A SYSTEMATIC COMPARISON BETWEEN FLUTTER AND REACT NATIVE FROM AUTOMATION TESTING PERSPECTIVE

*

1st Husam Abu Zahra
Department of Software Engineering and Computer Science
Birzeit University
Birzeit, Palestine
habuzahra@gmail.com

2nd Samer Zein
Department of Software Engineering and Computer Science
Birzeit University
Birzeit, Palestine
szain@birzeit.edu

2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT) | 978-1-6654-7013-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISMSIT56059.2022.9932749

***Abstract*—In recent years, mobile applications have attempted to significantly improve our lives. Android and iOS are the two most popular mobile platforms, yet they have varied settings (OS versions, screens, manufacturers, etc.) as a result creating a mobile application might be difficult. In recent years, several cross-platform frameworks, such as Flutter and React Native, have shown to be pioneers in this field. For developers, however, choosing cross-platform can be an intimidating task. In this research, we examine the automated testing capabilities of the Flutter and React Native frameworks in terms of automation testing. We concentrate on automation testing reusability, integration, and compatibility. We created a To-Do List mobile app in Flutter and React-Native by using Testproject.io as an automation testing platform. As a result of this research, the experiment demonstrates that React Native outperforms Flutter in terms of reusability and compatibility, with no significant difference in terms of integration.**

***Index Terms*—Cross Platform, Flutter, React Native, Testing, Automation Testing**

I. Introduction

As the need for electronic services rises, mobile applications (apps) have a tremendous impact on our lifestyles, habits, and many other aspects of our existence, the need for mobile applications is no longer a luxury, it has become an essential part of our daily lives [1]. According to current research, the mobile sector has grown dramatically in recent years, with estimates estimating that 5 billion people globally own a mobile device, more than half of which are smartphones.

With the increased demand in this business, firms and manufacturers are scrambling to create devices that target all client groups, including different screen sizes, CPUs, sensors, and a variety of other capabilities, placing pressure on developers to create high-quality apps.

Companies used to engage different teams to handle the development of native mobile applications when mobile development first began, to develop for the iOS and Android platforms, two or more dedicated teams were necessary, this increased the overall cost of the development process, which comprised development, testing, deployment, and the expense of running numerous teams.
[1]

With the introduction of cross-platform frameworks such as Flutter, React Native, and Xamarin by significant tech firms such as Google, Microsoft, and Facebook, developers and businesses began creating mobile apps that operate on both iOS and Android using the same versions of the source code, this helps the developers and software companies to reduce and mitigate the work and cost of the development process, statistics in "Fig. 1" show that React Native and Flutter hold the largest market share in comparison to other frameworks as they are developed and supported by Facebook and Google [2], [3].

II. Related Works

As part of our investigation, we discovered no studies comparing Flutter and React Native in terms of testing, as well as no studies comparing Flutter and React Native in general, therefore, various studies on the cross-platform issues have been found, as well as research on a testing strategy for the same app on multiple cross-platforms such as iOS and Android and studies on the testing process for mobile apps, several studies were identified.

The following criteria are used to choose the papers: 1) Papers written in English 2) Papers released in 2016 and after 3) Cross-platform research 4) documents longer than four pages in all, as a result, 22 articles out of 43 were reviewed for this study, and three groups were identified: 1) Cross-Platform challenges 2) Cross-Platform applications testing 3) Mobile applications cross platform evaluation.

A. Cross Platform Challenges

Six studies have been reviewed in this section to investigate the cross-platform challenges, accessing the device hardware

978-1-6654-7013-1/22/$31.00 ©2022 IEEE

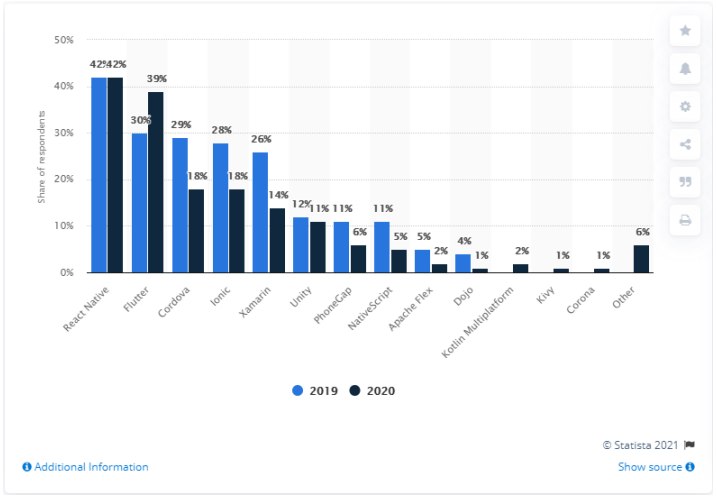

Fig. 1. Cross Platform Comparison

and modifying the cross-platform libraries were the results of [4], on the other the sub optimal experience of UI and UX was stated by [5], while [6] indicates that battery consumption is an issue in cross-platform mobile apps. Reference [7] shed the light on the backward compatibility issues in the cross-platform apps, while [4]–[6] agreed that performance is a major issue in cross-platform applications.

Reference [4] conducted a case study on the Cross-Platform and did qualitative research to investigate the challenges of the Cross-Platform, they found during their research that the developers sometimes must modify the cross-platform libraries to support some features which are not supported by the standard cross-platform itself, on other hand, some of the challenges were that the cross-platform frameworks are consuming the device performance, and developers need to rebuild their applications again using native language to avoid the performance issue on the device, they found that when the nature of the mobile application requires deep access to the hardware of the device, the developers select the native approach and avoid using the cross-platform, this is due to the simplicity of accessing the device hardware using the native language compared to the cross-platform.

Reference [8] has been argued that the Cross-Platform provide less UI and UX experience compared to the native platform, their results were built by the decision-makers may not question the truth, and they rely on the statements that being published as peer-reviewed scientific papers, Hansen,Grønli and Ghinea mentioned that breaking this circle of misconceptions requires research efforts to test and validate this claim, while [7] identified the challenges of cross-platform mobile development by the large number of the different devices manufacturers, where each manufactured device has its different version of the OS and different security rules, which make the process for developing applications using the cross mobile platform is not easy when trying to access the hardware, matching the results which have been discussed by [4], [5].

Reference [7] also experimented the Xamarin and identified that cross-platform applications are not backward compatible when the new version of the Mobile SDK is being updated, and the developers need to modify their Xamarin code once the Android or the iOS SDK is updated.

Reference [9] conducted a study on two data sets counting 85,908 questions, those questions are posted on Stack Overflow and Xamarin Forum and downloaded using Stack Exchange, their findings were matching with what has been discussed by [8], they conclude that the cross-platform still has an immature community and the official documentation of Xamarin still need to be improved.

B. Cross Platform applications Testing

Eight studies have been reviewed to investigate cross platform testing, [10], [11] both of them built a tool to discover the limitation and the challenges of the cross platform under certain circumstances, such as generating test scripts that can run on iOS and Android, having tools that can perform the test on the deployed app, while [12] perform an automated test to measure the UX of the cross-platform apps, [13] developed a framework for mobile application automation testing on the Android platform only, while [14] introduce the automation for testing the accessibility on the mobile applications.

Reference [15], [16] surveyed to investigate the state of the art tools in this area, while [17] used the mutation testing to discover the remaining mutant over the different platform for the cross-platform applications.

Reference [10] did research using the design science paradigm described by [18] to discover the testing for the cross-platform, they built a framework called Mobilette for testing mobile cross-platform applications, the framework was supporting only touching elements, setting text and getting text, they did the test on a simple application that displaying greeting message based on the input text, the test has been executed over 2 different platform iOS and Android, where each platform has been tested using an emulator and the real device, in their conclusion, they mentioned that the tools can handle only simple UI, and in the future, they can extend it to handle complex and multi-screens UI.

Reference [18] developed a framework called x-PATeSCO to generate test scripts for the applications built using different cross-platform, the tool was developed based on Appium and the Selenium WebDriver API where the output is Appium script, the framework used to capture the events and parse the XML based on two combined strategies, Expressions-MultiLocator where it depends on the following attributes: XPath-attribute values, XPath-Identifiers and XPath-Absolute Path, while the ExpressionsInOrder depends on six attributes CrossPlatform, ElementType, IdentifyAttributes, AncestorAttribute, AbsolutePath, and AbsoluteIndex, the experiment has been done over 9 open-source applications built using three cross-platform Cordova, React Native, and Xamarine and tested over different six configurations, the results show that 4 strategies are applicable for all types of tests which are (ExpressionsInOrder, AncestorIndex, ExpressionsMul-tiLocator,

and AbsolutePath) while (ElementType, AncestorAttributes IdentifyAttributes, and CrossPlatform) are less applicable.

Reference [12] in their study performed an automation test to measure the UX performance between the native app and the cross-platform applications, they developed a simple application in Flutter, React and the Native Android application, the app is a social media-like app, which is used to post comments and does scroll for the profile timeline, the application stores the data on the device and is not connected to the network, they used detox, Flutter driver and espresso to automate the testing process, even [12] were able to measure the UI response, but in their conclusion, they mentioned that the used tools were not fully support the automation testing, where each tool has different approaches, and they faced a lot of difficulties to simulate the scrolling and opening the camera, they mentioned the needs for studies that investigate the cross-platform automation for the different platform as well.

Reference [13] developed a general framework to cover the automation testing techniques for the mobile apps but didn't shed the light on any area related to cross-platform, while [14] introduce the automation for testing accessibility on the mobile applications same as [13], they didn't touch the cross-platform in specific but settled to cover only apps for the Android platform, on the other hand [15] conducted a survey for the current state of the art tools, frameworks, and services available for the applications developers to do automation test, they performed the survey on ten Automation's frameworks, their findings were that ten platforms were able to do the GUI automation's, while two of them were able to perform the same test on cross-platform apps, four of the frameworks were able to perform the test but with limited capabilities and the remain three were not able to perform the tests.

Reference [15] in their conclusion second on what has been concluded by [12] that the frameworks have limitations when it's dealing with the apps developed using cross-platform, where the framework will be supporting single object like the scrolling, pinching, and zooming, while [17] in their research proposed mutation testing to perform the test for cross-platform applications, they developed an open-source application called MuHyb, in their research, they test the framework on an application used for food delivery, the application built using Ionic framework, the testing tools managed to generate 149 mutants, 84 mutants killed, 22 discarded and 43 using survived, with total mutations score of 66.14%.

Ahmed, Taj-Eddin, and Ismail in [17] mentioned that MuHyb was the first tool in the market that performed mutation testing for cross-platform applications, in their conclusion they mentioned that MuHyb supports only Ionic as a limitation to their research, and it will be developed to support extra cross platforms in the future.

Kong et al. [16] conducted a systematic literature review to shed the light on the main trends, methodologies, and pain points for testing android apps, they listed seven tools used to do automation testing for the Android applications, thier results didn't mention if the tools were used to test cross-platform, or if the applications used were built using cross-platform and running on Android apps.

C. Mobile Applications Cross Platform Evaluation

In this category, eight studies have been reviewed to investigate how the cross-platform is evaluated, studies [19], [20] developed a framework to enable the researchers, developers, and others to evaluate the cross-platform, while [21] conduct a comprehensive comparison between three cross-platform to help the developers choose one of them, on the other hand, [22] evaluate the cross-platform against the native platforms, [23] conducted research to evaluate the effect of using the cross platforms on the UX, [20] proposed a framework that helps both researchers and practitioners to evaluate a cross-platform, the framework is based on weighted holistic evaluation criteria which were developed with help of the domain experts judge and critical review for 30 papers about the cross-platform, by taking into consideration future evaluation of the technology stack, this proposed framework is an extension to a prior work [20] that also attempts to provide an evaluation criterion based on a predefined catalogue.

Reference [20] identified 34 criteria classified into four major categories two evaluate the cross platform frameworks, application perspective, usage perspective, development perspective and infrastructure perspective, while [19] developed a framework to evaluate the cross- platform, but their comparison was based on the below measures, User Experience, Easiness of Development, and appearance, they finally mentioned in their conclusion that the cross-platform still have weakness on the UX side, while [21] performed a comprehensive comparison analysis for three cross-platform, React Native, Fuse, and Ionic, the research aimed to provide the developers with advice when choosing between the three platforms, they performed an online survey with more than 100 receiving responses, in addition to that, survey was supported by building a prototype model to assets the survey.

Reference [21] comparison shows that User experience, technical implementation, performance, and item testability are the most common issues when developing cross-platform applications, Majchrzak, Biørn-Hansen, and Grønli [21] in their conclusion mentioned that there is no specific winner between the selected frameworks, and a few problems need to be investigated such as the human side particularity in the form of the end-user experience.

Reference [22] investigated the difference between the native and the cross platforms, their goal was to evaluate the cross-platform for non-experienced and students' developers, they provide a four-month course that included the simple features in both the native and the cross-platform, which was then followed by a questionnaire shared with 169 applicants including 30 experienced developers, Meirelles, Aguiar, Assis, Siqueira, and Goldma [22] in their results mentioned that students and experienced developers preferred native development when need to achieve high performance, while if the performance and the device access is not an essential part of the application, the cross-application can be considered as a choice for the developers matching the findings achieved

by others in [4]–[6], on the other hand [24] developed a framework to evaluate the cross platforms based on the energy consumption, considering that it's one of the most important factors when selecting cross platforms.

Approach	Quality Concern	Android		iOS	
		Avg.	Std. Dev.	Avg.	Std. Dev.
Native (3)	Performance	0.645%	0.496%	1.473%	1.291%
	Reliability	3.458%	2.691%	18.325%	18.013%
	Usability	0.106%	0.071%	0.745%	0.909%
	Security	0.165%	0.206%	0.091%	0.277%
Interpreted/Generated (35)	Performance	0.786%	1.056%	3.261%	7.760%
	Reliability	7.736%	8.900%	9.448%	10.434%
	Usability	0.132%	0.208%	0.883%	2.092%
	Security	0.309%	0.751%	0.040%	0.177%
Hybrid (17)	Performance	1.304%	1.296%	3.785%	11.377%
	Reliability	18.294%	22.194%	12.012%	13.111%
	Usability	0.389%	0.574%	0.605%	0.854%
	Security	0.201%	0.470%	0.024%	0.058%

Fig. 2. Users Review results [23]

Reference [24] conclude the same that has been discovered by others in [4]–[6], [22] , where the cross platforms energy consumption is higher than native platforms and considering selecting the cross-platform could be a bad choice when the performance is a key factor for the application, and the native languages could act better than the cross-platform, while [23] perform research to investigate the effect of the cross-platform on the user experience, they used the Natural programming languages (NLP) to collect the reviews posted by the users on Google Play and Apple Store, Mercado, Munaiah, and Meneely collected 787,228 related to 50 apps to investigate the relations between the User Experience and cross-platform, they classified all collected comments into four metrics, performance, usability, security, and reliability where also they used machine learning to train the model to perform such clarifications, the results of their investigation have been concluded in "Fig. 2".

METHODOLOGY

The section's goal is to present our research question as well as the research methodologies we used, to achieve exact findings for this study, we used a methodical approach to working with and comparing the two frameworks from an automated testing standpoint, we looked for identical application created in Flutter and React Native in open-source code repositories (GitHub and SourceForge) so we could conduct our experiment, there were no apps that met our criterion, and some others required payment to obtain the source code, so we created an identical application on Flutter and React Native, as the same coding approaches will be utilized for both frameworks, we will gain superior outcomes, all created artifices and source code have been uploaded and contributed to GitHub to ensure verification and to allow others to use the source code in the future [2], the following research questions have been developed to draw the path of our research:

- RQ1: What are the differences between Flutter and React Native in term of testing automation record and play?
- RQ2: What are the differences between Flutter and React Native in term of test automation Integration from external SDK?
- RQ3: What are the differences between Flutter and React Native in term of test automation compatibility when running on different Android devices?

We later developed ToDo List apps in Flutter and React Native for the following reasons:

- Developing a ToDo List application is achievable within the limits of this research.
- The nature of these applications are straightforward to implement.

The goal of this application is to assess how automation frameworks handle new items that need to be added to the UI, delete objects from the UI, test scrolling functionality, and test gesture actions, we performed the test on Android OS. only for the following reasons:

- Android is an open-source operating system that enables unlimited study.
- Android devices have a bigger market share than iOS devices.
- The availability of Android emulators and the ability to link it with different IDE's compared to iOS.
- Running on iOS requires the use of Mac computers for compilation, which is not available.

We utilized TestProject.io Platform as a service to conduct our experiment, TestProject.io simply require an internet connection and a web browser to use, the selection of this tool was based on various benefits that it has over other testing tools, including but not limited to:

- Detection of elements using Xpath, element ID, and position.
- Playback and recording.
- Script generation in several languages, including Java.
- Reporting tools for completed test cases.
- It is free and does not require any licensing.

We evaluated and compared the two cross-platforms, Flutter and React Native, and established the evaluation criteria that need to be examined systematically based on our literature analysis, we determined three parameters to be assessed, which are the follwoing: 1) Re-Usability 2) Integration 3) Compatibility, to measure each parameter in this study, four criteria have been developed, Table 2 presents the criteria used to utilize and evaluate our parameters.

Criteria C1 identify the ability of the automation testing tools to detect the UI elements on each cross-platform inspired by [25], while criteria C2 identifies the ability of the teasing tools to generate scripts that can be used to run the same test for the same artifact on different cross platforms inspired

[2]https://github.com/habuzahra/

TABLE I
SELECTED CRITERIA FOR EVALUATION OUR PARAMETERS

ID	Criteria Name	Category
C1	Element Detection	Re-usability
C2	Scripts Generation	Integration
C3	GUI Testing	Re-usability
C4	Running on different devices	Compatibility

by [26], criteria C3 identifies the ability to run the test on the generated artifacts on different cross platforms such as scrolling and navigation inspired by [27], criteria C4 identify the ability of the automation testing tools to run on the same tests on other devices inspired by [25].

To meet the aims of this research and compare Flutter and the React Native framework from the standpoint of Automation testing, the two frameworks will be compared, and the criteria that have been chosen will be evaluated, we chose data collection methodology for each of the selected criteria so that we can build our measurements and quantify the results of the comparison between the selected frameworks, which will help to answer the research questions, for the sake of this research test cases have been written as per Table 6 to be able to execute our experiment on it.

As stated at the outset of our research, the assessment criteria were divided into three categories:

- The re-usability criterion will describe how the UI components can be identified using the automation tools, finding the elements will allow the reuse and extension of the scripts instead of recording again, Table 3 describes the criteria for measuring this characteristic.

TABLE II
SELECTED CRITERIA DATA COLLECTION

ID	Criteria Name	Data Collection
C1	Record and play	Identify the UI elements
C2	Scripts Generation	Are both artifacts accept integration.
C3	GUI Testing	Are both artifacts can support GUI test
C4	Running on different devices	Can the recorded scripts run on different devices

TABLE III
RE-USABILITY CATEGORY

Criteria Name	Description
Element Detection	This criterion will tell if the UI elements of the generated artifacts can be detected, this will include the detection of the text box, Popup message, views, Buttons, Labels, etc.

- The compatibility criteria will explain how the testing for the generated artifacts will work for each cross-platform and examine if the generated scripts of the automation tool will work on different Android devices, Table 4 explains the criteria that will be used to measure this characteristic.

TABLE IV
COMPATIBILITY CATEGORY

Criteria Name	Description
Different devices	This criterion will tell if the automation scripts will be work successfully on different android devices from different manufacturers, we used the following three different devices: – Samsung Note 20, – Oppo F11 – Xiaomi Redmi Note 9

- The Ability to integrate criteria will explain if the automation testing tools will be able to integrate with the generated artifacts from each framework, where we will examine the possibility to integrate and pass data from external sources to be able to perform the test and integrate using external SDK, Table 5 explains the criteria that will be used to measure this characteristic.

TABLE V
ABILITY TO INTEGRATE CATEGORY

Criteria Name	Description
Integration	This criterion will tell if the automation testing tools will be able to integrate with the generated artifacts from each cross platform using external SDK, this will help to write unit test cases by the developers and extend the testing abilities.

RESULTS AND DISCUSSION

This section discusses the results that were acquired during the run of the test scripts over the two cross platforms, the total amount of executed test cases were 90 test cases distributed over the 3 identified categories, Re-usability, Integration, and compatibility and performed over 3 devices, Samsung Note 20 , Xiamomi Redmi Note 9, and Oppo F11, Judging by the results, we could conclude that for the Re-usability and compatibility, React Native was performing better in term of the success rate of the executed test cases, we measured that for the Re-usability, the success rate of the executed test cases for React Native was 88.89% while for Flutter it was 66.67% as shown in “Fig. 3”

RQ1: What are the differences between Flutter and React Native in term of testing automation Record and Play? To answer this research question, we compared both Flutter and React Native based on the generated results, as demonstrated, there is a significant difference between results on each artifact, this is due to the fact that the automation tool was not able to detect the UI elements correctly in the generated artifacts from Flutter, while for the artifact that is generated

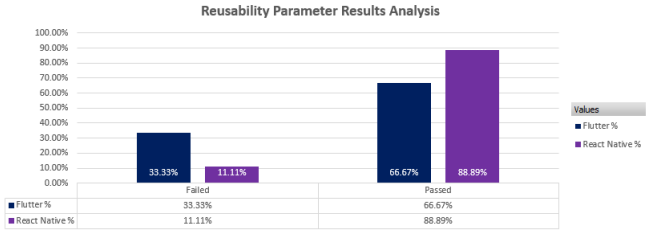

Fig. 3. Re-usability Parameter results analysis

from React Native, the tool was able to detect all the of the UI elements.

For the integration it has been measured that both frameworks showed the same capability and support with 100% success rate for the executed tests as shown in "Fig. 4".

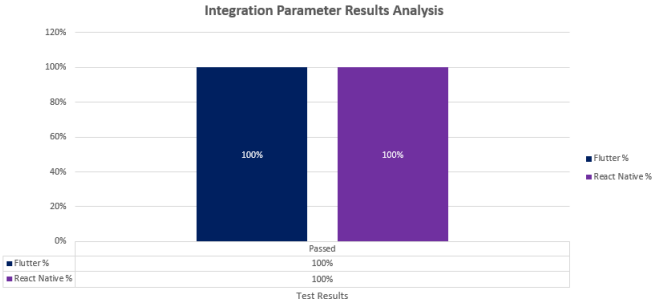

Fig. 4. Integration Parameter results analysis

RQ2: What are the differences between Flutter and React Native in term of test automation Integration from external SDK? To answer this research question, we compared both Flutter and React Native based on the generated results, as shown in Figure 4, we found that there is no significant differences between the results on each artifact.

On the other hand, the success rate for the compatibility for React Native was 88.9% on Samsung and Xiamoi, while on Oppo the success rate was 77.78%, while on Flutter the success rate of the executed test cases was 66.67% on all devices.

RQ3: What are the differences between Flutter and React Native in term of test automation compatibility when running on different android devices? To answer this research question, we compared both Flutter and React Native and executed the test cases in Table 6 on three different devices as shown in "Fig. 5", based on the generated results, there is significant differences between the results on each artifact on each different device, this due to the fact that the screen size of Opp F11 is different where it's 6.5 while the others are 6.7, hence extra step need to be added to handle the different screen size.

The hypothesis prior running the tests was that both Flutter and React Native would perform significantly the same, that guess was because the automation platform will not be able to distinguish between the final generated artifacts (APK) file from each platform, the results that have been acquired

TABLE VI
TEST CASES

Test case Name	Expected results
In the login screen enter valid email address and password	The application should route you to the Today Tasks Screen.
In the login screen enter invalid email address and password	The application should display error message popup.
On the login screen press on the sign-up link	The application should route you to the sign-up page.
In my task screen add new task and press add to List	The task should be added.
In my task screen try delete the added task	The task should be added.
Try to add empty task	The task should be deleted.
Try to add empty task	The application should display to you error message popup.
Add 10 tasks and scroll down	The application screen should allow you to scroll down and view all the added tasks.

disproved that, even our application is small application, but the results showed that the final output of each artifact has led to different results, this is due to the fact that the underneath compilation and the way the generated artifact is being produced is totally different in each framework, and may have significant effect on the testing phase when the applications are large and have complex features such as, location detection, device sensors, body motions and so on.

CONCLUSION

In this study, React Native and Flutter were systematically compared from the perspective of automation testing. It was discovered that React Native outperformed Flutter in terms of reusability and compatibility, with no discernible difference in terms of integration.

However, this study pursued a different strategy to contrast React Native with Flutter from the perspective of automation testing. Prior research mainly examined the effectiveness and performance of cross-platform frameworks. There are a few warnings made to the current study. First, we removed iOS from our testing and solely used an Android device, which may have revealed other variances. Similarly, we chose just one platform for automation testing. Even if the majority of automation tools offer comparable features, more automation tools ought to be included. If work is to take place in the future, iOS, additional automation testing frameworks, and other types of devices might be included.

REFERENCES

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

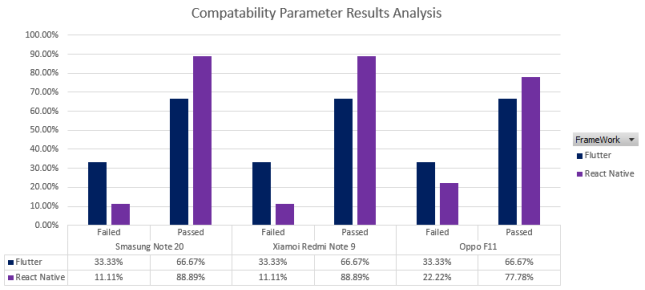

Fig. 5. Compatibility Parameter results analysis

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

REFERENCES

[1] K. Mehmood, M. K. Ashraf, and M. Nouman, "Mobile UX design".

[2] A. E. Fentaw and Cross, "Cross platform mobile application development: a comparison study of React Native Vs Flutter," p. 98, 2020.

[3] S. Stender and H. Åkesson, "Cross-platform Framework Comparison: Flutter React Native," vol. 6, 2020.

[4] T. Zohud and S. Zein, "Cross-Platform Mobile App Development in Industry: A Multiple Case-Study," Int. J. Comput., vol. 20, no. 1, pp. 46–54, 2021.

[5] A. Biørn-Hansen, T. M. Grønli, G. Ghinea, and S. Alouneh, "An Empirical Study of Cross-Platform Mobile Development in Industry," Wirel. Commun. Mob. Comput., vol. 2019, 2019.

[6] L. Delia et al., "Development approaches for mobile applications: Comparative analysis of features," Adv. Intell. Syst. Comput., vol. 857, pp. 470–484, 2019.

[7] M. Huynh and P. Ghimire, "Browser app approach: Can it be an answer to the challenges in cross-platform app development?," J. Inf. Technol. Educ. Innov. Pract., vol. 16, no. 1, pp. 47–68, 2017.

[8] A. Biørn-Hansen, T. M. Grønli, and G. Ghinea, "A survey and taxonomy of core concepts and research challenges in cross-platform mobile development," ACM Comput. Surv., vol. 51, no. 5, 2019.

[9] M. Martinez, "Two datasets of questions and answers for studying the development of cross-platform mobile applications using Xamarin framework," Proc. - 2019 IEEE/ACM 6th Int. Conf. Mob. Softw. Eng. Syst. MOBILESoft 2019, pp. 162–173, 2019.

[10] T. M. Gronli and G. Ghinea, "Meeting quality standards for mobile application development in businesses: A framework for cross-platform testing," Proc. Annu. Hawaii Int. Conf. Syst. Sci., vol. 2016-March, pp. 5711–5720, 2016, doi: 10.1109/HICSS.2016.706.

[11] A. A. Menegassi and A. T. Endo, "Automated tests for cross-platform mobile apps in multiple configurations," IET Softw., vol. 14, no. 1, pp. 27–38, 2020.

[12] M. Mahendra and B. Anggorojati, "Evaluating the performance of Android based Cross-Platform App Development Frameworks," ACM Int. Conf. Proceeding Ser., pp. 32–37, 2020.

[13] D. Amalfitano, N. Amatucci, A. M. Memon, P. Tramontana, and A. R. Fasolino, "A general framework for comparing automatic testing techniques of Android mobile apps," J. Syst. Softw., vol. 125, pp. 322–343, 2017.

[14] M. M. Eler, J. M. Rojas, Y. Ge, and G. Fraser, "Automated Accessibility Testing of Mobile Apps," Proc. - 2018 IEEE 11th Int. Conf. Softw. Testing, Verif. Validation, ICST 2018, no. Figure 1, pp. 116–126, 2018.

[15] M. Linares-Vásquez, K. Moran, and D. Poshyvanyk, "Continuous, evolutionary and large-scale: A new perspective for automated mobile app testing," Proc. - 2017 IEEE Int. Conf. Softw. Maint. Evol. ICSME 2017, pp. 399–410, 2017.

[16] P. Kong, L. Li, J. Gao, K. Liu, T. F. Bissyandé, and J. Klein, "Automated testing of Android apps: A systematic literature review," IEEE Trans. Reliab., vol. 68, no. 1, pp. 45–66, 2019.

[17] S. Ahmed, I. A. T. F. Taj-Eddin, and M. A. Ismail, "MuHyb: A Proposed Mutation Testing Tool for Hybrid Mobile Applications," ACM Int. Conf. Proceeding Ser., pp. 67–72, 2020.

[18] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Essay in Information Design Science systems," Manag. Inf. Syst., vol. 28, no. 1, pp. 75–105, 2004.

[19] V. Ahti, S. Hyrynsalmi, and O. Nevalainen, "An evaluation framework for cross-platform mobile app development tools: A case analysis of adobe PhoneGap framework," ACM Int. Conf. Proceeding Ser., vol. 1164, no. June, pp. 41–48, 2016.

[20] C. Rieger and T. A. Majchrzak, "Towards the definitive evaluation framework for cross-platform app development approaches," J. Syst. Softw., vol. 153, pp. 175–199, 2019.

[21] T. A. Majchrzak, A. Biørn-Hansen, and T. M. Grønli, "Comprehensive analysis of innovative cross-platform app development frameworks," Proc. Annu. Hawaii Int. Conf. Syst. Sci., vol. 2017-Janua, pp. 6162–6171, 2017.

[22] P. Meirelles, C. S. R. Aguiar, F. Assis, R. Siqueira, and A. Goldman, A Students' Perspective of Native and Cross-Platform Approaches for Mobile Application Development, vol. 11623 LNCS. Springer International Publishing, 2019.

[23] I. T. Mercado, N. Munaiah, and A. Meneely, "The impact of cross-platform development approaches for mobile applications from the user's perspective," WAMA 2016 - Proc. Int. Work. App Mark. Anal. co-located with FSE 2016, pp. 43–49, 2016.

[24] M. Ciman and O. Gaggi, "An empirical analysis of energy consumption of cross-platform frameworks for mobile development," Pervasive Mob. Comput., vol. 39, pp. 214–230, 2017.

[25] P. Raulamo-Jurvanen, M. Mäntylä, and V. Garousi, "Choosing the right test automation tool: A grey literature review of practitioner sources," ACM Int. Conf. Proceeding Ser., vol. Part F1286, pp. 21–30, 2017.

[26] H. V. Gamido and M. V. Gamido, "Comparative review of the features of automated software testing tools," Int. J. Electr. Comput. Eng., vol. 9, no. 5, pp. 4473–4478, 2019.

[27] E. Börjesson and R. Feldt, "Automated system testing using visual GUI testing tools: A comparative study in industry," Proc. - IEEE 5th Int. Conf. Softw. Testing, Verif. Validation, ICST 2012, pp. 350–359, 2012.