A Comparative Study of Cross-platform Mobile Application Development

Conference Paper · July 2021

CITATIONS
3

READS
1,383

2 authors, including:

Minjie Hu
Wellington Institute of Technology
18 PUBLICATIONS **367** CITATIONS

SEE PROFILE

A Comparative Study of Cross-platform Mobile Application Development

Dongliang You, Minjie Hu
Whitireia Polytechnic, New Zealand
dongliang.you01@whitireianz.ac.nz

Abstract

Mobile applications development has increasingly become crucial as the number of mobile phone users has grown exponentially. However, it is even more difficult for developers to build applications that are efficient and effective than ever before, because of multitude of functions that are needed to be implemented in both Android and iOS at the same time. This study explored and discussed the approaches and applications in cross-platform application development through an experimental methodology. Therefore, a sample project was implemented with the native framework and then with three cross-platform frameworks, such as React Native, Flutter, and Xamarin. The data from each project was collected and analysed in terms of functionality, workload, development procedure, and performance in order to find their advantages and disadvantages. Finally, the study compared these three cross-platforms and also compared them to the native framework, and then draw a conclusion of which cross-platform framework is the best for mobile application development.

Keywords

cross-platform mobile development, android native, react native, flutter, xamarin

1. Introduction

Mobile applications development has increasingly become crucial as the number of mobile phone users has grown exponentially. However, it is even more difficult for developers to build applications that are efficient and effective than ever before, because of multitude of functions that are needed to be implemented in both Android and iOS at the same time. Therefore, cross-platform development using a native developing approach was proposed in order to be more efficient and productive for developers (Xanthopoulos & Xinogalos, 2013).

A recent study showed that although there were many existing cross-platform frameworks or libraries in mobile development, few of them were accepted widely by developers, such as React Native, Ionic, and PhoneGap (Biørn-Hansen et al., 2019). In 2020, Flutter and Xamarin became very popular frameworks.

The goal of the research is to answer the following three research questions: 1) how do these frameworks improve development efficiency compared with native development; 2) what are their limitations; and 3) which cross-platform framework is the best.

This quality assured paper appeared at the 12th Annual Conference of Computing and Information Technology Research and Education New Zealand (CITRENZ 2021) & 34th Annual Conference of the National Advisory Committee on Computing Qualifications, Wellington, July 14-16 i.c. with ITx 2021

2. Literature Review

Nowadays, smartphones have become increasingly important since the first release of the iPhone. There are two mobile systems that dominate the market, namely iOS and Android. They are built with different architectures, programming languages, and frameworks. Consequently, programmers must implement twice by using a platform-dependent programming approach, which revealed that cross-platform technologies are more efficient and productive for application programmers (Xanthopoulos & Xinogalos, 2013).

According to Biørn-Hansen et al. (2019), although many cross-platform technologies were emerging before 2015, such as Ionic, PhoneGap, and Titanium, none of them was used widely because of poor performance and inferior stability. The research also suggested that React Native, Flutter and Xamarin were becoming increasingly popular due to better performance and robustness.

2.1 React Native

React Native is a cross-platform framework developed by Facebook (Danielsson, 2016). The goal is to use only one programming language and framework to build mobile applications. It is a derivative of React, which is an open-source JavaScript (JS) framework from Facebook in the native mobile platform and supports iOS and Android platforms currently (Gill, 2018). Although it may need little effort of learning for those who are familiar with Web front-end development, React Native enables developers to write native mobile applications only using JavaScript programming language. It is consistent with React in terms of design principles and declarative component mechanism in building a rich User Interface (UI).

The advantages include: 1) it uses JavaScript to composite various components; 2) it transforms mark-up elements into native UI elements; 3) it separates the working thread from the main UI thread, which results in a better application performance with full functionality; and 4) it significantly saves time in terms of development and maintenance (Kravtsov, 2018). Conversely, the disadvantages include: 1)it does not support What You See Is What You Get (WYSIWYG) in UI Design (Gu et al., 2017); and 2). it requires platform-specific code to implement some functions (Cantù et al., 2018).

2.2. Flutter

Flutter is a free mobile application Software Development Kit (SDK) released by Google. The goal of Flutter is to deliver high-performance applications on Android and iOS platforms rapidly and smoothly. In a study by Cheon and Chavez (2020), Flutter, using neither WebView nor JavaScript, implements a UI framework by itself. It transfers UI components and renderers from platform to application, which makes them customizable and extensible. It only requires the system to provide a canvas so that customized UI components can appear on the device's screen, which is the key to be cross-platform and efficient. In addition, Flutter implements a state machine that is used widely in React Native to render the minimum changed area when updating the UI (Fayzullaev, 2018).

The merits include: 1) Dart is the only programming language used in Flutter (Idan Arb & Al-Majdi, 2020). And 2) it can be integrated with Android Studio through plugins, allowing Android developers to transit their work seamlessly (Dagne, 2019). The drawbacks include: 1) the plugins do not support WYSIWYG in UI design. And 2) it highly depends on components that were published in their community (Fentaw, 2020).

2.3. Xamarin

Xamarin is a cross-platform solution that aims to build iOS, Android, Mac, and Windows applications. In a study by Delia et al. (2015), the Xamarin application runs efficiently as a native one because it employs native controls and Application Programming Interfaces (API). Also, Xamarin creates a bridge to allow the platform-specific library can be called directly using C#, which is a mainstream programming language in the Windows platform. Furthermore, Xamarin also supports the .Net Standard library used in its projects, which expands the range of options during the developing period (Radi, 2016).

The advantages include: 1)it uses the C# programming language, which is quite similar to Java, so that the Android Native developers can rapidly study (Al-Bastami & Naser, 2017); 2)it is integrated with Visual Studio (Radi, 2016), which is the most powerful Integrated Development Environment (IDE) on Windows; 3)it can generate high-performance programs in an experience similar to native development (Willocx et al., 2016); and 4) it supports downloading components from the NuGet extension (Martinez, 2018), which accelerates the developing speed significantly. According to Avdic (2019), the disadvantages include: 1) it cannot support the latest native framework in time; and 2) it is not stable.

Although Flutter was considered better than React Native in UI performance (Jagiełło, 2019) and Xamarin was reflected as efficient as Android Native in terms of encryption processing speed (Dobrzański & Zabierowski, 2017), there is lack of study comparing these three cross-platform frameworks directly based on building a practical project. Furthermore, there is also lack of research comparing their characteristics beyond their performance, such as functionality, workload, and applicability, which is the knowledge gap that this paper is endeavouring to fill.

3. Research Methodology

3.1 Data Collection

The experimental methodology was applied in this study. The research developed a sample practical project, a coin wallet application based on block-chain, in order to answer the research question. The sample application was implemented in Android native and three different cross-platform frameworks respectively. Table 1 shows the data to be collected from each application. Most of the data, such as UI Design, functionality, and workload were collected through the process of the development, while the performance data was collected using the “mobileperf”, which is an open-source tool for monitoring performance, such as CPU and memory usage.

Table 1. Plan of data collection

Type	Description
UI Design	How efficient and friendly are the UI design tools?
Functionality	Whether the framework can meet the functions required in the project, such as cryptographic algorithms, UI elements, and communication with the server.
Performance	The performance of the application, including CPU usage, memory usage, and the size of the release package.
Workload	How much code was written in the project?

3.2 Project Design

The Coin wallet is a financial application running on the phone that allows users to transfer their money to others. The features, such as UI, network communication, and various algorithms, were developed in the sample project. The project contains a server-side and four client-sides (see Figure 1). The server-side is a daemon service running on a workstation to transmit the blocks data to clients. The client-sides include four applications that were implemented by Android native, React Native, Flutter, and Xamarin respectively. To ensure the project can be tested, both server-side and client-side were deployed under a single subnet.

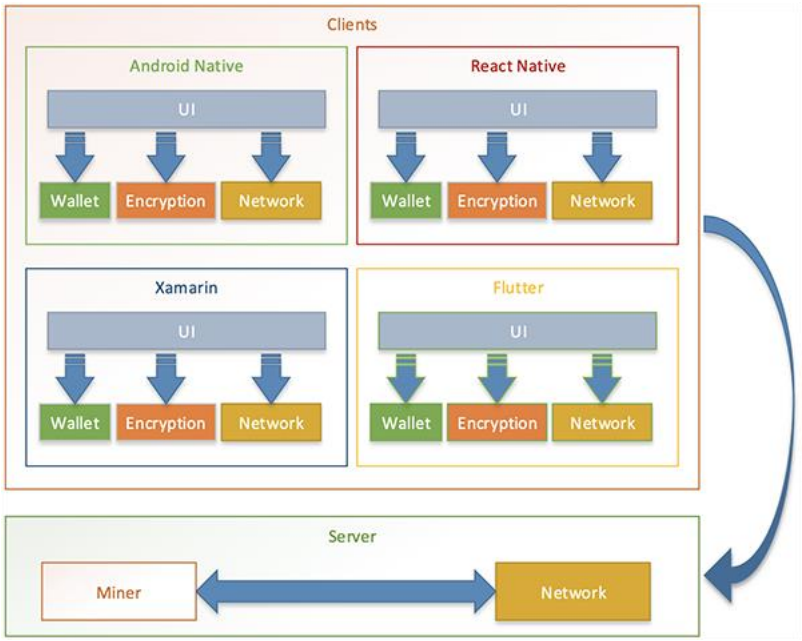

Figure 1. Workflow of Server-Clients architecture

Each client application is composed of the wallet, encryption algorithm, and network implementation. The UI module is in charge of screen rendering related functions by popping up a notification, accepting input text, and presenting rich graphic elements on the phone. The wallet is responsible for transferring or collecting money from others. The encryption module needs to generate, store, and compute the key on the phone. The network module encapsulates RESTful API to make the communication between clients be easy.

An example of workflow for transferring money of $30 from Client2 to Client1 using the sample project is in Figure 2. First of all, both clients need to input the server's IP address in order to ensure all applications communicate through the server properly. Secondly, the receiver (Client1) needs to be ready by clicking the "Collect" button to show its account (QR code). Thirdly, the sender (Client2) is ready of transaction with its initial balance (e.g. $200) by clicking the "Transfer" button; entering the amount (e.g. $30); and scanning the QR code on receiver (Client1) to transfer the money. Fourthly, the receiver (Client1) receives the money and the balance is updated on both clients (e.g. Client1's balance is $30; Client2's balance is $170 = $200 - $30). Finally, both clients click "Transaction" button to check their transaction recorders.

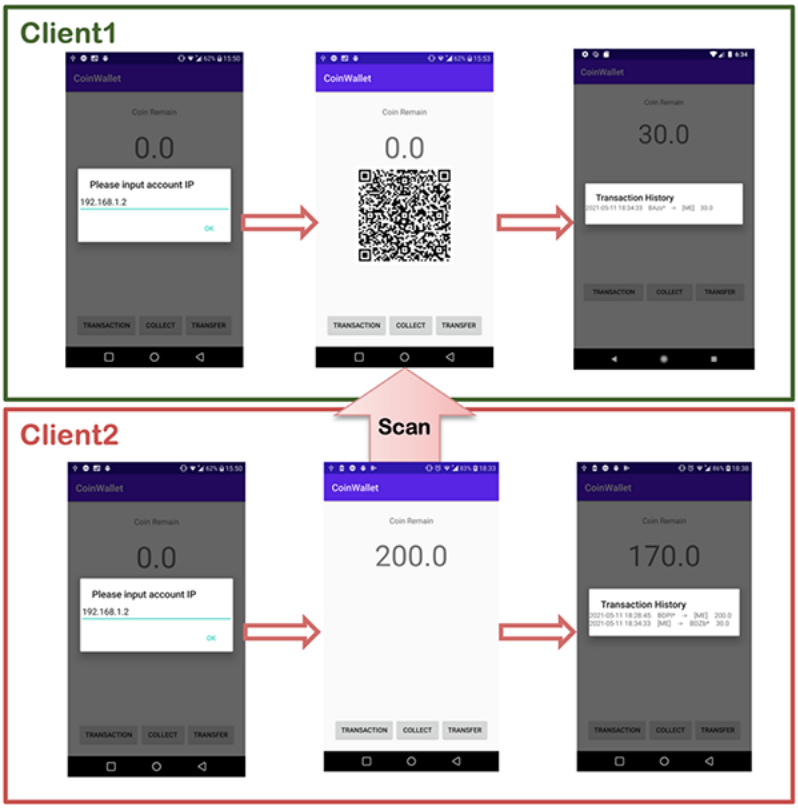

Figure 2. Workflow of applications

4. Research Findings

4.1 Workload

As Table 2 shows, Android Native takes the highest workload in code lines than the other three cross-platform frameworks. Flutter, using Dart in the most concise syntax, requires the lowest workload that is only 47% of the total workload from Android Native. Although React Native costs as much close as that to Flutter on the scale, Xamarin produces a higher workload than both of them.

Table 2. Workload statistics

Workload	Android Native	React Native	Flutter	Xamarin
Java	600	0	0	0
JavaScript	0	445	0	0
C#	0	0	0	524
XML	317	0	0	115
JSON	0	58	0	0
Dart	0	0	432	0
Total	917	503	432	639

4.2 Performance

As to performance shows in Table 3 and Figure 3, the study revealed that the Android Native development takes the lowest CPU usage, the smallest memory usage, and the most compact binary package. Also, the study found that Xamarin runs more efficient not only in CPU usage, but also in memory usage with the smallest binary, while React Native has the lowest performance with the biggest binary package.

Table 3. Memory and APK size comparison

Size	Android Native	React Native	Flutter	Xamarin
Memory Usage	82.67MB	177.92MB	167.45MB	99.78MB
APK Size	3.3MB	30.1MB	21.3MB	12.9MB

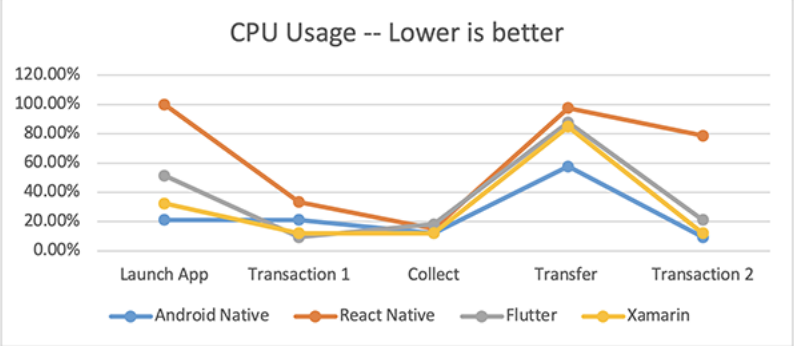

Figure 3. CPU usage chart

Moreover, the study explored that the different development procedure was used among these cross-platforms. Using Android Studio through a plug-in component, React Native divides its development environment into a single tool. However, users must employ a command line to debug and distribute their applications. Using Visual Studio, one of the most powerful IDEs on the Windows platform, Flutter and Xamarin are more user-friendly than React Native.

4.3 UI Development

All the above frameworks can implement the same features of mobile application development. However, users have to face different experience of UI development because each framework has its own characteristics.

Android Studio is an integrated development environment that covers all the life cycle of application development. It provides WYSIWYG experience in code wizards, UI designing, debugging, and deployment. In contrast, React Native divides these processes into several parts that are developed by different single tools. It needs to employ a terminal or command line interface in order to install the environment and also requires to apply different editors in order to edit the code without any auxiliary hint, compose the UI code in the developer's mind instead of WYSIWYG tools. Moreover, both its debugging and publishing application have to go through the command line. In a word, the overall use of React Native is not as user friendly as Android Studio.

Conversely, Flutter is more user-friendly than React Native in the sample project development experience. It utilises Android Studio for code editing and debugging, and for project deployment. It shares the key feature from Android Studio, such as code hint, breakpoint set, variant watch, etc. Although it does not support WYSIWYG, it allows developers to preview the change when debugging.

Xamarin is supported by Visual Studio, which is one of the most powerful IDEs on the Windows platform and provides many features to help developers in their code editing and debugging, and products delivering. It also provides the same WYSIWYG UI design tool as Android Studio. Besides, Visual Studio also has features that allow developers to distribute their applications directly to Google Play and Apple Store. In a word, among the above three cross-platforms, Xamarin offers the best user's experience in UI development for the sample project.

5. Discussion

5.1 Comparison of Cross-platform Frameworks

Since its core is developed by JavaScript, React Native requires an interpreter during execution and rendered UI through native controls. However, Flutter is different because it uses neither WebView nor the native controls of the operating system. Instead, it implements a high-performance rendering engine to draw widgets, which built with C, C++, and Dart programming language. When React Native introduces virtual DOM (Document Object Model) to optimize its renderings, Flutter goes a step further by integrating UI components and renderers into the application, which makes its performance higher than React Native.

Next, it is easy to learn React Native for those who have experiences in web developing or JavaScript coding. On the contrary, Flutter is a new and innovative platform, using Dart language that is not as popular as JavaScript, which means most developers need to learn Dart language when using Flutter.

Nonetheless, Flutter supports Android Studio quite well through plugins, making developers feel like they are using a full-featured IDE, while React Native does not.

Furthermore, Flutter only can be used on the mobile platform, while Xamarin supports Android, iOS, Windows and MacOS platforms, which brings a significant advantage over Flutter. The merit of Xamarin lies in the C# programming language, which is widely used by Windows and .Net developers. Therefore, Developers can use Xamarin directly if they already have C# and .NET programming skills.

Although both Flutter and Xamarin frameworks have powerful IDEs, namely Android Studio and Visual Studio, in order to compile Dart code into machine code, Flutter requires a Dart Virtual Machine (VM) in its running time, which results in decreasing its efficiency. In contrast, Xamarin directly compiles C# code into machine code, which runs much faster than Flutter.

All in all, Xamarin is preferred when the performance is the primary concern or developers are good at .NET technology. React Native is a good choice for those who are proficient in web programming. Flutter is suitable for C++ or Java developers who want to move their current projects to cross-platform projects.

5.2 Limitation

Although all four applications developed by different cross-platform frameworks were run as expected and the data were collected, there were limitations during and beyond the development process, which included:

1. Only limited features were implemented and tested in the sample project, which means it may have exceptions in the performance.

2. During the data collection, all applications were only tested on Android 8.1. The results may vary if using on other versions.

3. During the data collection, all applications were only run on an emulator and a mobile phone, which may lead to inaccurate results.

4. The updates and iterations of all cross-platform frameworks are quite fast. Therefore, the analysis results may vary from version to versions.

6. Conclusion

The study compared three modern cross-platform frameworks through a sample practical project. The results indicate that using an appropriate framework can significantly reduce the workload although all frameworks attempted to achieve the performance in native development. That is to say, different cross-platform technologies are appropriate to different development. Firstly, the Android Native framework is preferred if the project is sensitive to running speed and memory usage. Secondly, React Native is rather suitable for web developers because JavaScript is widely used in web programming. Thirdly, Flutter is recommended for those who are familiar with C++

or Java or who need to port current projects from a native framework to a cross-platform framework because it employs a C-like programming language, Dart. And finally, Xamarin is more suitable for .NET developers or those who are concerned about performance because it is easy to cross-platform development and to implement the most efficient application among cross-platform frameworks using C# in Visual Studio. In one word, considering the limitations above, there is no silver bullet as the best cross-platform framework among them, which is suitable to all the developers in different development background.

For further improvement, the following future works would be considered. Firstly, more features should be involved and tested in various APIs in terms of the performance. Secondly, more devices should be used to reveal whether there are any differences among them, or whether there are any device-independent restrictions. Meanwhile, different devices should run on different OS versions, which can verify whether the data are the same from them. Last but not least, the data should be collected and analysed again after a new update occurs from the frameworks, because each update may include optimizations or degradation.

In summary, all cross-platforms can improve development efficiency greatly, but different frameworks are suitable for different applications and different developers and different scenarios, which means decision-makers need to make choices based on actual projects. The future research would trace the effects of using various cross-platforms for the further improvement.

References

Al-Bastami, B. G., & Naser, S. S. A. (2017). *Design and Development of an Intelligent Tutoring System for C# Language*.

Avdic, D. (2019). *React native vs xamarin-mobile for industry*.

Biørn-Hansen, A., Grønli, T.-M., Ghinea, G., & Alouneh, S. (2019). An Empirical Study of Cross-Platform Mobile Development in Industry. *Wireless Communications & Mobile Computing (Online); Oxford*, *2019*. http://dx.doi.org.whitireia.idm.oclc.org/10.1155/2019/5743892

Cantù, N., Ducci, M., Ahmetovic, D., Bernareggi, C., & Mascetti, S. (2018). MathMelodies 2: A Mobile Assistive Application for People with Visual Impairments Developed with React Native. *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility*, 453–455. https://doi.org/10.1145/3234695.3241006

Cheon, Y., & Chavez, C. (2020). *Creating Flutter Apps from Native Android Apps*. 10.

Dagne, L. (2019). *Flutter for cross-platform App and SDK development*. 37.

Danielsson, W. (2016). *React Native application development: A comparison between native Android and React Native*. http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-131645

Delia, L., Galdámez, N., Thomas, P., Corbalán, L., & Pesado, P. (2015). *Multi-platform mobile application development analysis* (p. 186). https://doi.org/10.1109/RCIS.2015.7128878

Dobrzański, D., & Zabierowski, W. (2017). The comparison of native apps performance on iOS (Swift) and Android with cross-platform application-Xamarin: Student project. *International Journal of Microelectronics and Computer Science*, *8*(3).

Fayzullaev, J. (2018). *Native-like Cross-Platform Mobile Development: Multi-OS Engine & Kotlin Native vs Flutter* [Fi=AMK-opinnäytetyö|sv=YH-examensarbete|en=Bachelor's thesis|]. Kaakkois-Suomen ammattikorkeakoulu. http://www.theseus.fi/handle/10024/148975

Fentaw, A. E. (2020). *Cross platform mobile application development: A comparison study of React Native Vs Flutter*. https://jyx.jyu.fi/handle/123456789/70969

Gill, O. (2018). *Using React Native for mobile software development* [Fi=AMK-opinnäytetyö|sv=YH-examensarbete|en=Bachelor's thesis|]. Metropolia Ammattikorkeakoulu. http://www.theseus.fi/handle/10024/143282

Gu, Y., Xu, C., & Zheng, M. (2017). *Using React Native in an Android App*. 6.

Idan Arb, G., & Al-Majdi, K. (2020). A Freights Status Management System Based on Dart and Flutter Programming Language. *Journal of Physics: Conference Series*, *1530*, 012020. https://doi.org/10.1088/1742-6596/1530/1/012020

Jagiełło, J. (2019). *PERFORMANCE COMPARISON BETWEEN REACT NATIVE AND FLUTTER*. 26.

Kravtsov, D. (2018, December 20). *React Native for your project: Advantages and Disadvantages*. https://belitsoft.com/react-native-development/react-native-advantages

Martinez, M. (2018). Two Datasets of Questions and Answers for Studying the Development of Cross-platform Mobile Applications using Xamarin Framework. *ArXiv:1712.09569 [Cs]*. http://arxiv.org/abs/1712.09569

Radi, A. A. (2016). *Evaluation of Xamarin Forms for Multi-Platform Mobile Application Development*. 23.

Willocx, M., Vossaert, J., & Naessens, V. (2016). Comparing performance parameters of mobile app development strategies. *Proceedings of the International Conference on Mobile Software Engineering and Systems*, 38–47. https://doi.org/10.1145/2897073.2897092

Xanthopoulos, S., & Xinogalos, S. (2013). A comparative analysis of cross-platform development approaches for mobile applications. *Proceedings of the 6th Balkan Conference in Informatics*, 213–220. https://doi.org/10.1145/2490257.2490292