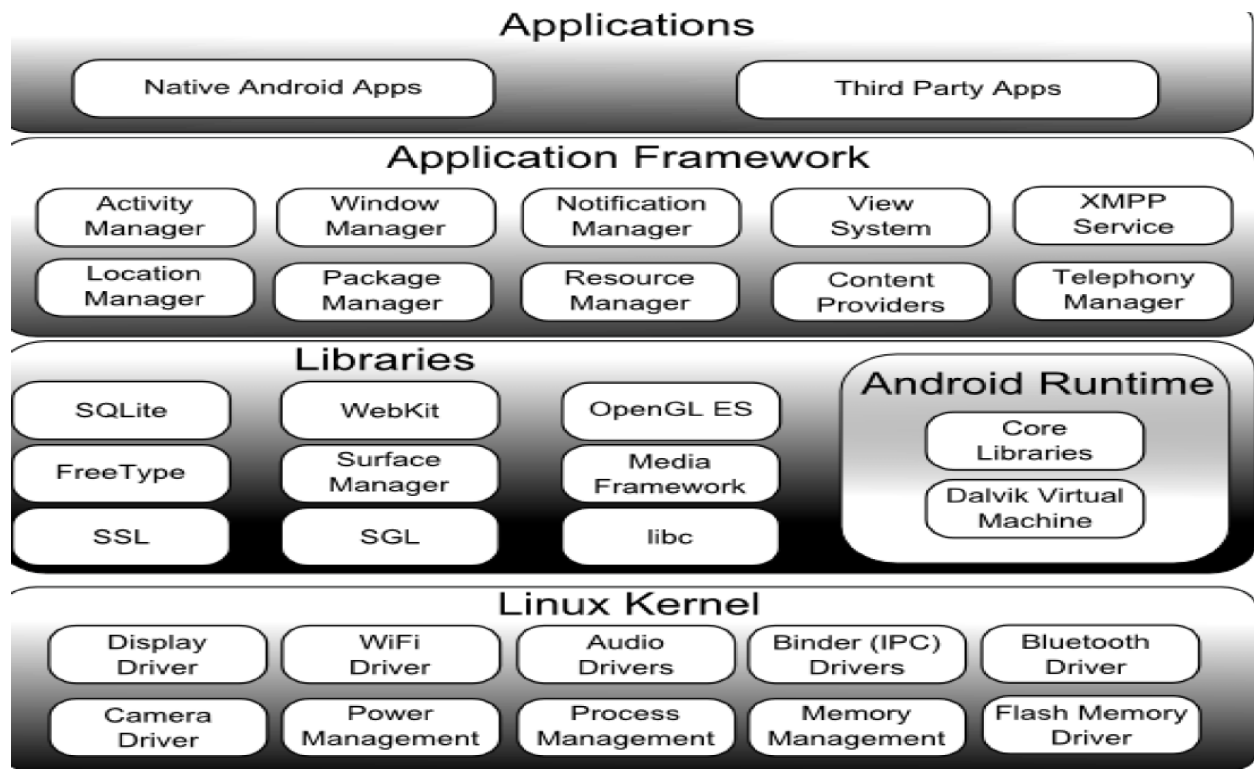


Figure 1 Android platform architecture (Lettner et al., 2011)

Each Android application contains a group of the components announced in a manifest file and always examines whether those components are on a manifest before running the application. The following components are (Yerima et al., 2013):

- Activities: Activities represents a graphical UI of an app such as a screen that the user interface in order to perform actions (take a photo, send emails)
- Services: Service is running in the background to perform the remote process.
- Broadcast receivers: It deals with application announcements such as screen off or low battery.
- Content providers: It enables modifying data that is stored in the file system, database, or other storage location.
- App permissions.

In our research , we selected the Android platform as our aim for various reasons. It has the largest market of mobile devices of 87% in 2019 (Motan et al., 2020), and also it is an open-source platform that helps researchers free to explore it. Finally, Android application testing is still an active area of research.

2.1 High Fragmentation of Android Devices

Over the past few years, a multitude of mobile devices and their apps have presented into markets. So, the increasing requirements have compelled the developers to rethink testing practices to build high quality apps at low costs. But, testing mobile apps isn't straightforward testing due to the multitude of devices and their fragmentation (Xavier et al., 2017).

Android is an open-source platform for mobile devices that contains an OS, key applications, and middleware. From that time up to now, it has enhanced either in terms of supported hardware or features and at the same time expanded to new types of devices that are different from original ones (Gandhewar and Sheikh, 2010). Recently, the Android platform achieves the largest market share when compared to other platforms because it is an open-source environment for developers and vendors.

They designed the Android to be compatible with a wide range of hardware to allow manufacturers free to process, design, or integrate ideal components of the Android device. The result has led to the development of powerful Android devices. Although compatibility is greeted in manufactures, consumers, and wireless providers, diversity leads to challenges for engineers (Hoog, 2011).

Android has a diversity of device configurations, each with different OS versions, screen sizes, and other software or hardware differences. The device manufacturer has the freedom to create a mobile device in any way so that it leads to diversity in hardware and software. Hardware diversities refer to differences in processing resource processes that operate in operating system versions such as tablets and smartphones that look physically different but run in Android (Halpern et al., 2015)

The Android device suffers from fragmentation as a critical problem among developers. The fragmentation can be device fragmentation or OS fragmentation, the classification of Android fragmentation can be shown in Figure 2. According to Mr.Dan Morrill, people understand Android fragmentation issues in different ways. Also, hence more than versions from Android operating systems which are inconsistent among different platforms which make this problem serious (Kamran et al., 2016).

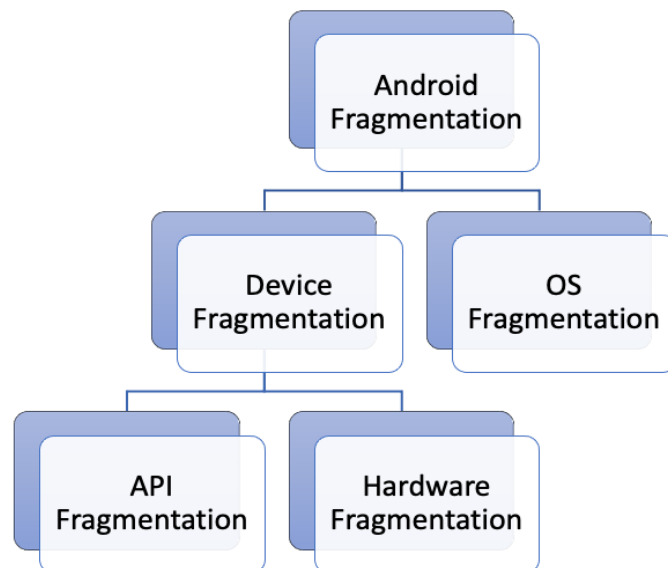


Figure 2 Android Fragmentation

OS fragmentation is caused by its various versions of a specific OS moreover, Android continuously updating or enhancing the OS makes the problem much harder. But according to new studies, operating system fragmentation can be reduced by reducing the oldest versions of operating systems (Park et al., 2013).

A major issue in fragmentation is device fragmentation which can be hardware or API fragmentation. Hardware fragmentation refers to a variety of Android based devices that have various hardware features such as screen size. However, between device-related challenges, a display resolution can be considered as the main problem. API fragmentation occurs because of the modification of API according to different strategies of various service providers (Park et al., 2013).

One unique challenge to develop Android applications is how to deal with the high fragmentation of Android devices. (Halpern et al.) proposed a capture and replay approach for testing and made a careful analysis of fragmentation caused by operating system version and hardware. To handle Android fragmentation (Park et al.) proposed two approaches at the code level and API level. But some existing cloud testing frameworks such as Perfecto, TestIn, Xamarin Test Cloud, and AppthWack offer sufficient coverage of real device models which solve that problem (Lu et al., 2016).

To solve some fragmentation issues, perform as much testing with real devices as possible. However, due to limited resources and time, testing on every real device is impossible. As a result, using the cloud testing framework is the best solution (Lanui et al., 2019).

2.3 Automated testing in the cloud

Cloud testing can be defined as a software testing approach that is done using cloud computing which offers different benefits compared to traditional strategies. Cloud testing helps to test cloud applications or using the cloud to test applications such as mobile apps or web apps. According to

Nivedan Prakash (Gao et al.,2011), testing in the cloud seeks to simulate user traffic in the real world as stress or load testing websites. Also, Wikipedia [5] defines cloud testing as one of the forms of software testing that web applications search to simulate user traffic in the real world to stress testing and load testing websites.

Cloud-based automation testing can implement some testing activities that are complex in traditional testing, such as providing testing resources that are accessible from anywhere and also maintaining software so that testers do not have to spend a lot of money. As a result, automation cloud testing offers numerous benefits when testing with various software and can expedite the testing process because users do not need to set up the environment (Lanui et al., 2019).

Furthermore, the cloud-based automation testing methods can be summarized as shown in Figure 3:

- Functional testing: it is the process of examining all features and functions of a system, such as a system testing, integration testing, and user acceptance testing, in order to ensure and maintain the quality of the product (Nurul et al.,2019).
- Non-functional testing: it is done to ensure that the application satisfies the required requirements such as security and performance testing. Therefore, it is reflected in the quality of the product and has a great influence on the customer (Shrivastva et al.,2014).
- Ability testing: it is done to make sure that the user receives suitable service from a cloud environment when demand such as compatibility testing, disaster recovery testing, interoperability testing, and multi-tenancy testing (Nurul et al.,2019).

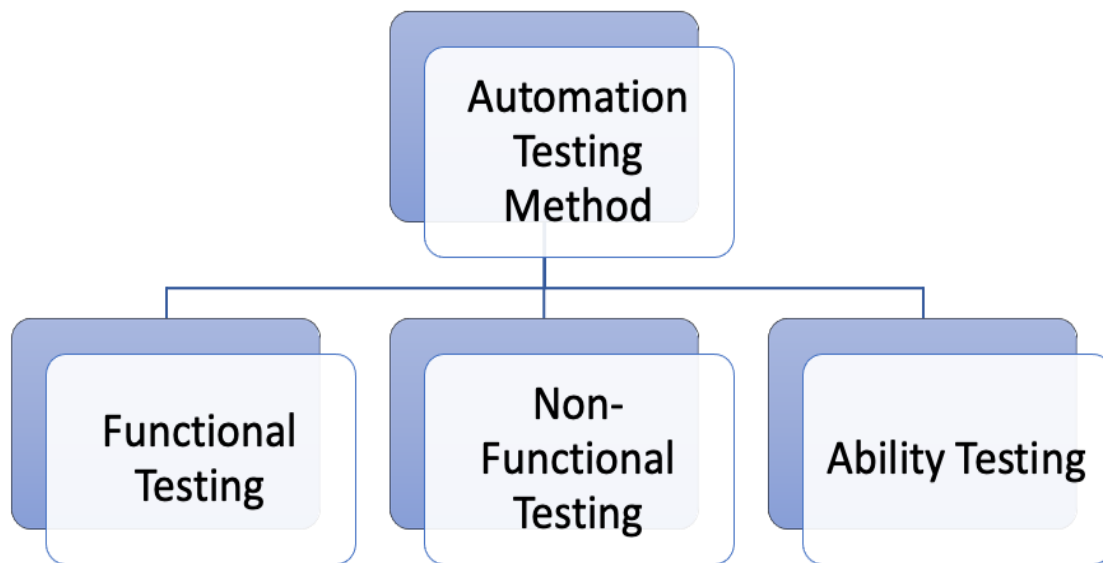


Figure 3 Automation Testing Method

Cloud testing of mobile applications means that the resources such as hardware, software, and infrastructure are required for testing are available on the cloud. There are many companies that are moving to cloud testing (TaaS) to test mobile apps (Kaur & Kaur,2016). TaaS is considered a popular scalable testing environment in cloud computing. Also, it uses cloud infrastructure based on the (pay-as-you-test) model to enhance resource sharing (Ya'u et al.,2019).

The TaaS environment is the most recommended environment by Inki et al. to perform and migrate testing on it because of the possibility of using the resources to execute tests. Besides, TaaS combines two ideas of doing fully automated testing in the cloud and providing software testing as an easily accessible and available web service. According to Rojas et al. TaaS has benefits and is important for several reasons such as scalable test, on-demand test, cost reduction, and quality certification by the third parties (Rojas et al.,2016).

Chapter 3 Literature Review

3.1 Introduction

In this chapter, we present a review of related work that will be used in this comparative study. We provide an overview of existing related research in our research fields and reduced it because exclusion/inclusion criteria must be defined (Petersen et al., 2008). The analysis of the included studies identifies the research gap in cloud testing, allowing future research options to be identified

The majority of the research proposes their solution using the Android platforms. That might be due to the high flexibility of the Android platform and is the most popular platform. (Villanes et al.,2015). Moreover, some studies propose approaches and solutions that don't focus on one platform, in other words, the solution or approaches can be used in any mobile platform. While some research does not focus on their solution to any platform which means it is an open solution.

To the best of our knowledge, there is only one comparative systematic study between cloud-based testing frameworks that motivated a comparative study and identified gaps in the field of cloud-based testing research. Several studies were generated as a result of the initial search results. Following the application of the exclusion/inclusion criteria, a total of 24 studies were included in the comparative study. We proposed three groups of research: (1) mobile app testing methods; (2) cloud testing evaluation methods; and (3) cloud testing contribution facts.

This chapter, in particular, described cloud testing based on mobile applications, with a focus on a variety of fields such as functional testing, non-functional testing, and frameworks. It is also organized as follows. The analysis of search methods is presented in section 3.2. Section 3.3, on the other hand, divides the related work in cloud testing into three categories: mobile app testing methods, cloud evaluation methods, and cloud contribution facts. The final section 3.4 provides a summary of the main research gaps.

3.2 Search Method

In this section, we used a critical review to examine previous research, focus on steps to prepare and gather discussions of related work. In addition, we used criteria to include or exclude studies and then categorize acceptance studies into groups.

3.2.1 Method

The methodology search for this research is inspired by Peterson and Kitchenham's studies (Mujtaba t al., 2018). According to Petersen, at first, we define the objectives of the research, search conduct, screen paper, abstract reading, and then critical review. After critical review, we applied include and exclude criteria (Mujtaba et al., 2018). Finally, we're pooling the related papers together.

3.2.2 Source Databases

The present study targeted the three online databases that commonly used in publishing comparative studies which are:

- ACM Digital Library.
- IEEEExplore.
- Google Scholar.

3.2.3 Search Strings:

In this study, we included studies directly related to cloud testing, the cloud testing framework, and cloud testing for mobile applications. The search strategy adopted in this research to construct the search string is suggested by the study of (Kitchenham et al., 2007):

- We are using logical operators like OR, and and. We use and operator to restrict our search to essential terms. We're also using an OR operator to expand our search.
- We are looking for synonymous spelling.

In our research , several attempts were made to analyze the perfect search string. On the other hand, our challenge was in fact that cloud testing is a new research topic. The keywords included in the search string were essentially inspired by our suggested research questions. The “ cloud testing framework for mobile applications” was the main search string. We also used synonyms for applications watch are apps and in some times each one refers to different studies.

Finally, the search approach referred to in (Feldrer et al., 2018) is as followed :

- Forwarded snowballing: get the studies that are cited in the selected studies.
- Backward snowballing: get the studies from the reference list of each selected study through careful monitoring.

3.2.4 Inclusion, Exclusion Criteria and selection process

To select appropriate studies, we need a clear definition of the exclusion criteria, the inclusion criteria, and the selection process. Criteria for inclusion are selected from all research papers that provide evidence of cloud testing. However, studies that fall into any of the following have been excluded:

1. Papers have not been written in English.

2. Papers that were published before 2015.
3. Studies that do not focus on cloud testing.
4. Short papers that are less than five pages long.

The phase selection of research studies was iterative and incremental. For filter studies that have been included in our research, each study will be carried out through three different filtration steps as shown in Figure 4. The first step was to apply a search string to the relevant studies. Subsequently, in the second step, we filtered papers based on their title and abstract and then excluded papers that were not related to cloud testing. In the last step, we applied selection criteria by reading the introduction, methodology, and conclusion, and then excluding papers that were not relevant to our research.

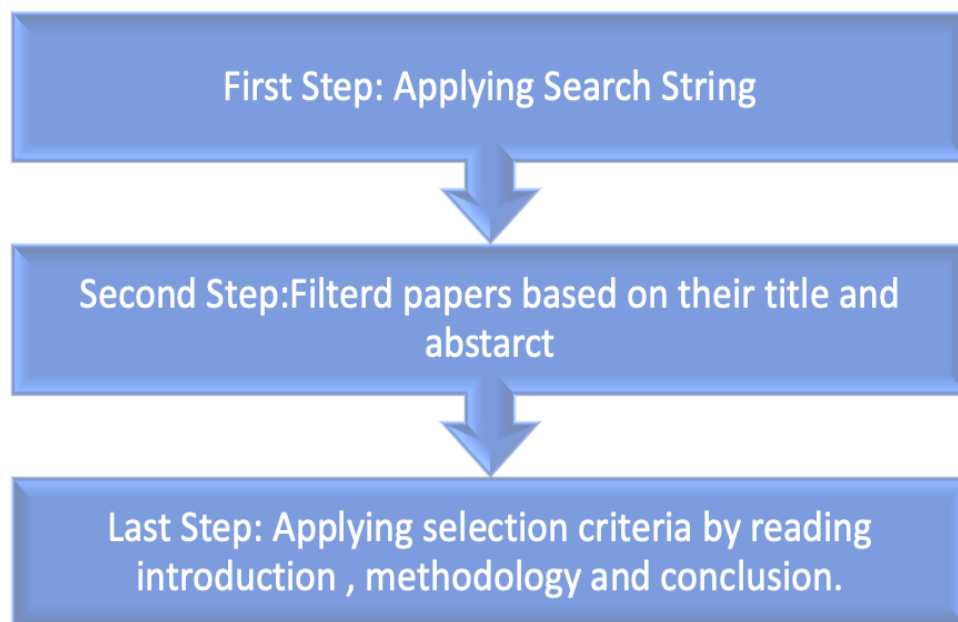


Figure 4 Paper selection process

After applying the exclusion, including criteria, 24 studies were selected and then categorized into three groups, as shown in Table 1.

Table 1 Included studies of related work.

Name of group	Related Paper
<i>Mobile apps testing method</i>	Tao et al.,2015 Zhang et al.,2015 Ali et al.,2018 Liu et al.,2017 Ahmad et al.,2018 Lanui et al.,2019 Nurul et al.,2019
<i>Evaluation Methods testing in the cloud</i>	Ya’u et al.,2019 Bertolino et al.,2019 Riungu-Kalliosaari et al.,2019 Ahmad et al.,2017 Qusef et al.,2019 Tao et al.,2016 Temkar et al.,2015
<i>Contribution Facts for testing in the cloud</i>	Gao et al.,2015 Rojas et al.,2016 Liu et al.,2015 Al-Ghuwairi et al.,2016 Bertolino et al., 2018 Tao et al., 2017 Chen et al., 2020 Chawla et al., 2019 Lo et al., 2015 Villanes et al.,2015

3.3 Related Work

3.3.1 Mobile apps testing method

Various approaches and techniques can be used to test mobile apps. In this group, the types of cloud-based mobile application testing are identified as part of primary studies. Seven studies have been described in this section, and articles in this group have proposed mobile app testing methods in the cloud. Some parts of the research introduced functional testing (Zhang et al.,2015) and the others introduced non-functional testing, security (Nurul et al.,2019), scalability (Ahmad et al.,2018), compatibility, and portability (Lanui et al.,2019).

The superB methodology is a new methodology that covers functional and non-functional aspects of testing techniques (Nurul et al.,2019). This new methodology deals with user acceptance, security, business requirements, and performance of cloud testing. Also, it is presenting in-depth the testing requirement in cloud testing, such as functional testing, non-functional testing, and ability testing techniques. To achieve security testing, they adopted Metasploit, which is a testing tool that offers security scanning in the test environment (Tao et al.,2015). In addition, they resolved the challenges that they faced when using traditional testing methods by cloud-based automated security testing.

The cloud testing platform (CTP) is a platform that is used to reduce the cost and time needed for Android multimedia app compatibility testing (Liu et al.,2017). Also, it allows users to test the Android multimedia application, which can be tested automatically by choosing a different scalable number of devices. Therefore, it supports various types of testing tools to assure compatibility with Android multimedia applications such as GUI testing (Ali et al.,2018), Crash testing, stress testing, etc. Furthermore, the system can provide screenshots, video, and performance data that help to make easy identification of defects in the application. Another study evaluated the cloud testing model for Android applications to solve the Android fragmentation

issues and helps Android developers and testers to meet quality characteristics especially compatibility and portability characteristics (Lanui et al.,2019).

To improve resource utilization and reduce the time (Ali et al.,2018), they used TaaS-based GUI testing for mobile applications described in CTP (Liu et al.,2017). This system guarantees the automatic generation of the test cases with higher coverage compared with other techniques. Also, cloud resource load balancing is achieved through resource allocation and scheduler modules. In addition, the functional test solution was used for Android native applications based on the TaaS platform (Zhang et al.,2015).In cloud testing, the test script was generated automatically based on a functional crossover. Also, the user is allowed to customize the test environment by configuring device context parameters such as language, location, and so on.

Two cloud-based software scalability metrics were performed through experimental analysis and run on the Amazon AWS cloud platform (Ahmad et al.,2018). One metric addresses the quality of the service's scalability and the other volume of the service's scalability. In addition, these metrics can be used on their own in the utility-oriented metrics of cloud-based service scalability. They have shown that technical scalability metrics can be used to design and perform scalability testing of cloud-based software systems.

Some studies evaluated their solution through experiments, or case studies, or interview testers. The first evaluation was carried out through an experiment by using two test cases and 100 mobile applications with different vulnerabilities (Tao et al.,2015). In the second study (Ali et al.,2018), they presented an experiment that showed the proposed system high performance of the test due to the simultaneous execution of the test cases on several virtual nodes. Other evaluations, the results of a case study were evaluated for four different applications and showed that CTP can be effective, including reducing time and cost. They conclude that the CTP only supports multimedia apps that can be tested by using the supporting tools of CTP (Liu et al.,2017). A final evaluation was conducted through an interview with 10 testers to compare the proposed system with other systems such as AppACTS. This evaluation study found that cloud testing has the possibility of managing challenging compatibility and probability on the Android platform (Lanui et al.,2019).

Finally, some gaps and limitations have been identified, which will be addressed in future works, and the authors themselves mentioned these gaps and limitations. The solution only focused on the Android mobile application, so that the study was extended to other OSs to validate security for mobile applications and required more reliable testing tools (Tao et al.,2015) (Zhang et al.,2015) (Ali et al.,2018). However, superB is presented to explore all phases of the cloud testing life cycle. The solution was not proposed only on the Android platform same as (Zhang et al.,2015), but it proposed it in general, which means that it can be applied to different platforms. (Nurul et al.,2019) Also, the CTP solution only focused on the Android platforms, so they need to experiment with other platforms to validate it (Liu et al.,2017). The scalability metric solution has only used (Amazon AWS) cloud platform and only one cloud based software service to demonstrate the scalability metrics (Ahmad et al.,2018). When solved the fragmentation problem by the Android platform, it wasn't compatible with the Android version 3.0 and below and doesn't provide a screen capture feature, there was no application screenshot provided in the test report would cause more effort and time for the testers to verify and analyze the result (Lanui et al.,2019).

3.3.2 Evaluation Methods testing in the cloud

Previous studies examined cloud testing indifference of ways; some conducted empirical research on cloud testing in practice, while others examined cloud testing opportunities and challenges. This section will go over seven previous studies. The majority of the studies described in this section used empiric or systematic studies to evaluate cloud testing, and some of them described the challenges of cloud testing and compared it to traditional testing. Finally, all researchers discovered that cloud testing research is still in its early stages and requires additional attention from researchers.

Cloud-based mobile Tass offers great advantages such as offering large scale in mobile testing anywhere and any time providing elastic scalable test mobile automation; reducing costs by sharing mobile resources. The comparison between the conventional testing and cloud testing

infrastructure has shown that the cloud testing base mobile focuses on typical features such as scalability (Tao et al.,2016). The benefits of cloud testing that helps to reduce testing costs, decrease the testing time, and extensibility by using the service model test as a service (TaaS). TaaS infrastructure is a new service model in which the cloud service provider performs the testing activities for a given mobile app. TaaS provides the following service, such as real device; emulator; automated testing tool (Temkar et al.,2015). Cloud testing helps with flexibility, collaboration, and scalability in more appropriate testing approaches. In general, testing in the cloud is the most effective solution when studied and implemented well and helps companies to apply testing practices and take efficient solutions when using cloud-based systems (Qusef et al.,2019).

Through empirical research to develop a classification scheme, the research used 69 primary studies as found in 75 research publications and focused only on how to evaluate proposed cloud testing techniques and what is the application domain. The result showed that most of the studies present only preliminary results and often describe an example of cloud-based software testing methods or a simple application experiment to evaluate the proposed approach. However, some of the studies used a unique experiment to evaluate their proposed solution and needed more experiments to validate it (Ahmad et al.,2017).

When analyzing responses by qualitative reports through 35 interviews with a tester from 20 organizations, they found that there is a growing interest in opportunities offered by cloud testing tools. Hence, the result has shown that cloud computing has the potential to provide the organization with suitable resources supporting different needs. Also, it helps practitioners and researchers to understand how cloud resources can be used to perform positive outcomes. However, some test problems have also been identified that are still unsolved, such as cloud testing, which expands both the automated and manual test offerings but does not provide a generic test automation environment to meet testing needs. (Riungu-Kalliosaari et al.,2019). This study depended on quality report evaluation research through interviews and did not depend on previous studies as well (Ahmad et al.,2017).

The main challenges needed a potential link between the software testing research and progress in cloud computing to facilitate resource management, as well as research related to cloud testing is in continuous evolution, especially in the mobile application. However, performance indicators have been identified as the main objective for cloud testing such as latency, response time, and execution time (Bertolino et al.,2019). In a systematic mapping study on cloud based mobile applications testing, 23 studies were collected from different search phrases, and the area of research on cloud-based mobile application testing was still new. Most of the studies focused on Android applications and neglected other mobile application platforms such as IOS and Blackberry. They, therefore, found that there was a lack of approaches that offered a solution for mobile applications that is not indicated on any specific platform. In the end, they also examined TaaS and found that 47.8% of the studies suggested that TaaS was the key to success in mobile testing, and 52.2% used different approaches than TaaS, so research is needed on the applicability of TaaS mobile devices (Ya'u et al., 2019).

Finally, the authors described the major gaps and limitations that cloud testing is still new and is a hot topic for research and needs more experiments to validate their solution. In addition, the research shows that there is a lack of approaches that offer solutions for mobile application testing that are not indicated on any specific platform (Ya'u et al.,2019).

3.3.3 Contribution Facts for testing in the cloud

Ten studies will be described in this section. Some studies proposed new contribution facts to develop cloud testing, such as the Framework and the Model and presented details of these contribution facts. Most of the frameworks were proposed in 2015 and some of them were only included in Android platforms and ignored any other platforms. We found ten studies describing frameworks that are well-structured and detailed, as well as two models that provide an abstract view of the subject and the problem.

The framework Automated Mobile Testing as a service (AM-TaaS) Figure 5 supports automated testing of mobile apps. It is primarily based on cloud technology and emulates mobile devices

using cloud infrastructure and virtual machines. AM-TaaS has been introduced to offer testing services to mobile devices because of a vast range of mobile devices with different platforms and their popularity. Also, this framework allows users to upload applications and then execute test scripts or can run default test cases that are implemented on the basis of the App Quality Alliance (AQuA) set criteria (Villanes et al.,2015). In another study, AM-TaaS was enhanced and the quality of mobile applications using cloud resources as a testing service improved. This framework (AM-TaaS) used an emulated device to reduce the complexity and costs of mobile testing. In other words, it focuses on the use of the emulated device rather than the acquisition of a real device due to the high cost of performing tests on several devices. It also allows compatibility and functional testing to be performed on several emulated devices (Rojas et al.,2016).

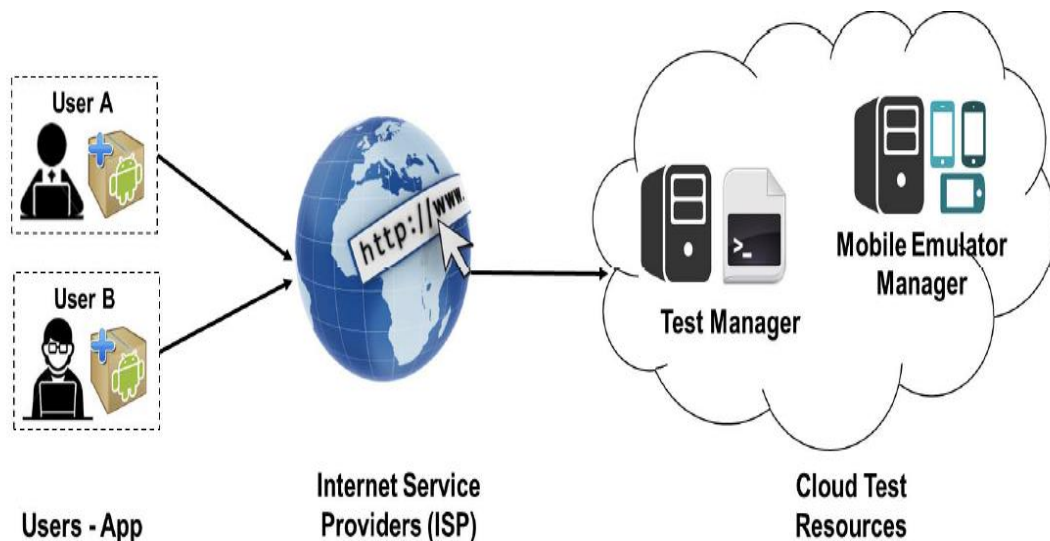


Figure 5 AM-TaaS Architecture (Rojas et al.,2016).

Mobile testing as a service (MTaaS) Figure 6 framework provides an infrastructure as a service (IaaS) for mobile testing. This framework supports mobile TaaS by providing instances such as mobile hubs, emulators, and server machines for testing applications. As a part of the analysis, the comparative study shows the advantage of the MTaaS framework and shows the effectiveness and feasibility. In addition, MTaaS is an effective framework for demanding large-scale mobile testing over the provision of services such as platforms and applications. Research, on the other hand, has limitations such as security issues and focuses only on the IaaS infrastructure (Tao et al., 2017).

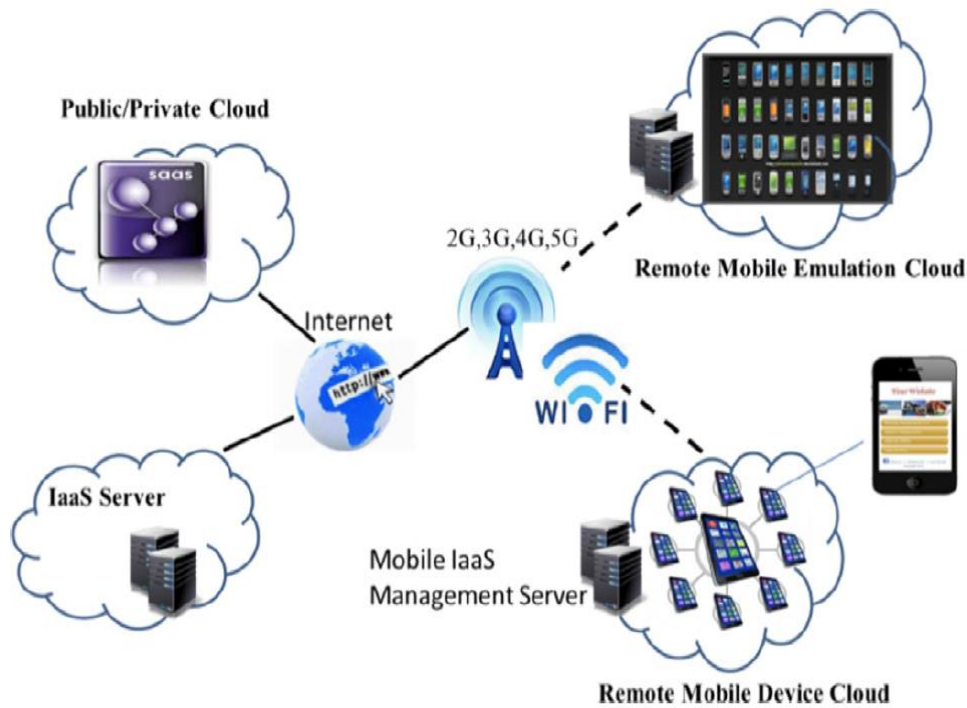


Figure 6 Cloud based infrastructure of MTaaS (Tao et al., 2017)

Mobile Infrastructure as a Service (MIaaS) framework for MTaaS to establish a mobile infrastructure. The infrastructure environment has supported the provision, billing, and monitoring of mobile resources across different clouds. They also presented a case study for load balancing algorithms and a comparison between them. However, the results of the study included only the use of the Android mobile platform and neglected any discussion to fit other platforms (Gao et al., 2015).

ElaSTset is the most comprehensive framework for testing and deployment of large systems. It has been validated through a quasi-experiment over real-world large-scale apps to propose a decision-making helper. The quasi-experiment showed that the ElaSTset is still incomplete and needed to add new features, and the essential result for them was to know what was improvements were needed by the partners involved in the quasi-experiments. In the end, most of the suggestions for improvement and feedback were relative to those of the partners involved in quasi-experiments (Bertolino et al., 2018).

A framework cloud testing platform (CTP) with application testing is designed to enhance and improve utilization of the resources on a cloud testing platform. This framework can dynamically adjust the number of the device and VMs based on the workload of CTP and also the number of available devices. It supports load balancing across multiple devices on the same type that helps to improve the use of the device. However, to make the result more reliable more experiments are needed to test the various mobile application devices (Liu et al.,2015).

Cloud-based software test data generation service (CSTS) framework aimed at facilitating the unit testing of systems under test. This framework focused on quality of service parameters such as security, fault detection, and cost. The framework provides the user with the most valid Hadoop and cluster configuration for improved cost and time performance. The framework has been evaluated by the experiment and has shown that the framework is effective and is a comprehensive, cost-effective, and efficient automatic software test (Chawla et al., 2019). However, in order to increase the flexibility of test customization and test scenarios, we used the IaaS-based cloud service testing framework. They presented solutions for quality and testing for both fault diagnosis and bottleneck detection using offline testing and online management techniques. The results show that the proposed system can effectively specify the bottleneck or fault of the target system (Lo et al., 2015).

A multilayer cloud testing model can implement a hybrid TaaS system based on a virtual network and virtual machine. In order to support the creation of a test environment, the system integrates the SaaS and IaaS, and also it provides a management portal for easy management of test services and test resources. To evaluate this model by experiments in terms of VM performance, VM provision time, and virtual network performance and compare the experimental results with those from Alibaba Cloud and QingCloud platforms. The results showed that the VM and virtual network can satisfy testers' basic demands and can provide low cost and accelerate the testing process (Chen et al., 2020).

(Tao et al., 2017) presented different cloud testing frameworks for frameworks besides MTaaS. For example, Perfecto provides mobile cloud monitoring and also mobile automation testing in

different platforms such as IOS and Androids. TestDroid offers mobile testing service on-demand with thousands of real devices of IOS and Androids. Another well Known Testin is a cloud-based application auto testing service provides. Cloud Testing frameworks provide service testing in industries. Some testing frameworks such as Testin, Xamarin Test Cloud WeTest from Tecent, MQC from Alibaba, and MTC from Baidu, focus on mobile testing. Also, it provides various mobile devices for testing the performance, function, compatibility, and security of mobile apps (Chen et al., 2020).

To evaluate the ability of TaaS a developing model approaches a mutation-based model are designed. The mutation-based model is designed to measure the quality, efficacy of TaaS and is helping to evaluate the effectiveness of TaaS based on the mutation score. the mutation score was used to implement it at the output of the TaaS process and the output of the mutation score was converted to criteria that could rank the ability of the cloud provider to perform TaaS To measure the effectiveness of TaaS, it required the insertion of cloud brokers between the customer and the cloud provider. This study is different from the previous one because it evaluates the efficacy of TaaS based on mutation score (Al-Ghuwairi et al.,2016).

In the end, AMTaaS found through experiments, 100% of the emulated devices could be tested using test cases of their framework. However, the approach was only limited to Android applications (Villanes et al.,2015). In addition, there is a lack of research describing the frameworks for cloud testing, such as Perfecto, Xamarin, TestDroid, etc., and providing only general information about it.

3.4 Summary

We are presenting a review of the state-of-the-art knowledge of cloud testing in the area of mobile app development. The three groups mentioned above are (1) mobile app test method; (2) cloud test Evaluation methods; (3) cloud test contribution facts. Mobile application testing methods are

defined as functional and non-functional testing methods. The second group evaluates cloud-based testing methods by comparing the study with traditional testing and presents the challenges of cloud testing. The facts of the contribution in the last groups are models and frameworks and describe the benefits for each group.

Based on previous studies, with the growth of mobile apps and the existence of an unlimited number of mobile apps, there are issues surrounding the testing of mobile apps. Cloud testing has thus attracted the attention of developers, testers, and researchers. Cloud testing provides real devices for mobile apps that can be tested on different platforms, such as IOS, Android. Apps can be tested on different devices in parallel and, as well, testing can be carried out anywhere.

In all previous studies, we discussed limitations and gaps. Group one focused on the Android platform and described the mobile app test method on cloud testing. Group two has shown that cloud testing is still new and is a hot topic for research and needs more experiments to validate their solution But group three presents only on some frameworks and neglected others. So the main gap in our research has been identified. Recently, there has been no comparative systematic study of cloud testing frameworks.

Chapter 4: Research Methodology

We describe our methodology in this chapter, which will use in a detailed method. Figure 7 depicts our research methodology. We began our research with a literature review and then chose two cloud mobile automation testing frameworks to evaluate. Following that, we chose the Android application to be tested during the comparison. Then, we looked into related work to determine our best comparison criteria. Following that, we carried out an experimental design and then wrote test cases for a chosen application. Finally, we examined the findings of our experimental study.

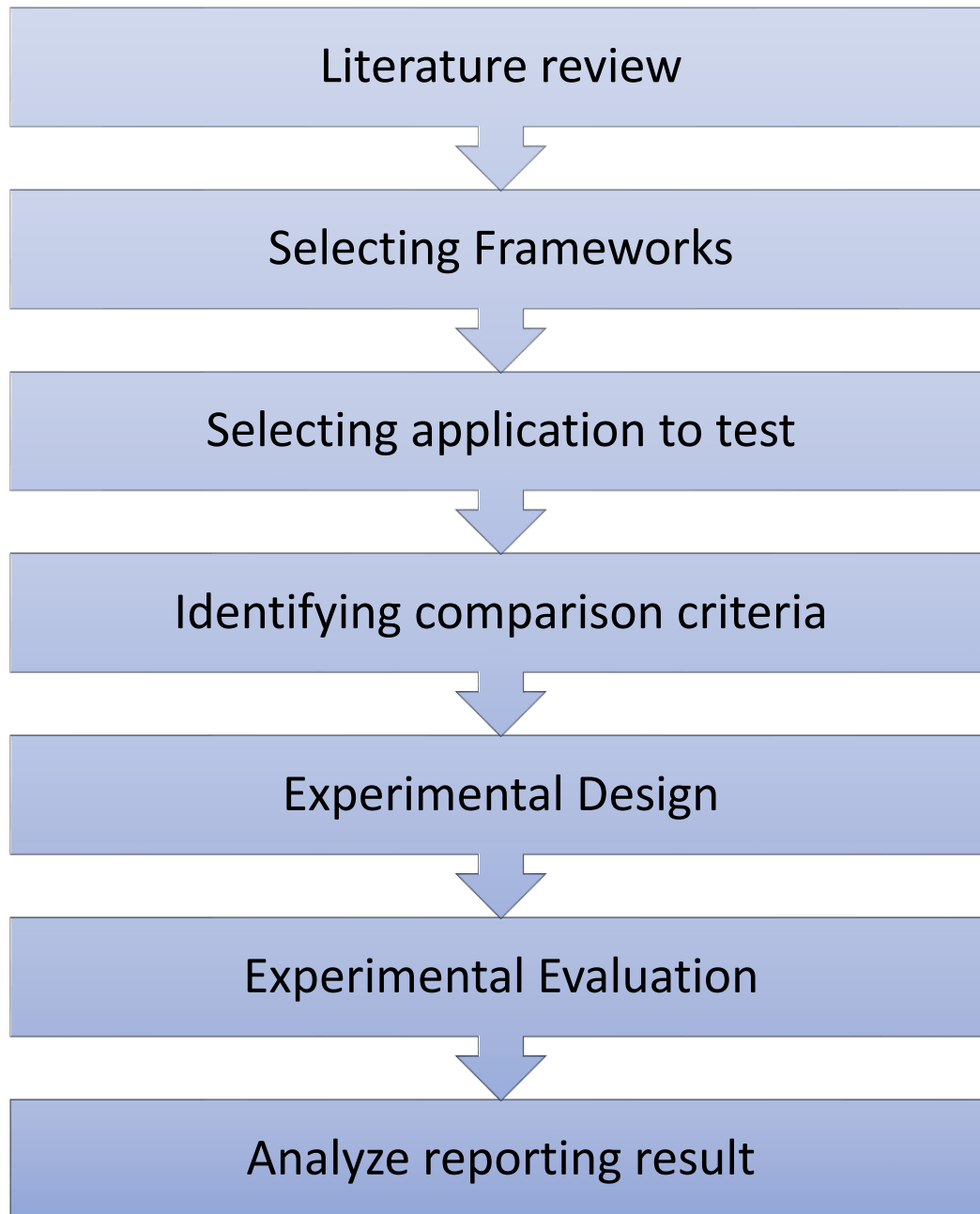


Figure 7 Research Methodology

In section 4.1, we describe the procedures for selecting and comparing cloud testing frameworks based on mobile. We present detailed information about the procedure for selecting the application to test in section 4.2, and we present information about the process for identifying criteria in section 4.3. We present information about data collection in section 4.4, and we describe data analysis in section 4.5. Finally, we describe the experimental design in section 4.6.

4.1 Selecting Frameworks

To automate the testing of mobile apps, cloud testing frameworks are proposed. Several cloud testing frameworks allow the test engineer or developer to automate functional, integration, performance, and acceptance testing of mobile apps. To compare frameworks, we searched for the most popular open source frameworks on a variety of blogs, tutorials, and web pages. In addition, we chose Android as our target due to its market dominance. Finally, we choose an open-source framework that is appropriate for Android.

SoftwareTestingHelp [28] is a most popular testing website established in 2006, has tutorials on QA testing, development. The website states the most popular cloud testing frameworks in 2020:

- Perfecto Mobile (Android and iOS)
- Xamarin Test Cloud (Android and iOS)
- Remote TestKit (Android and iOS)
- pCloudy (Android)
- Scirocco (Android)

QA InfoTech [29] is a testing service organization established in 2003. The company listed the most popular cloud testing frameworks as :

- Xamarin Test Cloud
- Firebase Test Lab
- Kobiton
- Perfecto

About Application Development Trends (ADT) [30] is a website that provides the latest trends to help the organization to enhance efficiency, competitiveness, and productivity. ADT provides 7 Top Device Clouds for Mobile App Testing such as :

- Experitest
- Sauce Labs
- Perfecto
- Kobiton
- Xamarin Test Cloud
- Firebase Test Lab for Android
- AWS Device Farm

EDUCBA is a website that is a leading global provider of skills based education and provides Data Science Courses. In EDUCBA blog [32], they listed the 10 most popular cloud testing frameworks as follows:

- SOASTA Cloud Test
- LoadStorm
- BlazeMeter
- Xamarin Test Cloud
- Nessus
- App Thwack
- Jenkins Dev@Cloud
- Test Link
- Test collab
- Watir

As a result of our research and after seeing most of the blogs, we have discovered that Perfecto and Xamarin are the most widely used to test mobile apps on the cloud. Each of the four resources

that we have mentioned above chooses Xamarin as the best cloud testing framework, and three are mentioned to Perfecto. So, we have selected these two frameworks for our research.

4.2 Selecting Application

We chose the Android DroidWeight health application, as shown in Figure 8, to evaluate the cloud testing frameworks. Its objectives include tracking weight, height, and other parameters. It is a free app available from the Google Play Store [31].

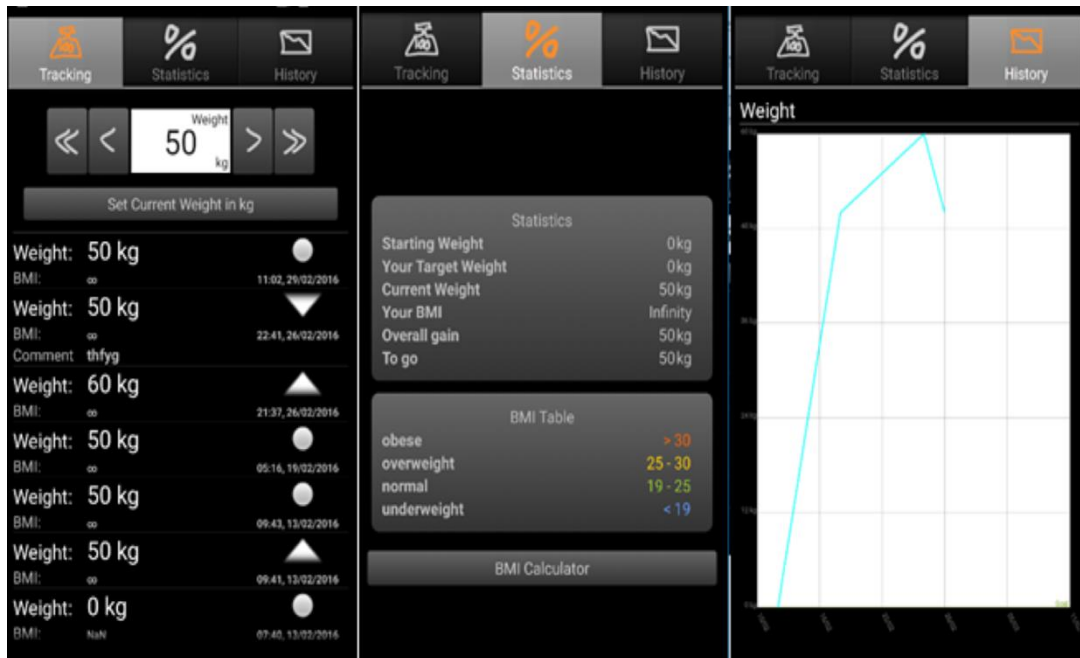


Figure 8 Droid Weight health application

We chose our target as the Android application for this research for several reasons. At first, Android is open-source and allows research freedom. Also, the Android platform achieves the largest target share of the mobile device.

Finally, the DroidWeight health application was chosen because it is an open-source application that is also free to download to any device from the Google Play store. Moreover, the DroidWeight health application provides functionalities that allow us to test the capabilities of two frameworks. For example, as shown in Figure 8, it has three different screens for tracking, history, and statistics. In addition, the statistics screen includes a button that allows you to enter your weight and height to calculate your BMI. As a result, it is a better application for testing framework capabilities.

4.3 Identifying Criteria

To identify criteria that aid in the evaluation of cloud testing frameworks, we first looked into mobile testing checklists. In addition, we investigated the properties, behaviors, potential user actions, and requirements of the mobile apps that must be tested. We discovered about 30 criteria, some of which were related to integration, usability, and installation, as well as GUI testing. Furthermore, the criteria included providing general information about frameworks such as ease of use and test results reports. We defined four criteria in our research . The criteria for evaluating mobile cloud test frameworks that will be used in our research are shown in Table 2.

Table 2 Evaluation criteria for mobile cloud test frameworks

C	Criteria
C1	Ease of Learning
C2	Execution speed of scripts
C3	Test result report formats
C4	Retention rate

- Criteria 1 represents whether the frameworks can be learned easily in a small-time or take time to understand capabilities and features (Kaur et al.,2013).
- Criteria 2 represents execution response time to show which frameworks are faster (Kaur et al.,2011).
- Criteria 3 represents the abilities of frameworks to generate test reports to show whether the script has passed or failed when running the test case ((Ali et al.,2018).
- Criteria 4 represents what you remember later on (Kaur et al.,2013).

4.4 Data Collection

This section will explain how we will collect our data. An experiment will be used to achieve our goals and compare cloud testing frameworks. Table 3 shows how each criterion was measured during the experiment.

Table 3 Data Collection for evaluation criteria

C	Criteria	Data Collection
C1	Ease of Learning	Time
C2	Execution speed of scripts	Time
C3	Test result report formats	Is it generating reports? Format report.
C4	Retention rate	Time, All features easy remembers

4.5 Data Analysis

After we identified the criteria, we have evaluated the cloud testing framework concerning our set criteria. Before the experiment, we searched on paper, blogs, and tutorials for each criterion to determine whether frameworks meet it. After experimenting, we compared the result and then compared each framework to show which one is best.

Different criteria were chosen to compare selected frameworks. Each criterion is significant because it aids in the comparison of the various frameworks. These frameworks are compared to each other in order to answer the research questions. The chosen criteria are classified into three categories: usability experience, technical requirements, and performance capabilities.

The usability experience category will display how easy tools can be configured and used. It is an important category because it will help to understand and learn of it. The usability experience category along their definition are listed such as :

Table 4 Usability Experience Category

Usability Experience Category	
Ease of Learning	This criterion will tell if it is easy or difficult to learn.
Retention rate	This criterion will tell if it is easy or difficult to remember it later.

The second category is the Technical Requirement and will give a quick overview of the frameworks. The Technical Requirement are listed such as:

Table 5 Technical Requirement Category

Technical Requirement Category	
Test result report	This criterion will display the ability of the frameworks to show test result reports.

The third category is performance capabilities is listed below:

Table 6 Performance Capabilities Category

Performance Capabilities Category	
Execution speed of scripts	This criterion will determine the speed and time of testing.

We will compare two cloud testing frameworks in our research by creating test cases and identifying criteria.

4.6 Experiment Design

The Hypothesis should be specific in order to determine the experiment's design specifications. An experiment hypothesis will be discussed in this section, followed by the experiment procedure.

4.6.1 Experiment Hypothesis

We will compare two cloud testing frameworks in our research by creating test cases and identifying criteria. Hypothesis testing is also used to determine whether there is enough evidence in a sample data set to conclude that a specific statement is true or false. For hypothesis testing, there are two hypotheses. A null hypothesis is a statement stating that there is no difference between them. The other type of hypothesis is alternative hypotheses, which are states that want to prove it is true (Lazar et al.,2017).

The hypothesis can be generated based on the findings of the literature review to derive variables and conditions (Lazar et al.,2017). The research objectives drive the formulation of hypotheses, which are precisely stated in the following statements.

The following null hypotheses will be tested in this research :

- *H₀₁: There is no significant difference between the frameworks (Perfecto, Xamarin) according to usability experience.*
- *H₀₂:There is no significant difference between the frameworks (Perfecto, Xamarin) according to Technical Requirement*
- *H₀₃: There is no significant difference between the frameworks (Perfecto, Xamarin) according to Performance Capabilities.*
- *H₀₄: There is no significant difference between the frameworks (Perfecto, Xamarin) according to fault detection.*

The following Alternative hypotheses will be tested in this research :

- *H₁₁: There is a significant difference between the frameworks (Perfecto, Xamarin) according to usability experience.*
- *H₁₂:There is a significant difference between the frameworks (Perfecto, Xamarin) according to Technical Requirement*

- *H₁₃: There is a significant difference between the frameworks (Perfecto, Xamarin) according to Performance Capabilities.*
- *H₁₄: There is a significant difference between the frameworks (Perfecto, Xamarin) according to fault detection.*

Variables

Following the generation of accurate hypotheses, the variable can now be defined clearly to be accurately measured. The dependent variables are the measured outcomes, and the independent variable is the controlled conditions under which the framework is used.

Independent variables

This refers to change factors or causes on dependent variable values that are independent of a participant's behavior (Oehlert et al., 2010).

Independent variables is: Used Framework.

(Perfecto, Xamarin)

Dependent variables

It refers to the outcomes or effects of experimentation that are dependent on the status of the independent variables or the performance of participants, such as how easy the frameworks are to learn and understand (Oehlert et al., 2010).

variables is: Execution time, Ease to use.

Group Design

The condition can be defined after identifying the independent variables and hypothesis. As a result, the conditions are Perfecto, Xamarin framework. The experiment in this research is a group experiment described in experiment research (Sheremeta et al., 2018) in which participants can be assigned to multiple conditions Table 7.

Table 7 Group Design

Phase	Group	Used framework	Task
1	1	Perfecto	Testing a mobile app using Perfecto framework.
	2	Xamarin	Testing a mobile app using Xamarin framework.
2	1	Xamarin	Testing a mobile app using Xamarin framework.
	2	Perfecto	Testing a mobile app using Perfecto framework.

4.6.2 Experiment Procedure

After identifying the research hypotheses and specifying their design, we should adequately prepare for the experiment to avoid obstacles and reduce risks, then the application stage can begin successfully. We selected an Android mobile application and two cloud testing frameworks to

compare it. With each framework, we have put in place the same procedure that has allowed us to compare two frameworks. The experiment procedure will be used for the following two frameworks:

- We begin by carefully designing tasks in order to make them as adequate and as brief as possible.
- We implement training material in the form of a short document, a five-minute video, and a test procedure.
- We choose and recruit participants with care, and then schedule the session.
- We begin the session by playing the prepared movie, and then monitor it to record spent times with a stop watch.
- We analyze the collected data using MS-Excel to report the results.

Chapter 5 Experiment Evaluation

The experimental evaluation of the framework comparison parameter will be covered in this chapter. The comparison of frameworks is based on three evaluation parameters: performance capabilities, technical requirement category, and usability experience category. The usability experience category is based on a closed-question questionnaire that was used to assess the usability of various frameworks. The closed question allowed participants to provide feedback on their usability experience. The performance and technical requirement categories contain information about the testing process for mobile apps.

5.1 Experiment

The planned experiment's goal is to compare two chosen frameworks using the criteria outlined in Chapter 4. In addition, the purpose of this experiment is to investigate the effects of various criteria on each framework. We used the open-source Droidweight Android app. In addition, a three-week experiment was conducted in which groups one and two implemented the first framework, followed by another framework and a specific exercise assigned to all participants on subsequent days. The majority of participants were subjected to an experiment during working hours, while others were subjected to an experiment outside of working hours. At the end of the experiment, a prepared questionnaire was introduced to question participants about which one was the best.

5.2 Usability Experience

According to José Antonio and Freddy Paz (Paz et al.,2016), usability testing can define as "the extent to which any product used by specified users to achieve specific objectives with effectiveness, efficiency, and satisfaction in a specific context of use." In order to evaluate the selected frameworks (Perfecto and Xamarin), a level for ease of learning and ease of use specific. Additional parameters, such as programming skills and user interface, are added.

5.2.1 Selection of Participants

A total of 6 participants were selected for the experiment to meet the criteria for usability testing. These participants work in different companies (Harri, Asal, and Foothill) and will all conduct the same experiments using two cloud testing frameworks. We divide the participants into two groups. Group one will conduct usability testing criteria using the Perfecto framework, then the Xamarin framework, while group two will conduct usability testing criteria using the Xamarin framework and then the Perfecto framework.

We carefully selected participants who were interested in testing mobile apps and who had not previously used the selected frameworks. Initially, 10 participants were selected for the experiment, but only 8 participants were accepted. Later, 2 informed us that they were busy and could not attend the experiments. In these 6 participants, 4 participants worked in mobile testing (Senior QA Engineer) and 2 participants had no experience in mobile testing (Junior QA Engineer), Table 8.

Table 8 Group Description

Group	Description	Experience
<i>Group 1</i>	<i>Two senior QA Engineer One Junior QA Engineer</i>	<i>Senior 1 (4 years) Senior 2 (5 years) Junior1 (one year)</i>
<i>Group 2</i>	<i>Two senior QA Engineer One Junior QA Engineer</i>	<i>Senior 3 (5 years) Senior 4 (6 years) Junior2 (one year)</i>

5.2.2 Scenarios based exercise

We have chosen the same exercise for all participants. The exercise is written in the English language and covers the functionality of the chosen framework. In fact, it is not easy to test all the functionality of the framework, but we have designed the exercise to achieve the objective of this comparative study.

The exercise was designed with the selected usability criteria in mind so that participants could answer the questionnaires and then we could answer the research questions. We first learned both frameworks before creating the scenario. In addition, in order to write a good exercise, we look at previous comparative studies about testing mobile apps (Paz et al.,2016). After writing it, we run it through both frameworks to see if anything is missing.

5.2.3.Test Location

The experiment will be carried out on laptops for the participants (Dell CORE i5). The exercise was sent by email or Facebook to perform the test in their spare time. So, without interruption, they can conduct the test. We conducted this experiment in the participant's workplace, where the internet speed was consistent. After the participants had finished the test, they replied to the questionnaire.

5.2.4 Usability Test Equipment and Material

To control the experiment, we've provided each participant with a video. In the video, we're recorded how each framework works. Upon completion of the test, we sent questionnaires to each participant individually using the Google Form.

5.2.5. Experiment Information

Before they started the experiment, we explained the purpose of assessing the usability of the software testing framework (Perfecto and Xarmin). When participants completed the test exercise, they answered the questionnaire by selecting the rating scale for each question.

5.2.6 Questionnaire for Usability Evaluation (Ease of learn)

A questionnaire will be conducted on the basis of closed questionnaires to evaluate the usability category. An exercise has been designed to reflect the usability of the frameworks selected. The objective of the questionnaire is to provide feedback on the selected frameworks for assessing the effectiveness of each framework. The questionnaire is prepared based on the "Post-Study System Usability Questionnaire (PSSUQ)" (Lewis et al.,1995). The User Agreement for usability is based on the efficiency, user interface, learning ease, and ease of use usability agreement.

Table 9 Set of Question

Parameter	Number of Question
<i>Ease of use</i>	5
<i>Sufficiency of learning material Result</i>	2
<i>Programming Skills</i>	1
<i>Fault Detection</i>	1
<i>Totals</i>	9

Questionnaire analyzed using a Likert rating scale with five options, as shown in Figure 9. Following the completion of the experiment over each framework, data collected from six participants. The result will be analyzed and discussed in the next chapter.

Point of scale	1	2	3	4	5
Agreement	Strongly disagree	disagree	undecided	agree	Strongly agree

Figure 9 Likert Rating Scale

5.2.7 Retention Rate Evaluation

We meet with participants after a week to review the same two chosen frameworks and determine which one is easier to remember. The first group examines the Perfecto framework first, followed by Xamarin, while the second examines the Xamarin framework first, followed by Perfecto. Furthermore, each person performs the same exercise without assistance.

5.3 Technical Requirements

The category of Technical Requirements is to determine if the frameworks are feasible, and we will evaluate the test result report for each framework. It is important to show the execution of the result after running the test to show the failed or passed while running the test (Kaur et al.,2011). The technical requirements are useful for QA engineers or researchers looking for the best cloud testing framework. After test execution, it is critical to obtain a result in order to perform effective analysis and demonstrate whether the test scripts failed or passed while running a test.

A test result report can be evaluated in two ways. First, we will add questions to questionnaires and also ask participants questions about the final testing report to see if it is good and what details are missing, and then we will analyze the results. The second approach is to read other comparative studies on testing reports before analyzing framework reports.

5.4 Performance testing

The aim of the framework assessment is to gather the information that may be useful for performance testing in order to meet specific needs. Maximum execution time must be included in the definition of the framework performance category. The criteria chosen for this research are the speed of execution of the scripts which is to determine the time of testing.

Speed of Execution: Evaluate the total run test time of each user.

When comparing the performance of various frameworks, it is critical to look at how quickly the scripts are executed. The execution speed is calculated in terms of the average test run time of the number of transactions, and the fastest one is then evaluated.

To determine the execution speed of each framework, the following step will be used:

1. We will visit the [Perfecto](#) and [Xamarin](#) frameworks websites.
2. We will select the same phone.
3. We will install the Android app.
4. We will test a mobile app.
5. We will execute test in the cloud
6. We will analyze the result.
7. We will fix the detects if we find them.
8. We evaluate the total time by using record and playback video.

Chapter 6 Result and Discussions

We discuss and analyze the results in depth in this chapter in light of the evaluation in Chapter 5. Graphs are used to discuss all of the observations made during an experiment or survey that resulted in answers to research questions. Furthermore, validity threats are investigated to describe the actions that help to mitigate negative consequences.

6.1 Usability experience result

The development of usability testing parameters is critical in determining which framework to use. This section will discuss the questioner's result.

6.1.1 Questioner result

Since the questionnaire follow-up did not include any complicated questions that needed statistical to be analyzed, the answers were transferred to Microsoft Excel, and mean and standard deviations were calculated.

The t-test was chosen to analyze our data because it is applicable when there are two data sets. To determine the difference between the two frameworks, we used the t-test. Microsoft Excel has a function that calculates t-values for any data set. Figure 10 depicts the function that was used.

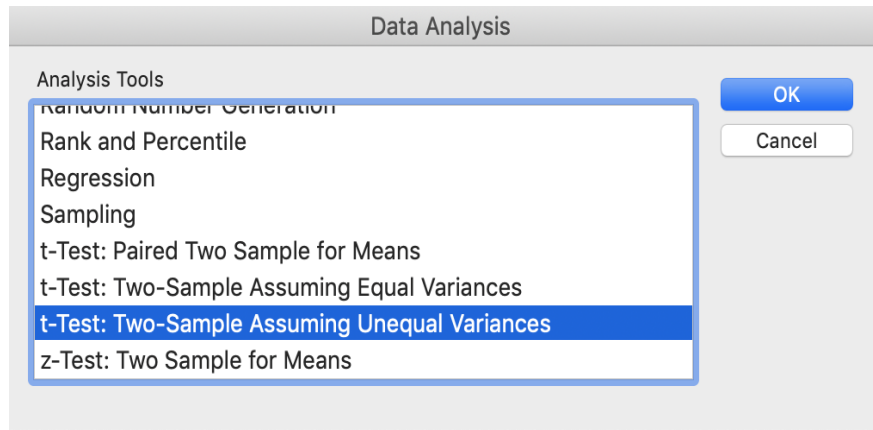


Figure 10 Excel Analysis Snapshot

Following the selection of the data analysis, we chose the t-test: "Two-Sample Assuming Unequal Variance." This tool allows you to select a cell range, as shown in Figure 11.

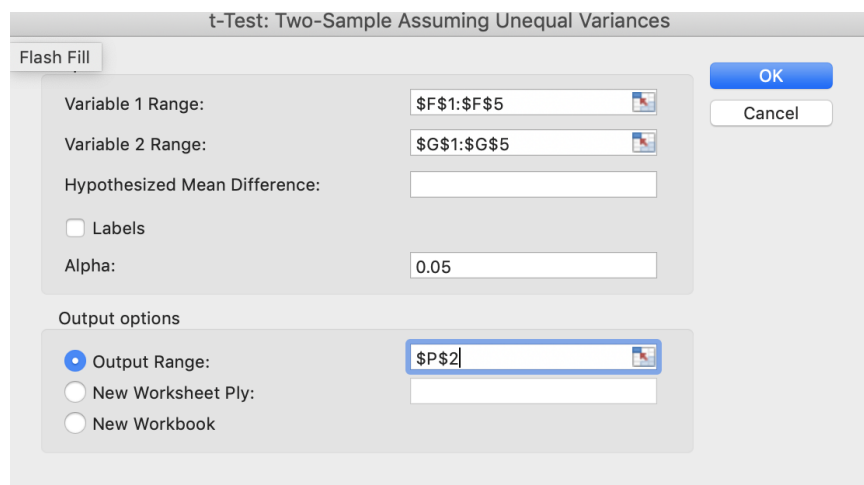


Figure 11 t-test two sample snapshot

When the t-test is evaluated, the result is shown in Figure 12. As shown, the result includes the following information: mean, variance, t-test(t State) value, and degree of freedom.

t-Test: Two-Sample Assuming Unequal Variances		
	Variable 1	Variable 2
Mean	4.6	4
Variance	44.8	26
Observations	5	5
Hypothesized Mean Difference	0	
df	7	
t Stat	0.1594482	
P(T<=t) one-tail	0.43890946	
t Critical one-tail	1.89457861	
P(T<=t) two-tail	0.87781893	
t Critical two-tail	2.36462425	

Figure 12 Example for t-test

Ease of use Result

It is very important to calculate which framework is ease to use, as below question, we can show all question on [Appendix](#) :

Q2: *I'm satisfied with how easy it is to test the mobile app using this framework.*

Table 10 Mean and SD for question two

Framework	Mean	Standard Deviation	t-Test: Two-Sample Assuming Unequal Variances	
			Perfecto	Xamarin
Perfecto	3.6	0.75	Mean	3.666666667
			Variance	0.666666667
Xamarin	2.3	0.745	Observations	6
			Hypothesized Mean Difference	0
			df	10
			t Stat	2.828427125
			P(T<=t) one-tail	0.008950062
			t Critical one-tail	1.812461123
			P(T<=t) two-tail	0.017900123
			t Critical two-tail	2.228138852

As shown in table 10, the results of the two-tailed unpaired t-test for question two confirm that (p-value = 0.017 < 0.05) that means the question has significant differences between the frameworks.

Q3:I could effectively complete the tasks and scenarios using this framework.

Table 11 Mean and SD for question three

Framework	Mean	Standard Deviation	t-Test: Two-Sample Assuming Unequal Variances		
				Perfecto	Xamarin
Perfecto	3.3	0.75	Mean	3.333333333	3.166666667
			Variance	0.666666667	0.166666667
Xamarin	3.16	0.0.37	Observations	6	6
			Hypothesized	0	
			df	7	
			t Stat	0.447213595	
			P(T<=t) one-t	0.33411552	
			t Critical one-	1.894578605	
P(T<=t) two-t	0.66823104				
			t Critical two	2.364624252	

Q6:I found the interface of framework user friendly.

Table 12 Mean and SD for question six

Framework	Mean	Standard Deviation	t-Test: Two-Sample Assuming Unequal Variances		
				Perfecto	Xamarin
Perfecto	4.2	0.37	Mean	4.166666667	2.333333333
			Variance	0.166666667	0.266666667
Xamarin	2.3	0.47	Observations	6	6
			Hypothesized Mean Difference	0	
			df	9	
			t Stat	6.8219104	
			P(T<=t) one-tail	3.8564E-05	
			t Critical one-tail	1.83311293	
P(T<=t) two-tail	7.7128E-05				
			t Critical two-tail	2.26215716	

Q9: I felt comfortable using this framework

Table 13 Mean and SD for question eight

Framework	Mean	Standard Deviation	t-Test: Two-Sample Assuming Unequal Variances		
				Perfecto	Xamarin
Perfecto	3.6	0.75	Mean	3.66666667	2.33333333
			Variance	0.66666667	0.26666667
Xamarin	2.3	0.47	Observations	6	6
			Hypothesized	0	
			df	8	
			t Stat	3.38061702	
			P(T<=t) one-t	0.00481642	
			t Critical one-	1.85954804	
P(T<=t) two-t	0.00963285				
			t Critical two	2.30600414	

Q10: It was easy to learn to use this framework.

Table 14 Mean and SD for question nine

Framework	Mean	Standard Deviation	t-Test: Two-Sample Assuming Unequal Variances		
				Perfecto	Xamarin
Perfecto	3.6	0.75	Mean	3.83333333	2.5
			Variance	0.96666667	0.3
Xamarin	2.5	0.5	Observations	6	6
			Hypothesized	0	
			df	8	
			t Stat	2.901905	
			P(T<=t) one-t	0.00991684	
			t Critical one-	1.85954804	
P(T<=t) two-t	0.01983369				
			t Critical two	2.30600414	

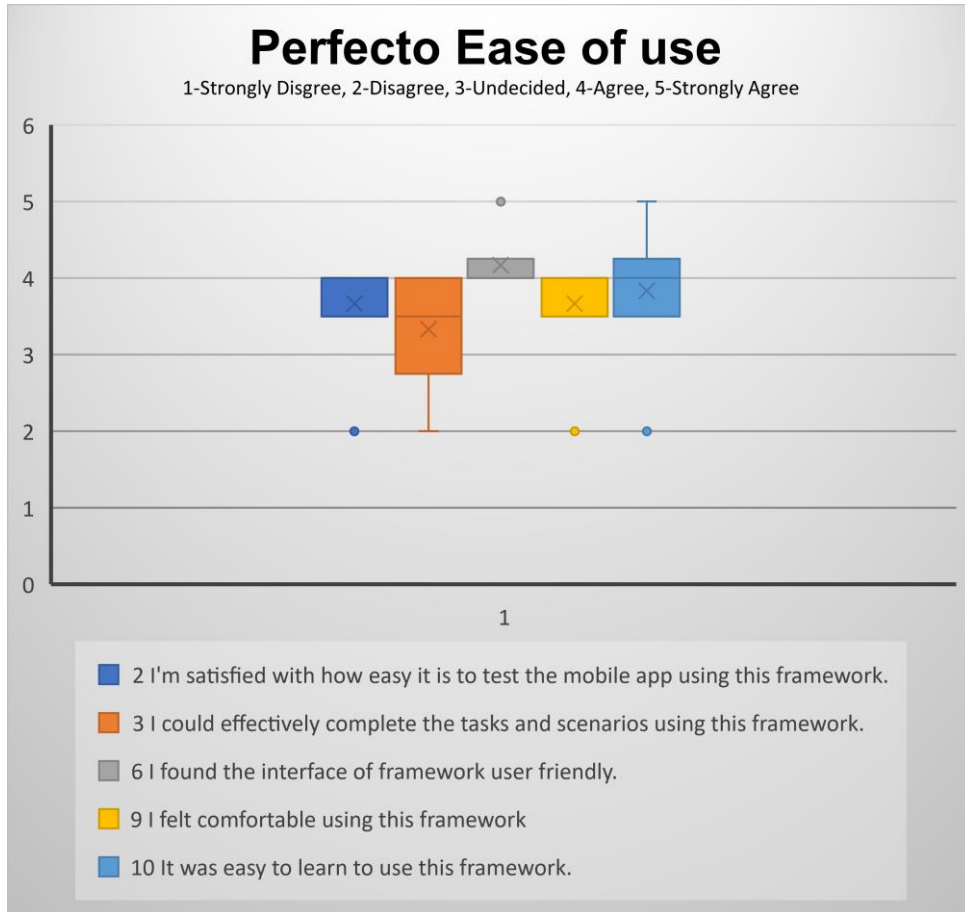


Figure 13 Perfecto Ease of use

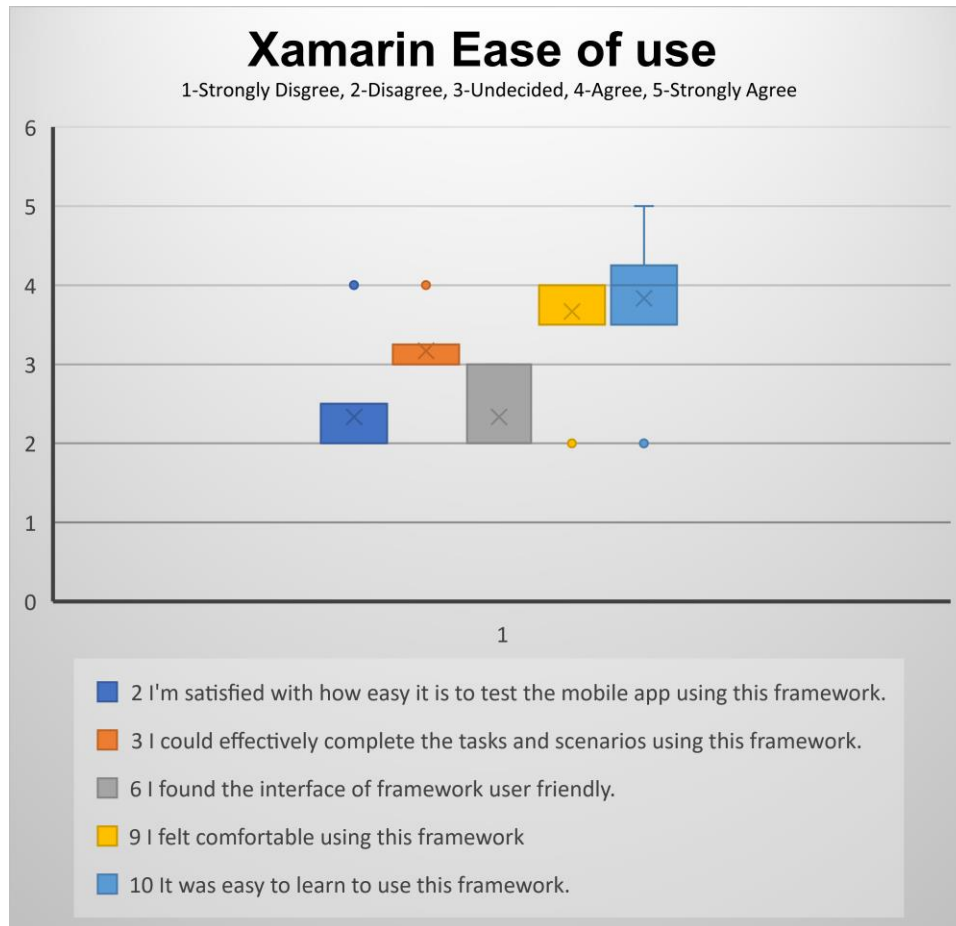


Figure 14 Xamarin Ease of use

As illustrated in Figures 13 and 14 (y axis represents Likert scale as Figure 9), the Perfecto framework produces more "agree" responses than the Xamarin framework. As a result, the response indicates that the Perfecto framework is simple to understand. Furthermore, as shown in previous tables, the majority of questions have p-values less than 0.05, indicating that the two frameworks have a significant difference in terms of ease of learning.

Sufficiency of learning material Result

Q1: The information (such as on-line help, on-screen messages and other documentation) provided with this framework was clear.

Table 15 Mean and SD for question one

Framework	Mean	Standard Deviation	t-Test: Two-Sample Assuming Unequal Variances	
			Perfecto	Xamarin
Perfecto	3.8	0.89	Mean	3.33333333
Xamarin	3.3	0.745	Variance	0.66666667
			Observations	6
			Hypothesized	0
			df	10
			t Stat	0.958314847
			P(T<=t) one-	0.180249834
			t Critical one	1.812461123
			P(T<=t) two-	0.360499668
t Critical two	2.228138852			

Q7: The documentation provided for the framework was easy to understand.

Table 16 Mean and SD for question seven

Framework	Mean	Standard Deviation	t-Test: Two-Sample Assuming Unequal Variances	
			Perfecto	Xamarin
Perfecto	3.16	0.89	Mean	3.83333333
Xamarin	3.8	0.68	Variance	0.56666667
			Observations	6
			Hypothesized	0
			df	9
			t Stat	-1.31876095
			P(T<=t) one-	0.109908878
			t Critical one	1.833112933
			P(T<=t) two-	0.219817756
t Critical two	2.262157163			

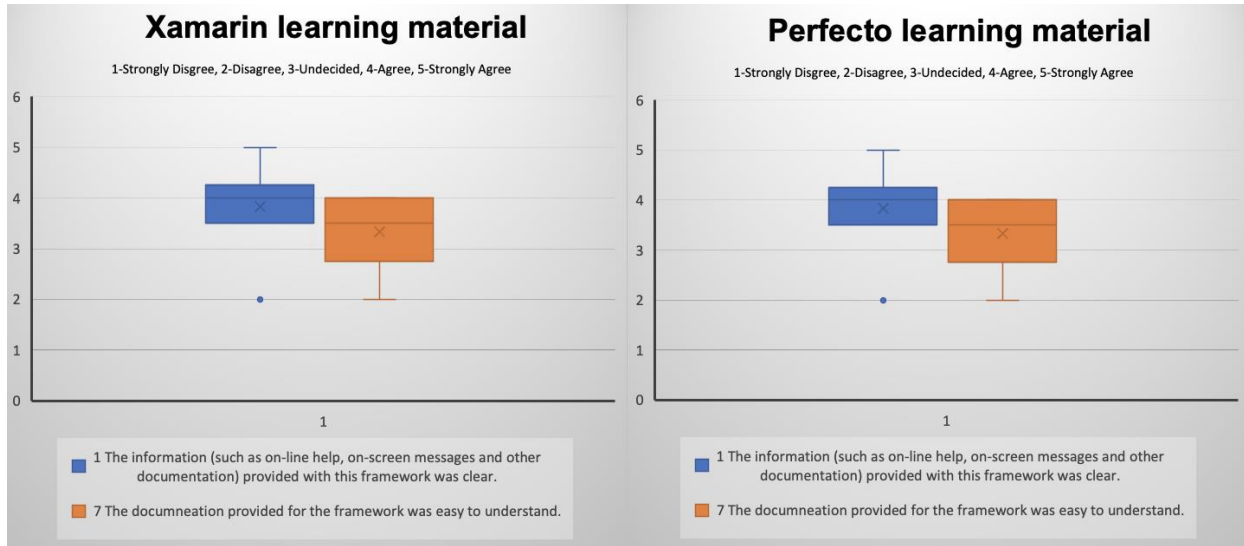


Figure 15 Sufficiencyof learning material Result

As illustrated in Figures 15, the Perfecto framework produces more "agree" responses than the Xamarin framework. As a result, the response indicates that the Perfecto framework material is more helpful than the Xamarin framework. But in the t-test, the result shows both frameworks have useful material and no significance between them.

Programming Skills

Q5: I need programming knowledge

Table 17 Mean and SD for question five

Framework	Mean	Standard Deviation	t-Test: Two-Sample Assuming Unequal Variances	
			Perfecto	Xamarin
Perfecto	3.5	0.76	Mean	3.5
			Variance	0.7
			Observations	6
Xamarin	3.8	0.68	Hypothesized	0
			df	10
			t Stat	-0.7254763
			P(T<=t) one-	0.24239561
			t Critical one	1.81246112
			P(T<=t) two-	0.48479121
			t Critical two	2.22813885

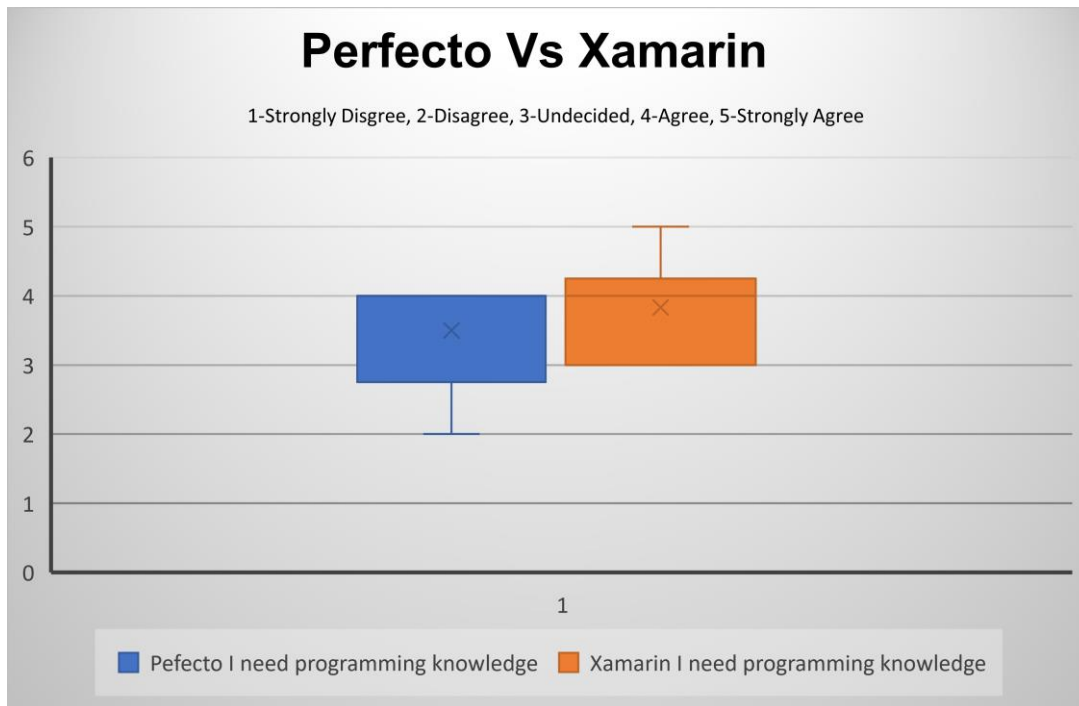


Figure 16 Programming skills material

As illustrated in Figures 16, the Perfecto and Xamarin framework produces "agree" responses more than other responses. As a result, the response indicates that both frameworks need programming skills to write scripts. And no significant difference between frameworks in terms of programming language.

We agreed with participants to hold a meeting after the first week of an experiment to assess retention rates and continue evaluating usability testing parameters. Without any assistance, all participants review two frameworks (Perfecto and Xamarin) to see which one is the easiest to remember. We divided participants into two groups, as previously stated: the first group reviewed the Perfecto framework before moving on to the Xamarin framework, and the second group reviewed the Xamarin framework before moving on to the Perfecto framework.

We interviewed participants after they had finished reviewing both frameworks to determine which one was the best. According to the results, four participants 67% (two in group one and two

in group two) agreed that Perfecto is easier to remember than Xamarin and also easier to understand when testing mobile apps as Figure 17. In contrast, two participants 33% stated that Xamarin and Perfecto are easy to remember and that both frameworks are a good choice for testing mobile apps on real devices in the future.

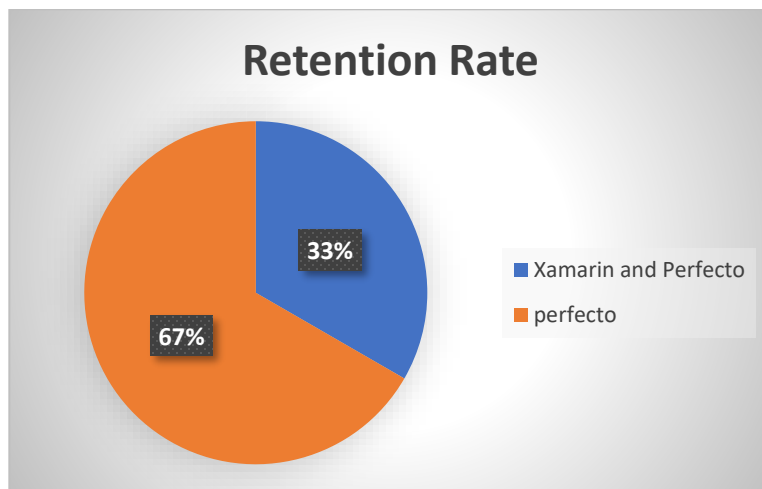


Figure 17 Retention Rate for Perfecto and Xamarin

RQ1: How cloud testing frameworks for mobile apps are compared in terms of Usability testing parameters?

Perfecto outperforms Xamarin in terms of ease of learning and retention rate, according to previous results. However, there is no difference in terms of the sufficiency of learning material, and programming skills; both require programming skills and documentation is useful.

Finally, we can conclude that the Perfecto framework is superior to the Xamarin framework in terms of usability testing. As a result, the null hypothesis can be rejected, and both frameworks have significant differences.

Fault Detection Result

Q4: Whenever I made a mistake using the system, I could recover easily and quickly

Table 18 Mean and SD for question four

Framework	Mean	Standard Deviation	t-Test: Two-Sample Assuming Unequal Variances		
			perfecto	xamarin	
Perfecto	3.3	0.95	Mean	3.3333333	3
			Variance	1.8666667	0.8
			Observations	6	6
Xamarin	3	0.81	Hypothesized Mean Difference	0	
			df	9	
			t Stat	0.5	
			P(T<=t) one-tail	0.3145356	
			t Critical one-tail	1.8331129	
			P(T<=t) two-tail	0.6290713	
t Critical two-tail	2.2621572				

As shown in above Table 20, the two-tailed unpaired t-test result for the Perfecto vs. Xamarin framework is ($p\text{-value} = 0.31 > 0.05$). As a result, we conclude that the no significant difference hypothesis is valid. In other words, the framework is significantly different, so H_{14} is rejected. In addition in Figure 18, we can see the most answer is agreed for if the system has mistaken, we can recover easily and quickly.

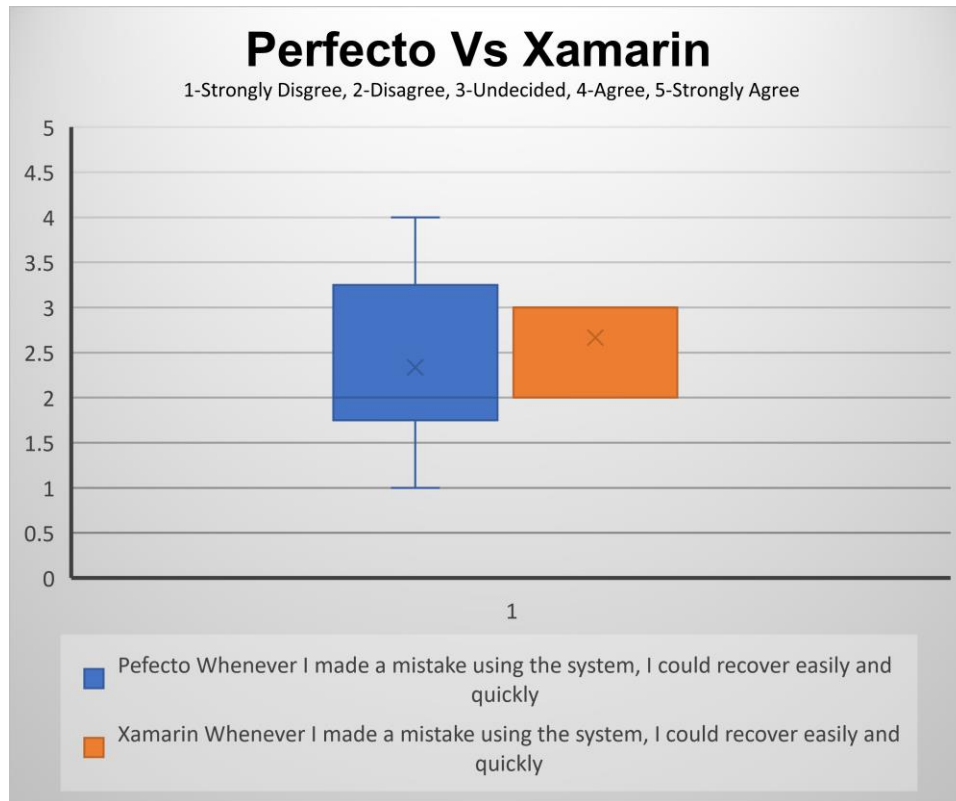


Figure 18 Fault Detection Graph

RQ4: How cloud testing frameworks for mobile apps are compared in terms of fault detection?

We compared both frameworks based on fault detection. As a result, the null hypothesis is accepted, while the alternative hypothesis is rejected.

6.2 Technical requirement result

Following the completion of the test scripts, it is critical to display the execution results in order to determine whether the test scripts failed or passed. We ask participants about test result reports in order to evaluate the technical requirement results.

Q8: After I finished the test, I found all the information on the report.

Table 19 Mean and SD for question eight

Framework	Mean	Standard Deviation	t-Test: Two-Sample Assuming Unequal Variances		
			Perfecto	Xamarin	
Perfecto	4.16	0.4	Mean	4.16666667	4.5
			Variance	0.56666667	0.3
Xamarin	4.5	0.5	Observations	6	6
			Hypothesized Mean Difference	0	
			df	9	
			t Stat	-0.877058	
			P(T<=t) one-tail	0.20162941	
			t Critical one-tail	1.83311293	
			P(T<=t) two-tail	0.40325883	
t Critical two-tail	2.26215716				

As shown in the above Table 19, the two-tailed unpaired t-test result for the Perfecto vs. Xamarin framework is ($p\text{-value} = 0.2 > 0.05$). As a result, we conclude that the no significant difference hypothesis is valid. In other words, the framework is significantly different, so H_{12} is rejected. Also, in Figure 19, we can show that most of the result is agreed for both frameworks. Furthermore, we can see in Figure 19 that the majority of the results are agreed-on responses at both frameworks.

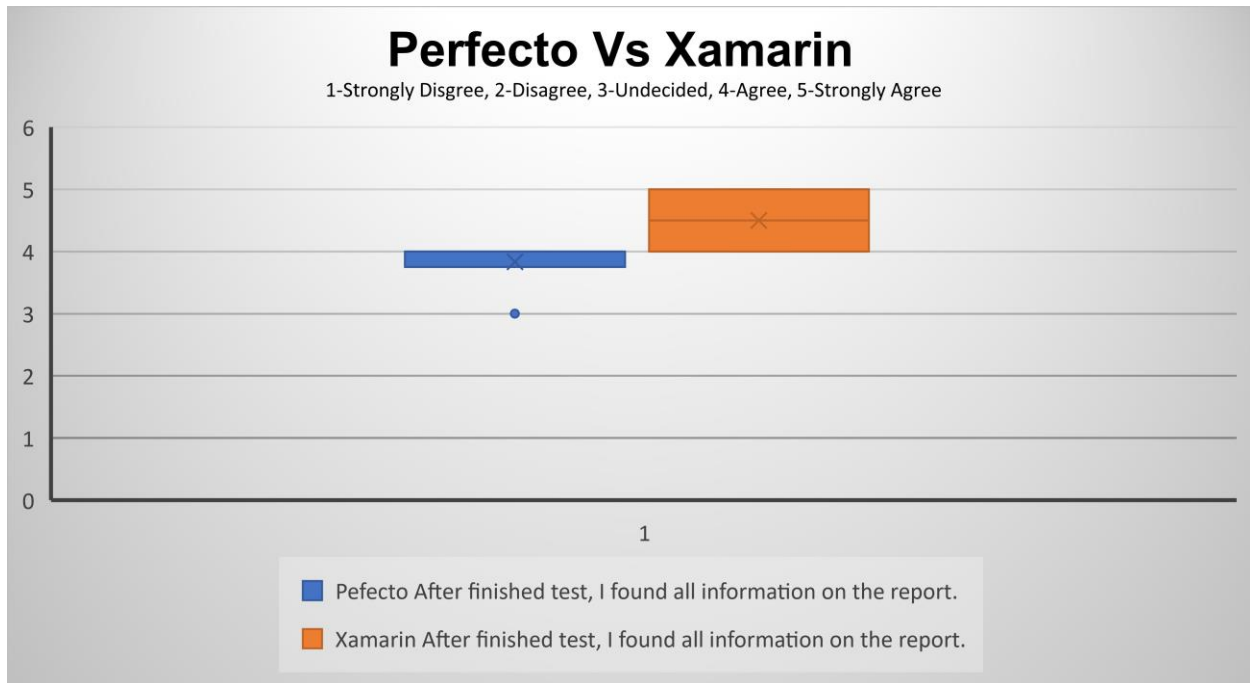
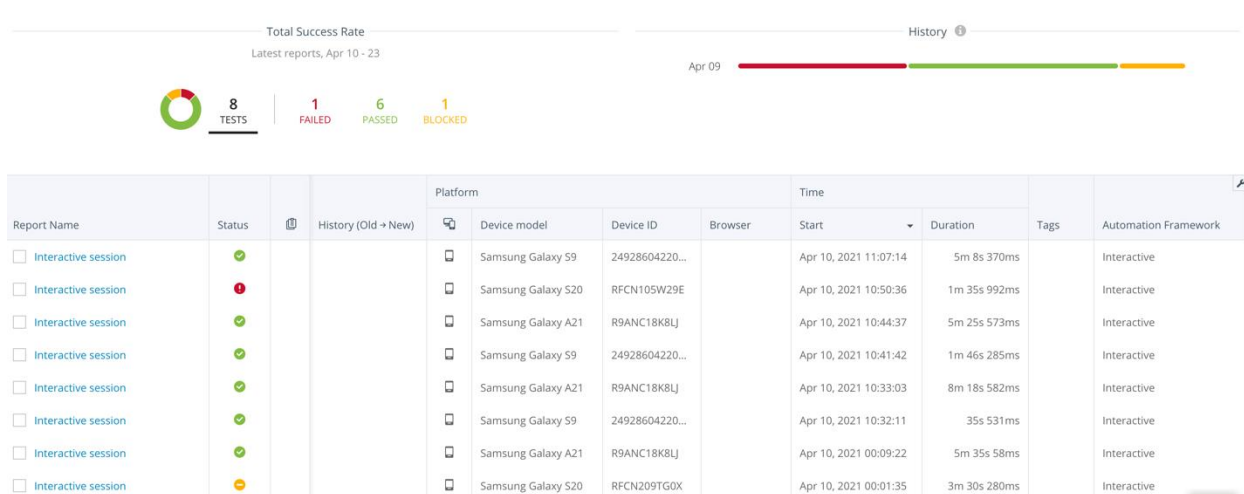
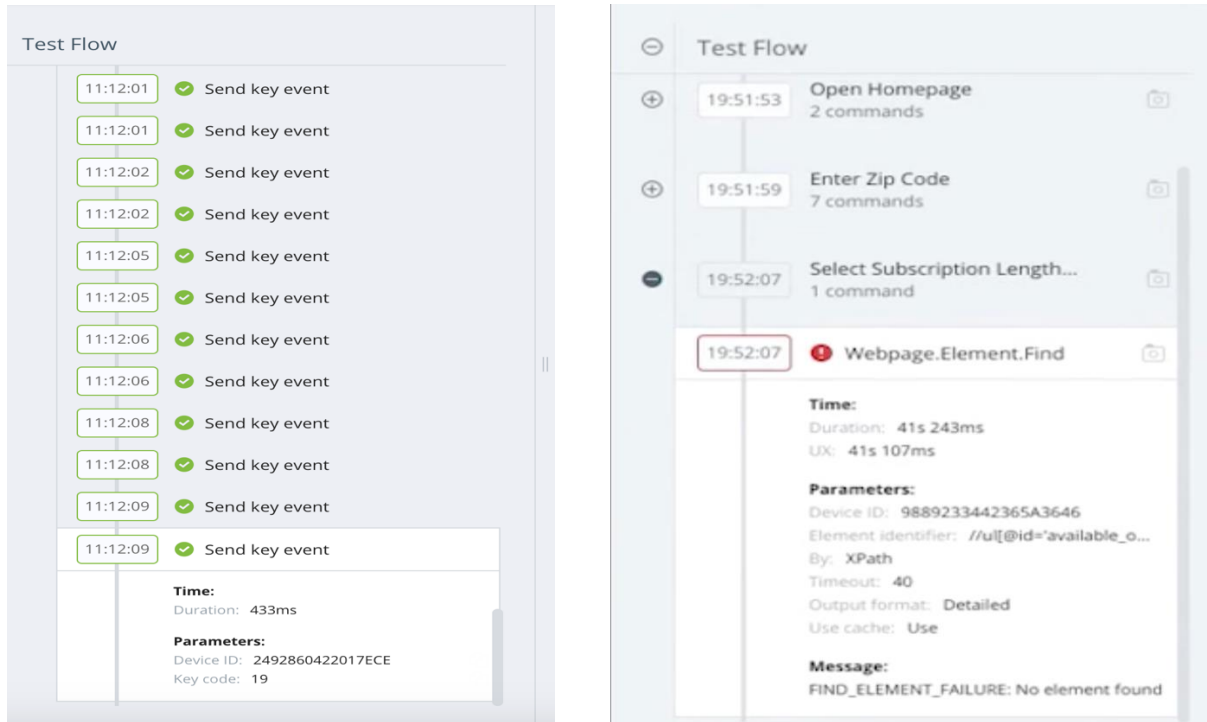


Figure 19 Perfecto Vs Xamarin test result report

After the participants have completed the test and answered the questionnaires, I conduct an interview with them to inquire about the testing report and which one is superior. Perfecto is better than Xamarin, according to two participants, and the report is easier to understand. However, three participants stated that both frameworks have approximately good reports that are easy to understand. One participant, on the other hand, stated that Xamarin has a better report.

Now, I'll describe the comparison of the two frameworks based on the test results report. The Perfecto framework provides a test summary. As illustrated in Figure 20, these results are simple and straightforward. It depicts the test flow, which provides a summary of each step (time and parameter). Also, having all of your results in one place (a "single pane of glass") makes it much easier to return the result when you need it as illustrated in Figure 20. Finally, Perfecto shows you what went wrong in their test so you can fix bugs quickly.



The summary of Perfecto reports contains Figure 21:

- Execution time: end time, duration, date, and start time.
- The Header which contains a summary of Perfecto result for all test

- Number of all test

In the Xamarin framework, you can see a summary of all results as Figure 21, how many failed or successful tests there were, how much time they took and also other parameters as Figure 22. But it does not provide information about each test step as Perfecto does.

Test runs					
Date	Version	Duration	Status	Results	Devices
Apr 11, 2021, 10:06:50 PM	1.0 (1)	1 min	✓ PASSED	1 test passed	1
Feb 1, 2021, 4:13:01 PM	1.0 (1)	1 min	✓ PASSED	1 test passed	1
Feb 1, 2021, 4:04:24 PM	1.0 (1)	0 mins	✓ PASSED	1 test passed	1
Feb 1, 2021, 1:30:14 AM	1.0 (1)	0 mins	✓ PASSED	1 test passed	1
Feb 1, 2021, 12:43:11 AM	1.0 (1)	1 min	✓ PASSED	1 test passed	1

Figure 22 All test in Xamarin Framework

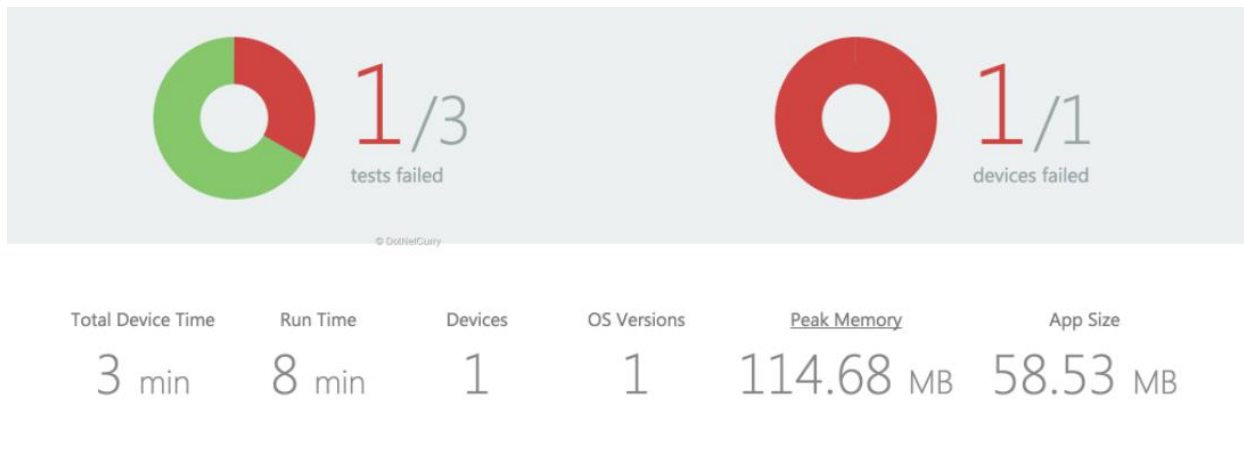


Figure 23 Xamarin test report

RQ2: *How cloud testing frameworks for mobile apps are compared in terms of technical requirements?*

We compared both frameworks based on test result reports to find the best answer to this research question. As demonstrated in the preceding description, both frameworks have a good report that is simple to understand. As a result, the null hypothesis is accepted, while the alternative hypothesis is rejected.

6.3 Performance capabilities result

The results of performance testing are taken into account in experiments on the selected application. This testing is necessary to obtain a performance comparison for each framework. We chose the Samsung Galaxy S9 Figure 24 for both frameworks before beginning the experiment to evaluate performance capabilities.

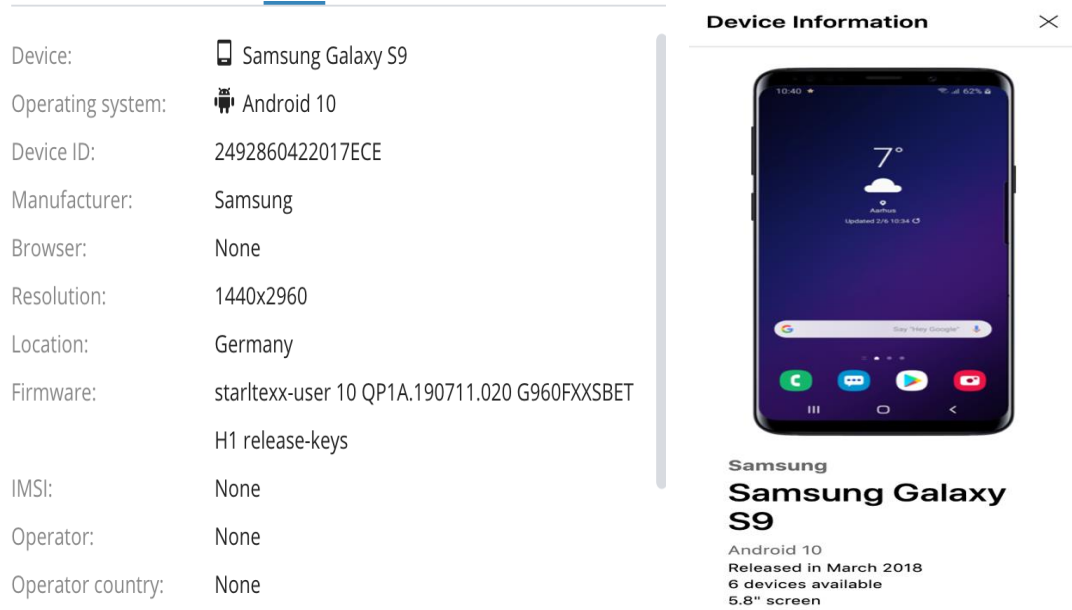


Figure 24 Samsung Galaxy S9

The execution speed calculated by the Perfecto cloud testing framework's record and play was used to calculate the performance capabilities. The total time required to complete a test is defined as "start test run time plus end test run time." The illustration depicts the result of the Perfecto cloud testing framework. The results show the total time required to run the test Figure 25, Figure 26.

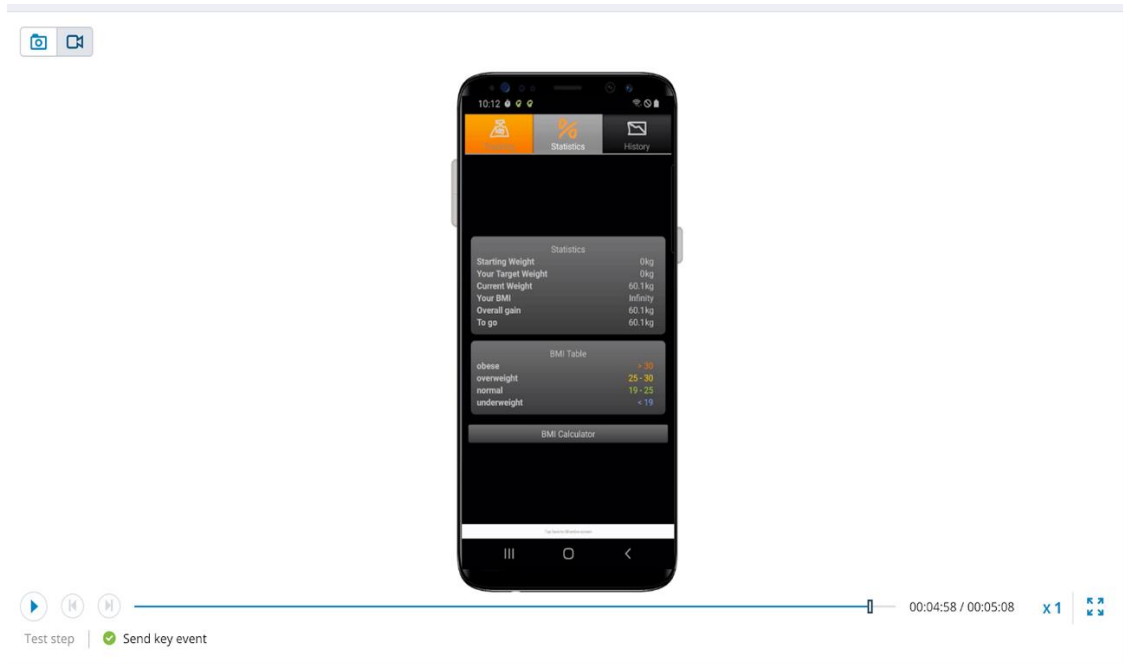


Figure 25 Perfecto Cloud Testing Time Execution

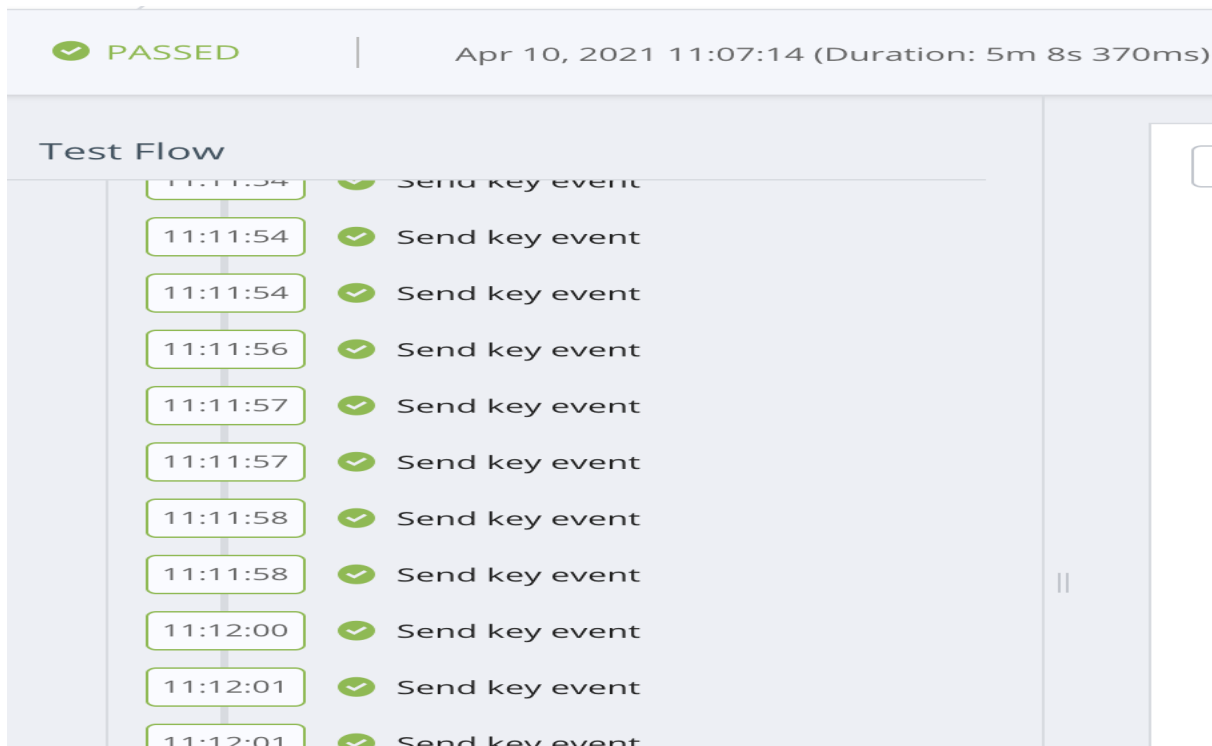


Figure 26 Perfecto Cloud Testing Time Execution

The Xamarin cloud testing framework's execution speed also calculated the total time required to run the test. Figure 27 shows the total, which is greater than the Perfecto framework. So, in our

case, the Perfecto cloud testing framework has arrived a little sooner than the Xamarin cloud testing framework.

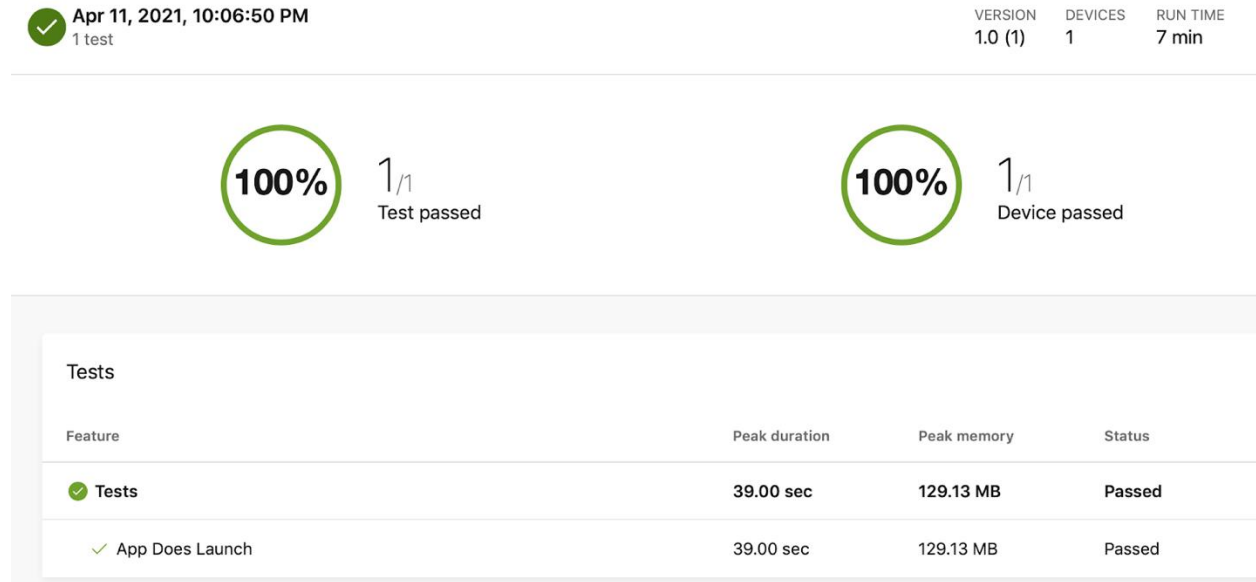


Figure 27 Xamarin cloud testing Time Execution

To determine which framework produced the best results, I ran the test on another device, such as the Samsung Galaxy A21, for both frameworks (Perfecto and Xamarin). The end result is the same : Perfecto takes less time (5m) Figure 28 than Xamarin (6m) Figure 29.

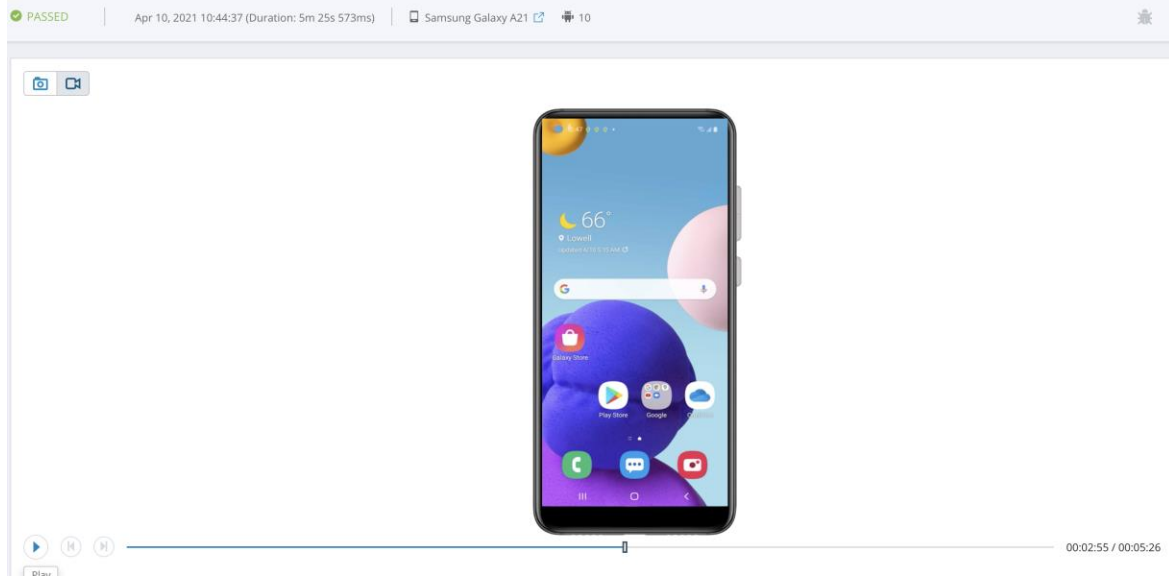


Figure 28 Perfecto Execution time

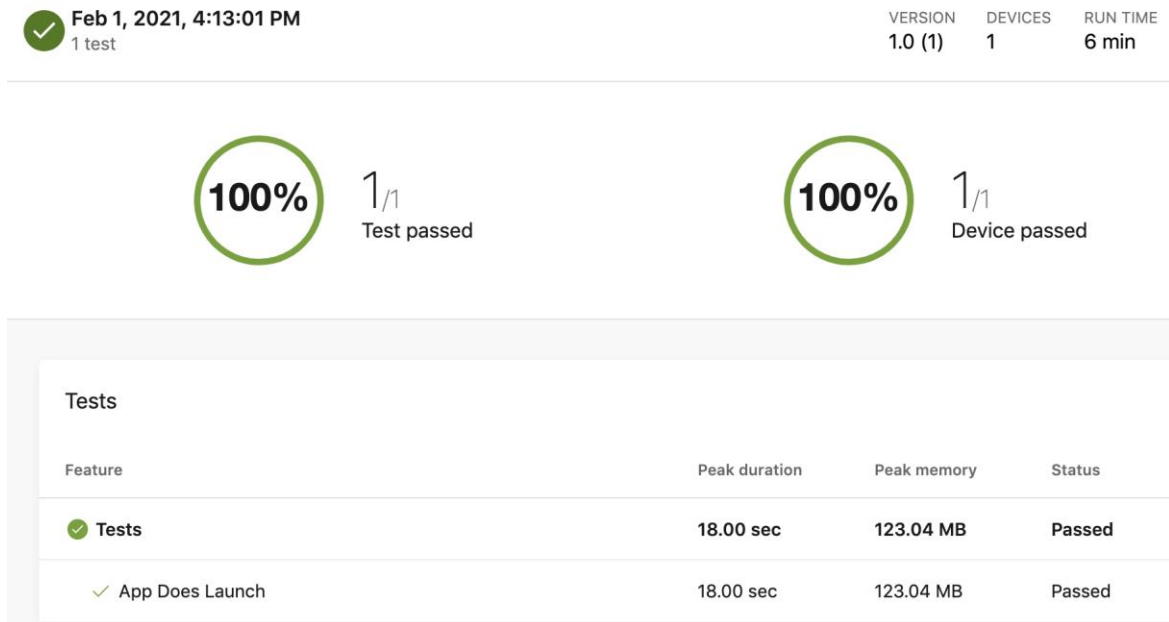


Figure 29 Xamarin Execution time

RQ3: How cloud testing frameworks for mobile apps are compared in terms of performance capabilities?

To find the best answer to this research question, we compared both frameworks based on execution time speed. Perfecto cloud testing framework is rated higher in terms of performance because it requires less execution time speed. As a result, we discovered that the Perfecto cloud testing framework outperforms the Xamarin cloud testing framework.

Based on the above results, we can conclude that H_{03} is rejected and H_{13} is accepted. So that, there is a significant difference between the frameworks (Perfecto, Xamarin) according to Performance Capabilities.

6.4 Discussion

In this section, we will discuss the results which were presented in the previous section. Also, it presents the experience and insight of the researchers based on the results. Based on the literature review there is no study focused on comparing cloud testing frameworks except one study that focused on analyzing advantages and disadvantages for some cloud testing frameworks. However, this study is a comparison of the most commonly used cloud testing frameworks (Perfecto and Xamarin) based on a literature review to assist quality assurance engineers or researchers in selecting the best one.

Perfecto and Xamarin cloud testing frameworks each have their own internal process and architecture, which form the basis of a framework comparative study in terms of usability testing, performance testing, and technical requirements. Furthermore, each framework has advantages that make it superior to others, and quality assurance engineers choose which one is best for them based on criteria such as ease of learning, execution speed, cost, and so on. Finding frameworks that meet the context's requirements, on the other hand, is a challenge. For that, context's requirements such as "what technologies are we using, how much do we want to invest, what kind of testing tool do we need, and so on" must be carefully derived.

Perfecto and Xamarin both provide a free trial period for testing a web or mobile app on a real device, but Xamarin provides a longer free trial period than Perfecto. However, Xamarin pricing information is only provided by the software provider in order to be negotiated, whereas the Perfecto framework pricing information can be found on the internet. In addition, Xamarin provides over 2500 devices for the iOS and Android platforms, but Perfecto provides over 10000 different devices.

Perfecto cloud testing is a public cloud device service with network and test automation capabilities. The Xamarin framework is a multiplatform cloud testing framework. Test engineers can connect to hundreds of devices using Visual Studio or a web portal. Following that, the test engineer can select from a wider range of smartphone devices. The device screen then appears in the web browser, and the device is ready to use.

One of the key variables to measure usability testing by initial questionnaire after the experiment. The participants test mobile apps using both frameworks before answering the questionnaire. We divided the participants into two groups of three people each. Although one's first instinct would be to say that the first frameworks used by the participants are the best, this is not the case. The result shows the most participants answer that the Perfecto is better than Xamarin according to usability testing based on ease to learn and retention rate. When compared to the previous literature review, the Perfecto framework has a rating of 8.5, and Xamarin has a rating of 7.8, indicating that the previous result is correct.

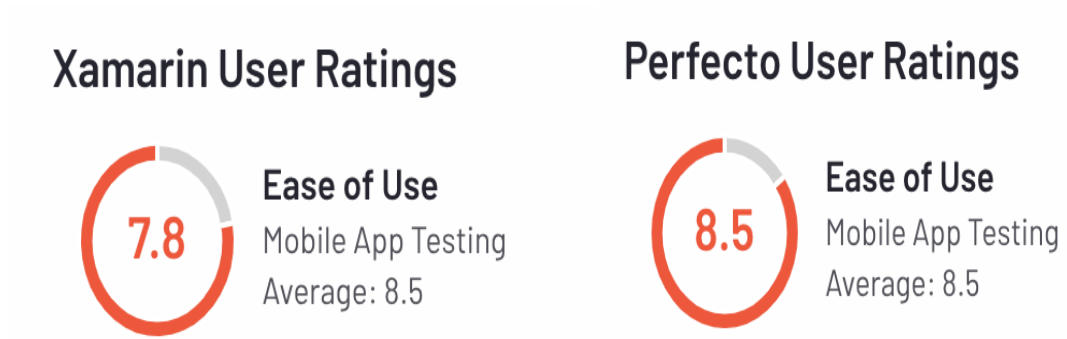


Figure 30 (A) Xamarin Rating, (B) Perfecto Rating

Performance testing results, on the other hand, show that the perfecto framework executes faster than Xamarin, which took longer. We tried various devices with both frameworks, but the results showed that the Perfecto framework outperformed Xamarin. Perfecto may be a better choice for professional quality assurance engineers looking for better performance. But, when comparing two frameworks based on technical requirements, the results show no difference and both provide a good technical report that is easy to understand, helps to detect errors, and provides quick feedback.

We also check websites, books, and literature reviews to back up our findings. As shown in Figure 31, all reviews preferred Perfecto over Xamarin frameworks, with Perfecto receiving 4.4/5 stars from 63 reviews and Xamarin receiving 4.0/5 from 9 reviews [74] [75].



<p>+ Add Product</p>	 <p>Perfecto</p> <p>🕒 Optimized for quick response</p> <p>Try for free</p> <hr/> <p>Answer a few questions to help the Perfecto community</p> <p>Have you used Perfecto before?</p> <p><input type="button" value="Yes"/> <input type="button" value="No"/></p>	 <p>Xamarin Test Cloud</p> <p>Get a quote</p> <hr/> <p>Answer a few questions to help the Xamarin Test Cloud community</p> <p>Have you used Xamarin Test Cloud before?</p> <p><input type="button" value="Yes"/> <input type="button" value="No"/></p>
<p>Star Rating</p>	<p>★★★★★ 63 reviews</p>	<p>★★★★☆ 9 reviews</p>

Figure 31 Perfecto and Xamarin Review

6.5 Threats to Validity

In this section, we will discuss experiment setting in order to account for significant factors that can affect the validity of results and to avoid errors that can reduce the reliability of the results. According to (Creswell et al., 2017), we will analyze four types of validity: external, internal, conclusion, and construct validity.

6.5.1 External validity

External validity is concerned with the generalization of results to others environments outside the scope of the research rather than the experiment itself (Feldt et al., 2010).

- The most dangerous external threat is the network infrastructure's unstable internet speed. To counteract this threat, we conducted this experiment in the participant's workplace, where the internet speed was consistent.
- There was a threat that the participants' and their experiences would have an impact on the outcomes. To mitigate this risk, we divided the participants into two groups based on their prior mobile testing experience and knowledge.

6.5.2 Internal validity

Internal validity is the investigation of how certain we can be that the treatment caused the outcome. Sometimes there are reasons that can cause a result that we can't control or measure (Feldt et al., 2010).

The following are the internal validity threats in this study:

- We had never used the selected frameworks before. To overcome this threat, we learned how to use each one through tutorials and online videos. The experiment began after we had finished learning.
- Quality assurance engineers and used laptops were chosen. Participants were carefully chosen to be familiar with testing mobile apps while also having diverse experiences. In addition, “Dell CORE i5” Lab PCs were selected to be quick and convenient for participants.

6.5.3 conclusion validity

Conclusion validity is concerned with how certain the treatment used in an experiment is truly related to the actual outcome obtained (Feldt et al., 2010).

- The incorrect conclusion is an important conclusion threat. To counteract this threat, we chose participants who had experience testing mobile apps as well as strong Verification

and Validation knowledge. In addition, we compared the results of each participant to the results of the researcher.

6.5.4 construct validity

The motivation for construct validity is based on the relationship between the theory underlying the experiment and the interpretations. The interpreted outcome may differ from the effect being measured (Feldt et al., 2010).

- There was a threat that the chosen frameworks would fail to meet the experiment's criteria. To address this threat, various websites and literature were examined, and it was determined that the frameworks chosen could meet the criteria.
- The task may be misunderstood by participants. To eliminate this threat, we responded to all of their problems they faced, and the task was double-checked before being sent.

Chapter 7 Conclusion

The main research was that the comparison of most cloud testing frameworks is not thoroughly investigated due to a lack of research. Our research is based on a comparison of two cloud testing frameworks so, we will compare the two most popular cloud testing frameworks (Perfecto and Xamarin). To accomplish this, we conducted extensive research to identify the comparison criteria for frameworks.

The cloud testing frameworks Perfecto and Xamarin were compared in terms of usability, performance, and technical requirements. Xamarin tool offers a free trial period, but we paid to use the Perfecto framework. The usability parameter comparison consists of ease of learning and retention rate. Following the completion of the experiment, the frameworks were evaluated using a questionnaire, and the results revealed that Perfecto outperformed Xamarin in terms of ease of use and retention rate.

In terms of performance parameters, the most important factor is the execution time, and the results show that Perfecto is better than Xamarin frameworks and takes less time. However, when it comes to technical requirements, both frameworks provide the best technical testing report, which is simple to understand and follow the testing steps. The comparison can provide quality assurance engineers with an overview of the potential benefits of cloud testing frameworks and aid in the selection of the best frameworks.

In conclusion, we learned that the cloud testing framework is extremely useful when compared to other testing tools. Furthermore, selecting software frameworks is not easy; it took a lot of effort and time to understand the frameworks. In addition, there are numerous cloud testing frameworks available, making the selection process more difficult.

In my view, I will recommend the tool which is easy to learn how to use it. Hence, in comparison to the Perfecto and Xamarin framework, I will recommend the Perfecto cloud testing framework.

Chapter 8 Future Work

Even though the results presented in this research indicate that Perfecto may be a better choice than Xamarin, more research and experimentation in cloud testing frameworks may be required.

Future research may include additional frameworks and additional participants in order to elicit more responses and thus increase the significance of our findings. Despite the fact that this study only used one mobile app and yielded positive results, a longer study would have necessitated the inclusion of more than one application to test the frameworks.

Finally, we plan to add other platforms, such as iOS, and see if all platforms use the same quality characteristics through a tested mobile app. We also intend to improve the comparison criteria by adding new ones and updating existing ones as more information becomes available.

References

1. Zein, S., Salleh, N., & Grundy, J. (2015, September). Mobile application testing in industrial contexts: an exploratory multiple case-study. In International conference on intelligent software methodologies, tools, and techniques (pp. 30-41). Springer, Cham
2. Zein, S., Salleh, N., & Grundy, J. (2016). A systematic mapping study of mobile application testing techniques. *Journal of Systems and Software*, 117, 334-356.
3. Malini, A., Venkatesh, N., Sundarakantham, K., & Mercyshalinie, S. (2014, December). Mobile application testing on smart devices using MTAAS framework in cloud. In International Conference on Computing and Communication Technologies(pp. 1-5). IEEE.
4. Girardon, G., Costa, V., Machado, R., Bernardino, M., Legramante, G., Basso, F. P., ... & Neto, A. (2020, March). Testing as a service (TaaS) a systematic literature map. In Proceedings of the 35th Annual ACM Symposium on Applied Computing (pp. 1989-1996).
5. Gao, J., Bai, X., & Tsai, W. T. (2011). Cloud testing-issues, challenges, needs and practice. *Software Engineering: An International Journal*, 1(1), 9-23
6. Hosseini, S., Nasiri, R., & Shabgahi, G. L. (2015). A new framework for cloud based application testing. *International Journal of Scientific Engineering and Applied Science (IJSEAS)*, 1(3), 112-118.
7. Kaur, K., & Kaur, A. (2016, March). Cloud era in mobile application testing. In 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)(pp. 1057-1060). IEEE.
8. Ciortea, L., Zamfir, C., Bucur, S., Chipounov, V., & Candea, G. (2010). Cloud9: A software testing service. *ACM SIGOPS Operating Systems Review*, 43(4), 5-10.
9. Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., ... & Zaharia, M. (2010). A view of cloud computing. *Communications of the ACM*, 53(4), 50-58.

10. Riungu, L. M., Taipale, O., & Smolander, K. (2010, November). Research issues for software testing in the cloud. In *2010 IEEE Second International Conference on Cloud Computing Technology and Science* (pp. 557-564). IEEE.
11. Mittal, V., Nautiyal, L., & Mittal, M. (2017, December). Cloud Testing-The Future of Contemporary Software Testing. In *2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS)* (pp. 131-136). IEEE.
12. Xavier, M. G., Matteussi, K. J., França, G. R., Pereira, W. P., & De Rose, C. A. (2017, March). *Mobile application testing on clouds: Challenges, opportunities and architectural elements*. In *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)* (pp. 181-185). IEEE.
13. Nachiyappan, S., & Justus, S. (2015). *Cloud testing tools and its challenges: A comparative study*. *procedia computer Science*, 50, 482-489.
14. Gandhewar, N., & Sheikh, R. (2010). Google Android: An emerging software platform for mobile devices. *International Journal on Computer Science and Engineering*, 1(1), 12-17.
15. Hoog, A. (2011). *Android forensics: investigation, analysis and mobile security for Google Android*. Elsevier.
16. Halpern, M., Zhu, Y., Peri, R., & Reddi, V. J. (2015, March). Mosaic: cross-platform user-interaction record and replay for the fragmented Android ecosystem. In *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)* (pp. 215-224). IEEE.
17. Lu, X., Liu, X., Li, H., Xie, T., Mei, Q., Hao, D., ... & Feng, F. (2016, May). PRADA: Prioritizing Android devices for apps by mining large-scale usage data. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)* (pp. 3-13). IEEE.
18. Park, J. H., Park, Y. B., & Ham, H. K. (2013, June). Fragmentation problem in Android. In *2013 International Conference on Information Science and Applications (ICISA)*(pp. 1-2). IEEE.

19. https://en.wikipedia.org/wiki/Cloud_testing
20. Kaur, K., & Kaur, A. (2016, 16-18 March 2016). Cloud era in mobile application testing. Paper presented at the 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom).
21. Kamran, M., Rashid, J., & Nisar, M. W. (2016). Android fragmentation classification, causes, problems and solutions. *International Journal of Computer Science and Information Security*, 14(9), 992.
22. Kaur, M. (2016, April). Testing in the cloud: New challenges. In *2016 International Conference on Computing, Communication and Automation (ICCCA)* (pp. 742-746). IEEE
23. Shrivastva, A., Gupta, S., & Tiwari, R. (2014). Cloud based testing techniques (CTT). *International journal of computer applications*, 104(5).
24. Mujtaba, S., Petersen, K., Feldt, R., & Mattsson, M. (2008). Software product line variability: A systematic mapping study. *School of Engineering, Blekinge Inst. of Technology*.
25. Kitchenham, B., & Charters, S. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering. School of Computer Science and Mathematics, Keele University.
26. Felderer, Michael & C. Carver, Jeffrey. (2018). Guidelines for Systematic Mapping Studies in Security Engineering. Empirical Research for Software Security: Foundations and Experience.
27. Suvesh, T.K. & Sanoj, S. (2014). How to Evaluate a Mobile Test Automation Tools for your Application?. Retrieved from <http://www.rapidvaluesolutions.com/how-to-evaluate-a-mobile-test-automation-tool-for-your-application/>
28. SoftwareTestingHelp Retrieved December 16, 2020 from <https://www.softwaretestinghelp.com/best-mobile-testing-tools/#Cloud-Based-Mobile-Testing-Tools-and-Services>

29. QA InfoTech Retrieved December 16, 2020 from <https://qainfotech.com/top-5-tools-for-cloud-based-mobile-testing/>
30. Application Development Trends Retrieved December 16, 2020 from <https://adtmag.com/blogs/dev-watch/2017/05/device-clouds.aspx>
31. Droid Weight – Android Application in Google Play Store. (n.d.). Retrieved January 5, 2021 from <https://play.google.com/store/apps/details?id=de.delusions.measure&hl=tr>
32. EDUCB. Retrieved January 13, 2021 from <https://www.educba.com/cloud-testing-tools/>
33. Tao, D., Lin, Z., & Lu, C. (2015). Cloud platform based automated security testing system for mobile internet. *Tsinghua Science and Technology*, 20(6), 537-544.
34. Zhang, S., & Pi, B. (2015, March). Mobile functional test on TaaS environment. In *2015 IEEE Symposium on Service-Oriented System Engineering* (pp. 315-320). IEEE.
35. Ali, A., Maghawry, H. A., & Badr, N. (2018). Automated parallel GUI testing as a service for mobile applications. *Journal of Software: Evolution and Process*, 30(10), e1963.
36. Liu, C. H. (2017, July). A cloud platform for compatibility testing of Android multimedia applications. In *International Conference on Frontier Computing* (pp. 169-178). Springer, Singapore.
37. Ahmad, A. A. S., & Andras, P. (2018). Measuring and testing the scalability of cloud-based software services. In *2018 Fifth International Symposium on Innovation in Information and Communication Technology (ISIICT)* (pp. 1-8). IEEE.
38. Lanui, A., & Chiew, T. K. (2019, December). A Cloud-Based Solution for Testing Applications' Compatibility and Portability on Fragmented Android Platform. In *2019 26th Asia-Pacific Software Engineering Conference (APSEC)* (pp. 158-164). IEEE.
39. Ya'u, B. I., Salleh, N., Nordin, A., Idris, N. B., Abas, H., & Alwan, A. A. (2019). A systematic mapping study on cloud-based mobile application testing. *Journal of Information and Communication Technology*, 18(4), 485-527.

40. Bertolino, A., Angelis, G. D., Gallego, M., García, B., Gortázar, F., Lonetti, F., & Marchetti, E. (2019). A systematic review on cloud testing. *ACM Computing Surveys (CSUR)*, 52(5), 1-42.
41. Riungu-Kalliosaari, L., Taipale, O., Smolander, K., & Richardson, I. (2016). Adoption and use of cloud-based testing in practice. *Software Quality Journal*, 24(2), 337-364.
42. Ahmad, A. A. S., Brereton, P., & Andras, P. (2017, July). A systematic mapping study of empirical studies on software cloud testing methods. In *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*(pp. 555-562). IEEE.
43. Qusef, A., Issa, L., Ayoubi, E., & Murad, S. (2019, December). Challenges and opportunities in cloud testing. In *Proceedings of the Second International Conference on Data Science, E-Learning and Information Systems* (pp. 1-7).
44. Tao, C., & Gao, J. (2016). Cloud-based mobile testing as a service. *International Journal of Software Engineering and Knowledge Engineering*, 26(01), 147-152.
45. Temkar, R., Gadekar, S., & Shah, D. (2015). Cloud Based Mobile Application Testing. *International Journal of Science, Engineering and Technology Research*, 4(6), 2097-2102.
46. Tao, C., Gao, J., & Li, B. (2015, October). Cloud-based infrastructure for mobile testing as a service. In *2015 Third International Conference on Advanced Cloud and Big Data* (pp. 133-140). IEEE.
47. Rojas, I. K. V., Meireles, S., & Dias-Neto, A. C. (2016, September). Cloud-based mobile app testing framework: Architecture, implementation and execution. In *Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing* (pp. 1-10).
48. Liu, C. H., Chen, S. L., & Chen, W. K. (2015, June). Improving resource utilization of a cloud-based testing platform for Android applications. In *2015 IEEE International Conference on Mobile Services* (pp. 202-208). IEEE.
49. Al-Ghuwairi, A. R., Eid, H., Aloran, M., Salah, Z., Baarah, A. H., & Al-Oqaily, A. A. (2016, March). A mutation-based model to rank testing as a service (taas) providers in

- cloud computing. In *Proceedings of the International Conference on Internet of things and Cloud Computing* (pp. 1-5).
50. Bertolino, A., Calabrò, A., Marchetti, E., Sala, A. C., de Hita, G. T., Pop, I. D. G., & Gowtham, V. (2018, December). Perceived needs and gains from an industrial study in Cloud testing automation. In *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)* (pp. 238-244). IEEE.
 51. Tao, C., & Gao, J. (2017). On building a cloud-based mobile testing infrastructure service system. *Journal of systems and software*, *124*, 39-55.
 52. Chen, J., Yan, H., Wang, C., & Liu, X. (2020). A Hybrid Cloud Testing System Based on Virtual Machines and Networks. *KSII Transactions on Internet & Information Systems*, *14*(4).
 53. Chawla, P., Chana, I., & Rana, A. (2019). Framework for cloud-based software test data generation service. *Software: Practice and Experience*, *49*(8), 1307-1328.
 54. Lo, W. T., Liu, X. L., Sheu, R. K., Yuan, S. M., & Chang, C. Y. (2015, July). An architecture for cloud service testing and real time management. In *2015 IEEE 39th Annual Computer Software and Applications Conference* (Vol. 3, pp. 598-603). IEEE.
 55. Nurul, M., & Quadri, S. M. K. (2019, January). Software Testing Approach for Cloud Applications (STACA)–Methodology, Techniques & Tools. In *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*(pp. 19-25). IEEE.
 56. Villanes, I. K., Costa, E. A. B., & Dias-Neto, A. C. (2015, June). Automated mobile testing as a service (AM-TaaS). In *2015 IEEE World Congress on Services* (pp. 79-86). IEEE.
 57. Kaur, H., & Gupta, G. (2013). Comparative study of automated testing tools: selenium, quick test professional and testcomplete. *Int. Journal of Engineering Research and Applications*, *3*(5), 1739-1743.

58. Kaur, M., & Kumari, R. (2011). Comparative study of automated testing tools: Testcomplete and quicktest pro. *International Journal of Computer Applications*, 24(1), 1-7.
59. Fazzini, M., Gorla, A., & Orso, A. (2020, September). A Framework for Automated Test Mocking of Mobile Apps. In *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)* (pp. 1204-1208). IEEE.
60. Grgurina, R., Brestovac, G., & Grbac, T. G. (2011, May). Development environment for Android application development: An experience report. In *2011 Proceedings of the 34th International Convention MIPRO* (pp. 1693-1698). IEEE.
61. Holla, S., & Katti, M. M. (2012). Android based mobile application development and its security. *International Journal of Computer Trends and Technology*, 3(3), 486-490.
62. Ghanem, T., & Zein, S. (2020, October). A Model-based approach to assist Android Activity Lifecycle Development. In *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)* (pp. 1-12). IEEE.
63. Lettner, M., Tschernuth, M., & Mayrhofer, R. (2011, February). Mobile platform architecture review: Android, iphone, qt. In *International Conference on Computer Aided Systems Theory* (pp. 544-551). Springer, Berlin, Heidelberg.
64. Moonsamy, V., Rong, J., & Liu, S. (2014). Mining permission patterns for contrasting clean and malicious Android applications. *Future Generation Computer Systems*, 36, 122-132.
65. Yerima, S. Y., Sezer, S., McWilliams, G., & Muttik, I. (2013, March). A new Android malware detection approach using bayesian classification. In *2013 IEEE 27th international conference on advanced information networking and applications (AINA)* (pp. 121-128). IEEE.
66. Motan, M., & Zein, S. (2020, October). Android App Testing: A Model for Generating Automated Lifecycle Tests. In *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)* (pp. 1-11). IEEE.

67. Lazar, J., Feng, J. H., & Hochheiser, H. (2017). *Research methods in human-computer interaction*. Morgan Kaufmann.
68. Oehlert, G. W. (2010). *A first course in design and analysis of experiments*.
69. Sheremeta, R. M. (2018). Behavior in group contests: A review of experimental research. *Journal of Economic Surveys*, 32(3), 683-704.
70. Creswell, J. W., & Creswell, J. D. (2017). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications.
71. Feldt, R., & Magazinius, A. (2010, July). Validity threats in empirical software engineering research-an initial survey. In *Seke* (pp. 374-379).
72. Paz, F., & Pow-Sang, J. A. (2016). A systematic mapping review of usability evaluation methods for software development process. *International Journal of Software Engineering and Its Applications*, 10(1), 165-178.
73. Lewis, J. R. (1995). IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction*, 7(1), 57-78
74. G2 Retrieved April 16, 2021 https://www.g2.com/products/perfecto/reviews?utf8=%E2%9C%93&filters%5Bkeyphrases%5D=execution%20time&order=g2_default&filters%5Bcomment_answer_values%5D=#survey-response-4596580
75. G2 Retrieved April 16, 2021 from <https://www.g2.com/products/xamarin-test-cloud/reviews#reviews>

Appendix

Task

Name of Framework: Perfecto framework

- i. Open the Chrome Browser.
- ii. Open the following URL : <https://www.perfecto.io>
- iii. Register (add your email, password) and then check your email.
- iv. If you have any problems, please watch the video again or contact me for assistance.
- v. Now, you can start testing using real devices.

Test a mobile app (Droid weight Android mobile app) by following the instructions. :

1. Select Android mobile platform.
2. Select Samsung Galaxy A21 (If you need another device, please ask me).
3. After opening the device, download the Droid weight app from the play store.
4. Open the Droid weight app and register your name and weight.
5. Register new weight, save it, register other weight and save it.
6. Try to register different weights.
7. Click to calculate BMI.
8. Add your weight, height and then click calculate to show the result.
9. Click to report at the Droid weight app.
10. End Test.
11. Save result

Name of Framework: Xamarin framework

- i. Open the Chrome Browser.
- ii. Open the following URL : <https://appcenter.ms/apps>
- iii. Select Sign IN (add your email, password) and then check your email.
- iv. If you have any problems, please watch the video again or contact me for assistance.
- v. Now, you can start testing using real devices.

Test a mobile app (Droid weight Android mobile app) by following the instructions. :

1. Select Android mobile platform.
2. Select Samsung Galaxy A21 (If you need another device, please ask me).
3. After opening the device, download the Droid weight app from the play store.
4. Open the Droid weight app and register your name and weight.
5. Register new weight, save it, register other weight and save it.
6. Try to register different weights.
7. Click to calculate BMI.
8. Add your weight, height and then click calculate to show the result.
9. Click to report at the Droid weight app.
10. End Test.
11. Save result

Questionnaire

The Questionnaire's purpose is to collect feedback on the frameworks (Perfecto and Xamarin) in order to evaluate each one.

Choose only number:

Table 20 Scaling Rate

1	2	3	4	5
Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree

Table 21 Questionnaire

Question	1	2	3	4	5
<i>“The information (such as on-line help, on-screen messages and other documentation) provided with this framework was clear.”</i>					
<i>“I’m satisfied with how easy it is to test the mobile app using this framework.”</i>					
<i>“I could effectively complete the tasks and scenarios using this framework.”</i>					
<i>“Whenever I made a mistake using the system, I could recover easily and quickly.”</i>					
<i>“I need programming knowledge”</i>					
<i>“I found the interface of framework user friendly.”</i>					
<i>“The documneation provided for the framework was easy to understand.”</i>					
<i>“After I finished test, I found all the information on the report.”</i>					
<i>“I felt comfortable using this framework.”</i>					

“It was easy to learn to use this framework.”

Perfecto and Xamarin Devices

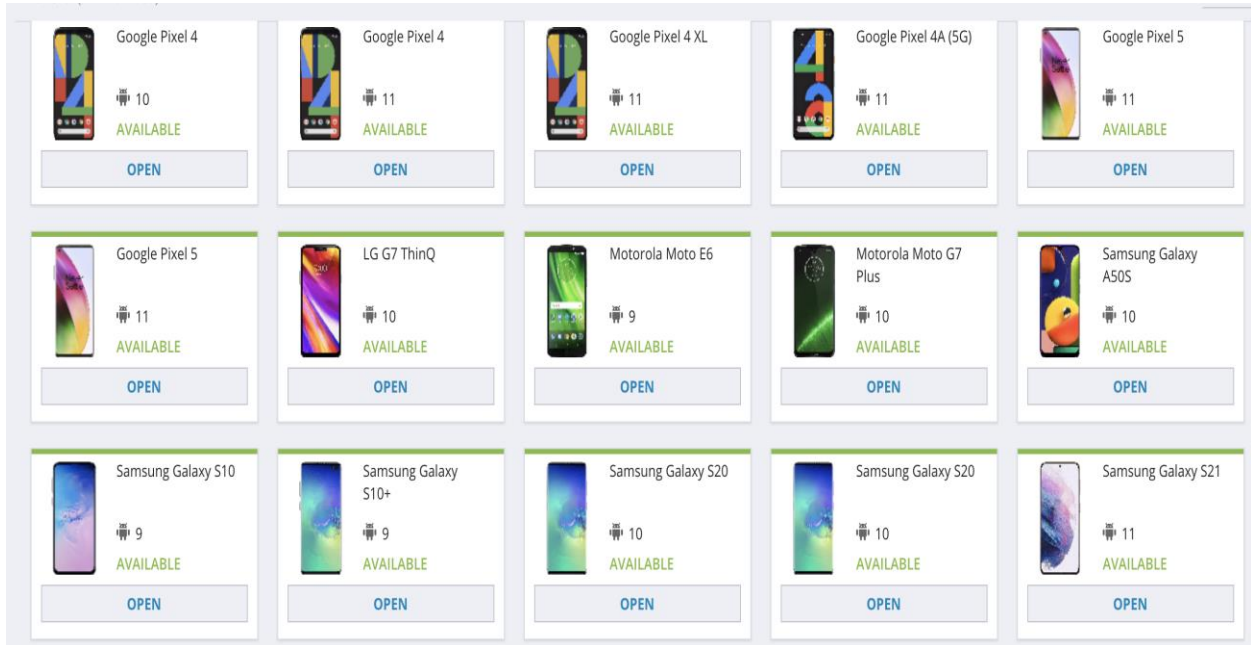


Figure 32 Perfecto Devices








<input checked="" type="checkbox"/>		Samsung Samsung Galaxy S10e	Android 11
<input type="checkbox"/>		Samsung Samsung Galaxy S20 FE	Android 11
<input type="checkbox"/>		Samsung Samsung Galaxy S10 Lite	Android 11
<input type="checkbox"/>		Samsung Samsung Galaxy XCover Pro	Android 11
<input type="checkbox"/>		Samsung Samsung Galaxy A70	Android 11
<input type="checkbox"/>		Samsung Samsung Galaxy A51	Android 11
<input type="checkbox"/>		Samsung Samsung Galaxy A50	Android 11

Figure 33 Xamarin Devices

Code example for Perfecto and Xamarin Framework

```

public void mobileTest() throws Exception {
    //Replace <<cloud name>> with your perfecto cloud name (e.g. testingcloud ) or pass it as maven properties: -Dcl
    String cloudName = "testingcloud";
    //Replace <<security token>> with your perfecto security token or pass it as maven properties: -DsecurityToken=<
    String securityToken = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpzZW50L3NpdCI6ImF6dUUiwi2lkIiA6ICI2ZDM2NmJiNS01NDYyLTQ4MmMtYTZhOC1kODZh

    //Mobile: Auto generate capabilities for device selection: https://developers.perfectomobile.com/display/PD/Sele
    String browserName = "mobileOS";
    DesiredCapabilities capabilities = new DesiredCapabilities("", "", Platform.ANY);
    capabilities.setCapability("platformName", "Android");
    capabilities.setCapability("platformVersion", "10");
    capabilities.setCapability("location", "NA-US-BOS");
    capabilities.setCapability("resolution", "720x1600");
    capabilities.setCapability("manufacturer", "Samsung");
    capabilities.setCapability("model", "Galaxy A21");

    try{
        driver = new RemoteWebDriver(new URL("https://" + "192.168.1.4:445/wd/hub"), capabilities);
        driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);
        driver.manage().timeouts().pageLoadTimeout(15, TimeUnit.SECONDS);
    }catch(SessionNotCreatedException e){
        throw new RuntimeException("Driver not created with capabilities: " + capabilities.toString());
    }

    reportiumClient = Utils.setReportiumClient(driver, reportiumClient); //Creates reportiumClient
    reportiumClient.testStart("Perfecto mobile web test", new TestContext("tag2", "tag3")); //Starts the reportium t
    reportiumClient.stepStart("browser navigate to perfecto"); //Starts a reportium step

```

Figure 34 Perfecto Code Example

```

public class AppInitializer
{
    public static IApp StartApp(Platform platform)
    {
        // TODO: If the iOS or Android app being tested is included in the solution
        // then open the Unit Tests window, right click Test Apps, select Add App Project
        // and select the app projects that should be tested.
        //
        // The iOS project should have the Xamarin.TestCloud.Agent NuGet package
        // installed. To start the Test Cloud Agent the following code should be
        // added to the FinishedLaunching method of the AppDelegate:
        //
        // #if ENABLE_TEST_CLOUD
        //     Xamarin.Calabash.Start();
        // #endif
        if (platform == Platform.Android)
        {
            return ConfigureApp
                .Android
                // TODO: Update this path to point to your Android app and uncomment the
                // code if the app is not included in the solution.
                .ApkFile ("droid")

                .StartApp();
        }

        return ConfigureApp
            .iOS
            // TODO: Update this path to point to your iOS app and uncomment the
            // code if the app is not included in the solution.
            //.AppBundle ("../../../../../iOS/bin/iPhoneSimulator/Debug/XamarinForms.iOS.app")
            .StartApp();
    }
}

```

Figure 35 Xamarin Code Example

```

public void myTest() {
    reportiumClient.testStart("myTest",
        new TestContext.Builder()
            .withTestExecutionTags("Ruba", "Esawi")
            .withCustomFields(new CustomField("perfecto
.vcs.filePath",
                "Java/main-
sample/src/main/java/com/perfectomobile/appTest/MyTest.java"))
            .build());
    ...
}

```

Figure 36 My test Perfecto Framework

Additional Details

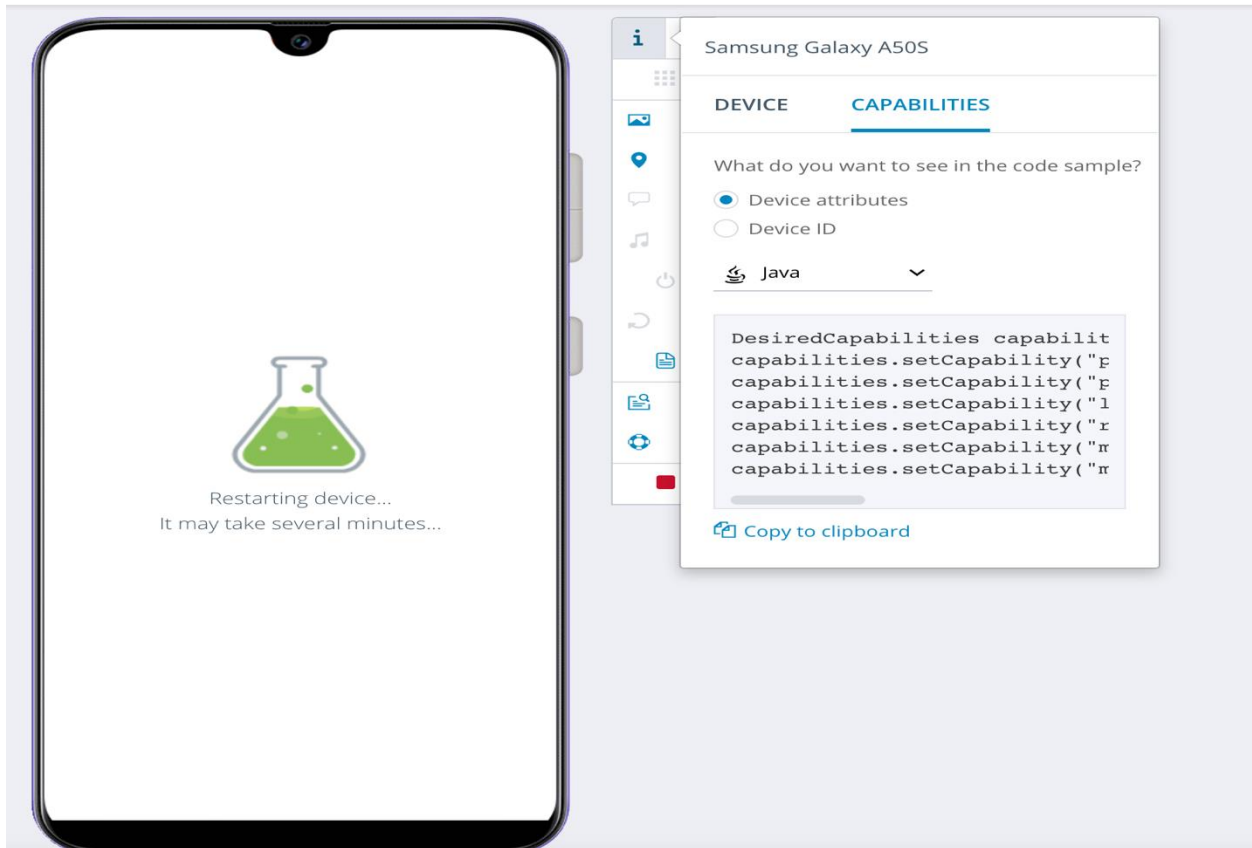


Figure 37 Device Capabilities (Perfecto Framework)

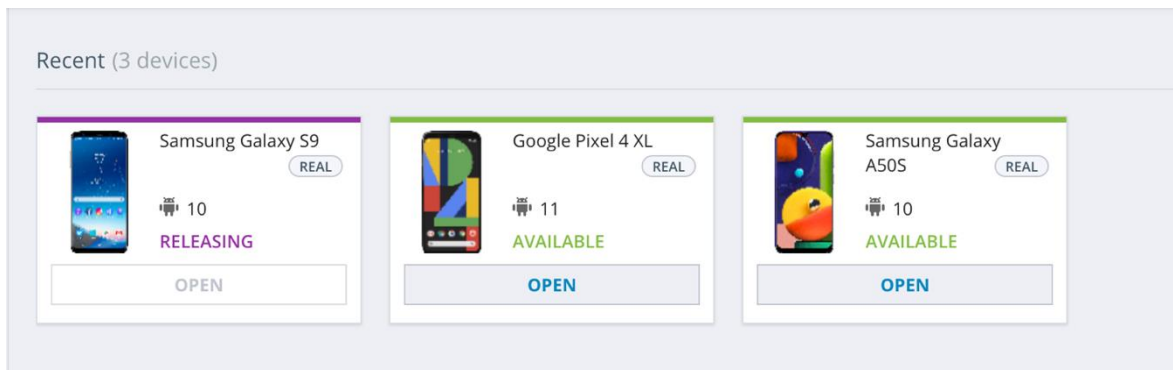


Figure 38 My Device (Perfecto Framework)