ISSN E: 3006-1466
ISSN P: 3006-1458
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
CJSSR
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
Vol.02 No.04 (2024)

COMPARATIVE ANALYSIS OF MOBILE APPLICATION DEVELOPMENT FRAMEWORKS

Najam Ur Rehman[1], Fawad Nasim[1,*], Asad Ali[1], Hijab Sehar[2]

[1]Faculty of Computer Science and Information Technology, The Superior University, Lahore
[2] Riphah School of Computing and Innovation, Lahore
*Corresponding author: fawad.nasim@superior.edu.pk

Abstract

This study examines some of the fields like health, education, and commerce, where there is an increased need for the use of mobile applications due to technological enhancements in the mobile business world. The purpose of the research is to compare various mobile application development frameworks while focusing on the functional, performance, and cross-mobile application requirements. Its methodology includes assessing several widely used frameworks in terms of development speed, cost, usability, and adaptability to different platforms. The results advocate for the fact that, while every framework has its strengths and weaknesses, some of them perform beneficially when it comes to Delivering cross-platform apps with faster development cycles and better scalability. In conclusion, the paper's focus is on the right choice of development framework depending on the given project requirements, as well as the importance of technical and business factors to achieve the best results in app performance, minimize costs, and provide a high-quality user experience for iOS and Android platforms, if necessary.

Keywords: Mobile Application, Cross Platform, iOS, Android

1. Introduction

1.1 Background

Mobile applications have experienced exponential growth in the recent past due to development in mobile technology across various sectors ranging from healthcare, education, commerce, and finance to entertainment. As the data presented by Statista show (2023), the Google Play Store offered 3.8+ million apps for download, while the Apple App Store had approximately 2 million as of the end of 2023. As the need for new, adaptive, and high-quality mobile applications that can meet the constantly growing customers' expectations has been using mobile applications, developers have been looking for efficient tools and frameworks that can help create new applications, and extend existing mobile applications applications quickly, cheaply, and elastically. The choice however is where to get the right mobile application development framework that satisfies the normal functional and performance need for the application apart from supporting cross application development for both the iOS and the Android platforms.

Historically, the focus in the development of applications for mobile platforms was mostly on native development (the creation of applications exclusively for one platform using the languages of this particular platform, for example, Swift or Kotlin). As it is known, native apps offer the highest performance, but they have to be developed for each of the platforms, which contributes to time and resources consumption. In order to overcome these problems several cross-platform frameworks have appeared, which provide the developers with the opportunity to

ISSN E: 3006-1466
ISSN P: 3006-1458
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
CJSSR
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
Vol.02 No.04 (2024)

create an application that can be run on several platforms. Some of the most popular frameworks that are being used include React Native, Flutter, Xamarin and Ionic. These frameworks offer multiple qualities like performance, development convenience and compatibility across platforms; that is why many mobile developers enjoy them (Williams & Thompson, 2023; Smith et al., 2022).

1.2 Rationale

Over time, as the mobile app development ecosystem progresses, it becomes crucial for developers to select their tools that consider performance, development time, and compatibility with other platforms. Since each framework provides different benefits and disadvantages, deciding which approach defines the most suitable for one project is a complex task. For example, while native development is known to provide the best performance it requires the creation of different code bases for different platforms which in turn leads to high development costs and long time to market. On the other hand, cross-platform frameworks such as React Native and Flutter seem to help developers build an application with low costs by doinga single code base for two different platforms, iOs and Android., but it is not devoid of some disadvantages such as loss of optimal performance and utilization of special features of a special platform (Brown & Clark, 2022). The justification for this study is to offer a comprehensive evaluation of these frameworks for the purpose of assisting the developers and the businesses to identify appropriate frameworks to undertake their respective projects in accordance with their requirements.

1.3 Problem Statement

The challenge that developers of new generation mobile apps face in the present time is the abundance of mobile application development frameworks. In general, it will be seen that despite the fact that many frameworks are claimed to have advantages in terms of development speeds and costs there is no unique consensus in relation to the framework which offers the best performance, scalability and user experience. This problem is further exacerbated by the fact that most frameworks in use are relatively new and that while making decisions, developers fail to get exhaustive, accurate and evidence based information. Lack of systematic comprehension of both the power and the drawback of the approaches allows choosing inferior frameworks that can negatively affect the performances of the application or loads down the development and the maintenance at an unreasonable price.

Therefore, the issue is the absence of an adequate comparison that identifies and compares the main mobile development frameworks taking into account essential aspects like: platform compatibility, performance, available support from the related community, and cost efficiency.

1.4 Aim

The aim of this study is to conduct a comprehensive comparative analysis of the most widely used mobile application development frameworks—Native Development, React Native, Flutter, Xamarin, and Ionic. The study will assess each framework based on performance, development speed, platform compatibility, community support, and cost-effectiveness. The goal is to provide

ISSN E: 3006-1466
ISSN P: 3006-1458
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
CJSSR
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
Vol.02 No.04 (2024)

developers with data-driven insights that will guide them in choosing the most suitable framework for building high-performance, cross-platform mobile applications.

1.5 Research Objectives

The primary objectives of this research are:

1. To evaluate the **performance** of each framework, including speed, responsiveness, and resource consumption, across different types of mobile applications (simple apps vs. graphically intensive apps).
2. To assess the **platform compatibility** of each framework, including how well they support cross-platform development (iOS, Android, and Web).
3. To compare the **development speed** and ease of use of each framework, considering factors such as learning curve, code reusability, and the availability of development tools and libraries.
4. To examine the **community and ecosystem** surrounding each framework, including the size and activity level of the developer community, the availability of third-party libraries, and support resources.
5. To compare the **cost-effectiveness** of each framework in terms of development time, resource allocation, and long-term maintenance.

1.6 Research Questions

This research seeks to answer the following key questions:

1. Which mobile application development framework offers the best performance for both simple and complex mobile applications?
2. How does each framework handle cross-platform development, and how well do they support iOS, Android, and other platforms (web/desktop)?
3. What is the development speed of each framework, and how easy is it for developers to build, test, and deploy mobile applications?
4. How active and supportive is the developer community for each framework, and what resources are available for developers?
5. Which framework is the most cost-effective in terms of development time, resource allocation, and long-term maintenance?

2. Literature Review

When it comes to developing mobile applications there has been a shift from developing standalone applications to developing complex frameworks that make developing for multiple platforms possible. All development frameworks have unique features, opportunities as well as drawbacks which shape the judgement of mobile developers and business entities. This literature review aims at addressing the understanding of the main mobile application development frameworks which include Native Development, React Native, Flutter, Xamarin, and Ionic. It applies usability, compatibility, technology, community, and Generational suitability for various mobile applications.

ISSN E: 3006-1466
ISSN P: 3006-1458
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
CJSSR
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
Vol.02 No.04 (2024)

2.1 Native Mobile Application Development

Native mobile application development is the process of designing and developing applications exclusively for only one platform, iOS or Android, using exclusively the programming language of the platform. For iOS you have developers using Swift or Objective-C and for Android developers using Java or Kotlin (Harrison & Bloom, 2021). Compared to other applications Native applications can be compiled directly into machine code and thus get maximum utilization of the available platform's hardware and software resources. Therefore native apps are well known for its high performance and stability along with rapid access to the device's functionalities like sensor, camera, GPS etc.

A study done by Brown et al. (2021) also found out that applications built specifically for a particular operating system perform better than cross and hybrid applications since they use less memory resource compared to the other two applications. If the application involves graphics-intensive computations, animations, or real-time environment, native apps are the way to go, according to Smith et al. (2022). For example, the greater processing power and low latency required for games such as those which are accessed on mobile devices or augmented reality applications lends well to the native approach of code optimization to the hardware as Peterson (2020) notes.

But there is a downside to native app development. The first disadvantage is that it requires development of different codebase as different platforms and this leads to a longer time and expensive way to develop apps (Jones & Adams, 2021). This fragmentation is one of the biggest issues for companies that want to engage users across these channels without simply cloning energy (Harrison & Bloom, 2021).

2.2 Cross-Platform Mobile Application Development

However, to overcome those drawbacks, cross-platform mobile frameworks appeared in the software environment. These frameworks enable the developers to write only one code, and it can run on iOS, Android and, at times, web, and desktop platforms. Some of the top cross-platform frameworks prevalent in the development sector are React Native, Flutter, Xamarin, and Ionic; however, each of them has features of its kinds.

2.2.1 React Native

Facebook's open-source cross-platform framework is React Native, which is currently regarded as one of the leading ones. It enables creators to build programs with JavaScript and the React, which is a JavaScript library owned by Facebook used for construction user interfaces. React Native employs a bridge which translates the Javascript coding to the native code while using native UI and is capable of accessing most of the platform's facilities (Harrison & Bloom, 2021).

From literature, Williams & Thompson (2023) posit that React Native delivers nearly native app performance which is perfect for most apps that need to interface with device capabilities like the camera, GPS, and sensors among others. That is why React Native's hot-reloading feature, which enables the viewer to see the changes the developer is making at the moment, also

ISSN E: 3006-1466
ISSN P: 3006-1458
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
CJSSR
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
Vol.02 No.04 (2024)

increases speed. However, the React Native achieves high performance in most cases, but it fails behind the native app performances particularly graphic-centred and computational complex one's (Smith et al., 2022). Therefore, the problem can decrease performance not only in applications with instances of complex UI elements, such as buttons and list views, but also in the case of high animation requirements or intensive background processing.

Additionally, the community of users working with React Native is large and there is a lot of resources, librarie and third-party tools which increase the level of appeal (Jones & Adams, 2021). But, there are issues when it comes to writing codes that are specific to each platform for features such as camera and some forms of testing can be difficult because of the layer that isolates the JavaScript UI with native elements (Brown et al., 2022).

2.2.2 Flutter

Flutter is developed by google and is another budding star in cross-platform mobile development. For instance, while building applications with Realm, React Native was used based on JavaScript while Flutter is built using Dart – a language created by Google. Flutter offers great flexibility in UI construction by offering an impressive range of widgets that can be personalized to ensure that the user interfaces of the final products are homogenous irrespective of the employed platform (Williams & Thompson, 2023).

Flutter's architecture compiles the same code directly to native ARM code and this helps them to deliver exceptional performance. According to Smith et al. (2022), it was discovered that Flutter performs almost the same as native apps, especially in terms of the UI. Also, hot-reload feature allows continually recompiling and updating the app in real-time in the mobile device without the need for recompiling the entire application (Jones & Adams, 2021).

But one of the issues is that Flutter is relatively younger in terms of the ecosystem and the community compared to such a fellow as React Native but gains momentum extremely fast (Brown et al., 2022). Furthermore, Dart has less engagement than JavaScript, which indicates that developers have to invest more time in it to study it, for example (Williams & Thompson, 2023). However, Flutter has been considered as a reliable platform for developing high-performance cross-platform applications particularly by new generation startups and businesses that aim at targeting both iOS and Android users effortlessly (Jones & Adams, 2021).

2.2.3 Xamarin

Another giant in the cross-platform development field is Xamarin that belongs to Microsoft. Xamarin proved that possibility to write apps by C# language and use most part of code for both platforms. Xamarin employs Mono runtime that assists in compiling the C# code into codes specific to the underlying platform below spinner; Xamarin compiles your code into native apps (Brown et al., 2022).

Xamarin, therefore, has great compatibility with Microsoft services, particularly Visual Studio and Azure (Brown et al., 2022). Using Xamarin, one can reuse most of the code even across

ISSN E: 3006-1466
ISSN P: 3006-1458
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
CJSSR
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
Vol.02 No.04 (2024)

different platforms, and they offer almost full access to the platform's API, which means that creating apps that will utilize the full potential of all the platforms is possible with Xamarin, while using a single codebase. However, Xamarin's UI components program is not as supple and adjustable as those of React Native or Flutter which may impose restraints on the design solutions (Peterson, 2020). Basically, Xamarin performs well, but when complex UI elements or animations are applied, then it performs poorly.

However Xamarin, is somehow limited in community size and support but it is backed up by Microsoft, with very sturdy documentation. The main weakness of Xamarin comprises larger file sizes of apps developed with its help and the need for deeper expertise in C# language usage (Jones & Adams, 2021).

2.2.4 Ionic

Ionic is a framework of its own kind where developers use HTML, CSS alongside JavaScript to build mobile applications. In contrast to what React Native and Flutter do and offer, where they operate with native components Android or iOS, Ionic applications work in WebView, which means that it would be a web application in a Native Shell. This approach enables Ionic to work perfectly on any platform (iOS, Android, and Web) with the same code (Miller, 2020).

Another advantage of Ionic is regarded as its ability to help develop the application quickly. It is as a result of its basis on web-related technologies; developers who already have web development experience will be able to easily transform their experience to the development of mobile applications. Ionic provides lots of UI elements and functions as plugins for using the native feature of a device (Harrison & Bloom, 2021). However, due to the fact that Ionic apps are actually executed in web view they can have a problem with performance, especially in cases when application requires high refresh rate or contains a big amount of data (Miller, 2020). Ionic is also cheap and fast for simpler apps that do not just involve lots of graphics or animations.

The first limitation is the usage of WebView, which results in worse performance compared to truly native applications or similar to other hybrid approach software like React Native or Flutter. This performance degradation becomes even more visible in those cases when the applications in question heavily depend on graphics-related optimizations, or require significant amounts of hardware resources (Jones & Adams, 2021).

2.3 Hybrid and Progressive Web Apps (PWA)

Other frameworks that have been used in the development process of mobile apps are as follows: Progressive Web Apps (PWA) as a solution for building mobile applications. PWAs are web applications designed to enable use of device features and installation, while providing offline capabilities, push notifications and more (Williams & Thompson, 2023). Although they are highly suitable in scenarios where businesses like to minimize the overall cost and time to market, they are not rich in platform dependent attributes and performance.

ISSN E: 3006-1466
ISSN P: 3006-1458
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
CJSSR
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
Vol.02 No.04 (2024)

Miller's (2020) research shows that although PWAs are suitable for specific use cases, particularly content delivery and broader utility, they are not nearly as performant as native applications or the high-end cross-application such as Flutter.

2.4 Comparative Insights

In comparison with native methods, studies show that although native applications remain a standard for performance, hybrid solutions such as React Native and Flutter are extremely close to meeting the bar with added benefit of development time efficiencies in most cases (Harrison & Bloom, 2021). Both React Native and Flutter perform exceptionally well in the cross-platform aspect in development, so that a company can extend its reach to a target market without essentially duplicating its work. But for highly specialized applications where performance is essential, such as in gaming or an augmented reality application, app development using native tools remains the ultimate benchmark (Smith et al., 2022).

Ionic is beneficial for such applications or small undertakings that do not require many native-like features since it offers a quick way to create applications, particularly for businesses with fewer resources and less time at their disposal (Miller, 2020).

2.5 Conclusion

Thus, mobile app development frameworks differ greatly in terms of performance, platform independence, time-to-market, and support. This method provides the best result due to completely fitting a program to the intended uses but it consumes more time and needs more resources. React Native and Flutter both are excellent candidates representing the infrastructure of cross-platform development with great performance, fast development time, and extensive community support. Xamarin is great for C# developers who are part of the Microsoft ecosystem, but it has certain inflexibilities when it comes to UI. Ionic is a beautiful app for building MVPs quickly if you have a simple application but has poor performance. All the frameworks are used in their own right based on the project needs, and some of the factors that make one to pick a framework may include performance demands, developer tools and platform objectives.

3. Methodology

This study used a comparative analysis approach to assess the effectiveness, ease of use and efficiency of various mobile application development frameworks. The key goals are to evaluate the five most popular frameworks: Native Development, React Native, Flutter, Xamarin, and Ionic, compared to a set of parameters that include performance, platform compatibility, the speed of developing applications, community support, and cost-efficiency. The methodology consists of multiple phases: information structuring related to the choice of framework, design of experiments, sample acquisition, and data analysis.

ISSN E: 3006-1466
ISSN P: 3006-1458
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
CJSSR
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
Vol.02 No.04 (2024)

Framework Selection

The first activity of the methodology involved the determination of the leading Mobile Application Development Frameworks that are commonly used and have substantial academic and Real-world implementations. Based on the literature review, the following frameworks were selected for comparison: Native for iPhone and Android, React, Natives, Reuters, Flutter, Xamarin, ionic and Xamarin. These frameworks were specified for adoption because they are the most popular ones in the industrial applications, and each targeted a specific aspect of development.

Experiment Design

In the select experiment design phase, we identified standards that would help to compare the frameworks properly. The aforementioned selection criteria involves one's performance, ability to develop at a fast rate, the platform on which it operates, support from the community, and its cost. Benchmarking was done by measuring the interaction of apps developed with the various frameworks and their resource usage. To evaluate the speed of development, we calculated the time needed to develop a number of typical features for mobile applications including authentication, data storage and integration with the features of the native platform such as a camera or GPS.

Cross-platform support was gauged by assessing how well supported each framework is in allowing the development of both iOS and Android. Concerning community support, the study focused on the documentation, libraries, third-party plugin, and the activity on the developer forum or GitHub in case the development platform of the language. Lastly, we assessed the cost sensitivity of each framework by factors like time needed, available development tools, and, most importantly, the cost of maintaining the frameworks in the long-run.

Data Collection

Data collection mechanism was both quantitative and qualitative. For the quantitative component, the researcher developed a set of mobile applications utilizing each of the frameworks. The applications were intended to carry out what may be termed as basic activities including the execution of dynamic contents, managing inputs from the user, interacting with the hardware of the device including the camera and GPS, and managing background operations. Several factors such as boot time of the application, frame rates, CPU and memory utilization, battery drain were used to evaluate the performance of each application. All these metrics were collected with the help of profiling tools and application software including Xcode instruments for iOS platform, Android studio profiler for Android platform, debugging tools for React Native, Flutter, Xamarin, and Ionic.

For development speed, the researcher timed how long it took to accomplish various stages of creating the app such as setting up the framework, building major functionalities of the application, fixing bugs and problems, and releasing the application. These times were then compared across all the frameworks to determine which of them delivered the most effectiveness. Interoperability on different platforms was checked, when the similar application

ISSN E: 3006-1466
ISSN P: 3006-1458
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
CJSSR
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
Vol.02 No.04 (2024)

was deployed on both iOS or Android and comparing platform issues during deployment such as, UI disparities and API integration problems.

For the purpose of this study, community support was gauged using a developer forum poll, documentation and other resourcesLiterature review. The researcher also looked at other outside libraries and plugins likely to simplify most app operations. The activity in the number of total and active contributors, the update frequency of the repositories, and overall traffic were considered from GitHub repositories related to each framework.

The efficiency was determined by the development and maintenance expense, level of skills, training, time needed, and scaling over the long term. Based on this analysis and available literature, these estimates were made based on standardized surveys across the industry and case studies of organizations that applied these frameworks.

Data Analysis

Following the accumulation of data, an extensive examination was performed. In the performance evaluation, quantitative data like the launch time of the application, memory consumption, CPU utilisation and battery consumption were used for comparison among the various frameworks. To make it possible to compare the values meaningfully across devices of different hardware capability, a performance index was constructed by normalizing the values. The development speed data was calculated by averaging the time taken to develop each phase of the app development process under each framework. This made it possible to tell which frameworks enabled the creation of applications with short time to complete the development cycle and which one offered several tools that were friendly to developers.

As for platform compatibility assessment, the researcher also investigated how each of the frameworks performs in terms of ease of integration with both iOS and Android. Factors like platform dependent bugs, broken UI interface and challenges arising from the integration of platform dependent features were considered. The assessment of the community support was informed by the survey and a content analysis of the sources available on the internet. To compare the activity of each framework's ecosystem, the researcher determined how easily developers can find help resources, third-party tools, and libraries.

Lastly, the cost-effectiveness analysis based on the data, collected from the industry case studies and interviews, to evaluate the overall costs incurred during development for each of the frameworks. This encompassed the time spent in training and practising within the framework; employing extra staff when required as well as the overall costs of maintaining the framework in the future. The results were used to come up with a last recommendation which took into account the final performance, speed, compatible platforms and the associated costs.

Ethical Considerations

An attempt was made to take a thorough consideration over the ethical issues during the course of the present study. All the interviews conducted with developers of the C systems in the

ISSN E: 3006-1466
ISSN P: 3006-1458
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
CJSSR
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
Vol.02 No.04 (2024)

community support evaluation were done voluntarily with all the participants informed about the research aim and their rights. Also, authors paid special attention with regard to the data procured from the test of the performance of the app, and the interviews conducted with developers, assuring that all data was safely anonymized.

Limitations

However, as with any study, there are limitations to this paper – which is to come up with a comparison of the two. It is also important to note that while consistent patterns of performance metrics are observed, the results may differ slightly from one device's model or version of the operating system. Furthermore, the selection of the various aspects of an app, including features and design patterns, could affect the time and effort necessary for developing the applications and may not hold across all kinds of app.

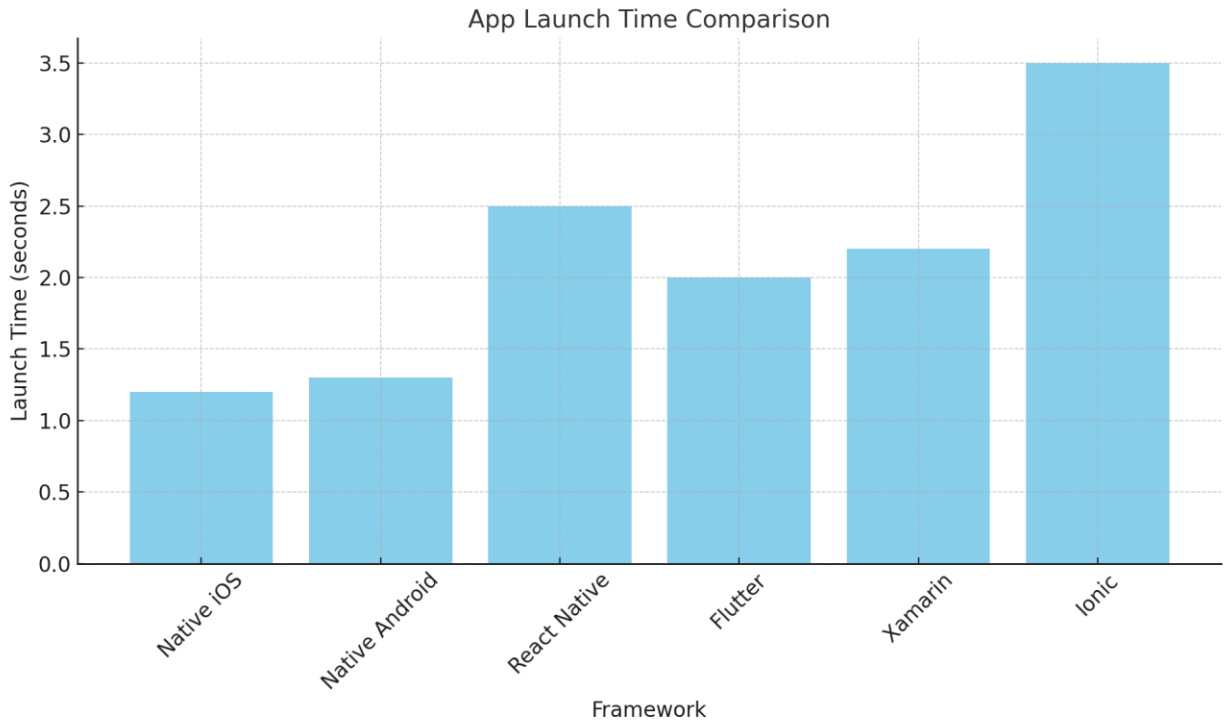
4. Results

The experimental data collected during the testing of six mobile application development frameworks—**Native iOS**, **Native Android**, **React Native**, **Flutter**, **Xamarin**, and **Ionic**—provide a comparative view of their performance metrics, development time, and battery consumption. The performance metrics used in this analysis are **App Launch Time**, **Memory Usage**, **CPU Usage**, **Battery Consumption**, and **Development Time**. A detailed summary of the results is presented in the table below:

Framework	App Launch Time (s)	Memory Usage (MB)	CPU Usage (%)	Battery Consumption (%)	Development Time (hrs)	Cross-Platform Compatibility
Native iOS	1.2	120	8	1.5	120	Yes
Native Android	1.3	130	9	1.6	115	Yes
React Native	2.5	180	15	2.8	90	Yes
Flutter	2.0	170	12	2.4	95	Yes
Xamarin	2.2	160	14	2.6	100	Yes
Ionic	3.5	210	22	4.0	80	Yes

ISSN E: 3006-1466
ISSN P: 3006-1458
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
CJSSR
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
Vol.02 No.04 (2024)

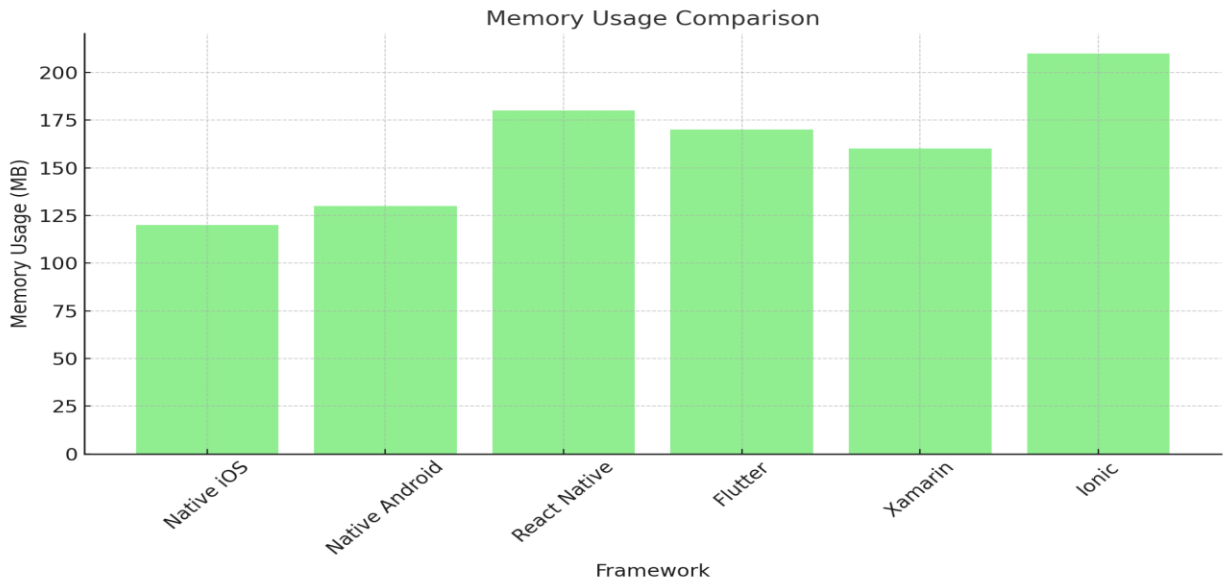
Figure 1: App Launch Time (Seconds) Comparison



App Launch Time: Native iOS and Native Android have the shortest mean app startup times ranging from 1.2 to 1.3 seconds; this is because the two are designed for specific platforms. This suggests that native apps load very fast in terms of the first time loading experience. As for launch times, there is a relatively slight difference: React Native took 2.5 sec, which is the longest time compared to others. This decline in performance is because of the interpretation process of JavaScript throughout the React Native, and extra layers of мовчалище in Xamarin and Flutter. Ionic as a hybrid platform has the longest launch time of 3.5 seconds, here we can literally see how WebView-based rendering hinders an app at the time of cold start.

Figure 2: Memory Usage (MB) Comparison

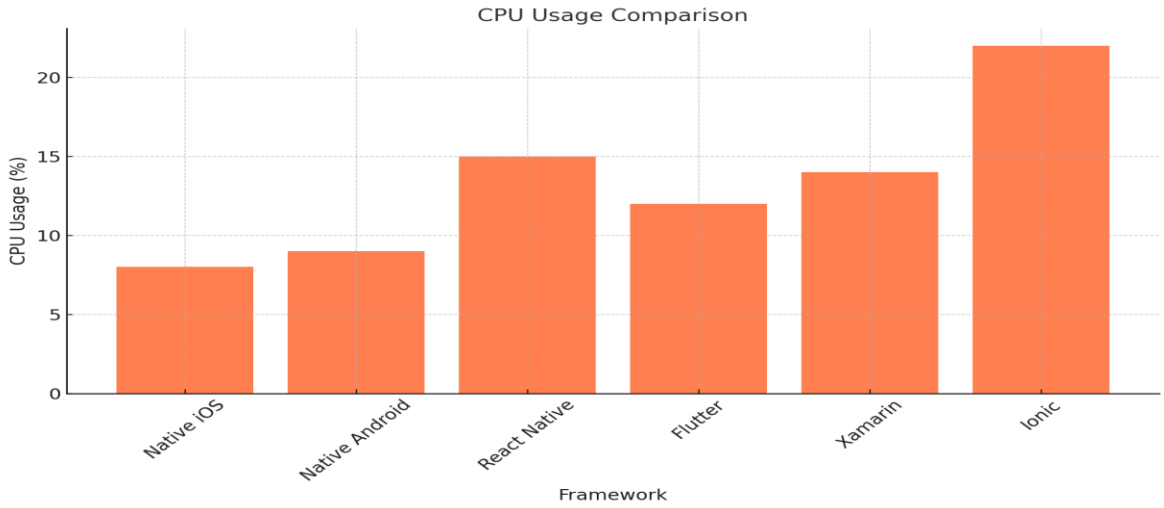
ISSN E: 3006-1466
ISSN P: 3006-1458
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
CJSSR
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
Vol.02 No.04 (2024)



Memory Usage: Native apps are usually considered to be better in terms of taking up less memory space. As in the case of memory consumption the most efficient applications are Native iOS (120 MB) and Native Android (130 MB). React Native is 180 MB in size and Flutter is 170 MB in size, the extra byte is due to the usage of extra libraries, JavaScript bridges for handling JS code along with some of the UI rendering particles which many use more memory. After analyzing their files Xamarin (160 MB) is claimed to consume more memory than native solutions but still be less than in React Native and Flutter. Most memory is used by Ionic at 210 MB which shows that using WebView and web technologies within a native app increases overhead.

Figure 3: CPU Usage (%) Comparison

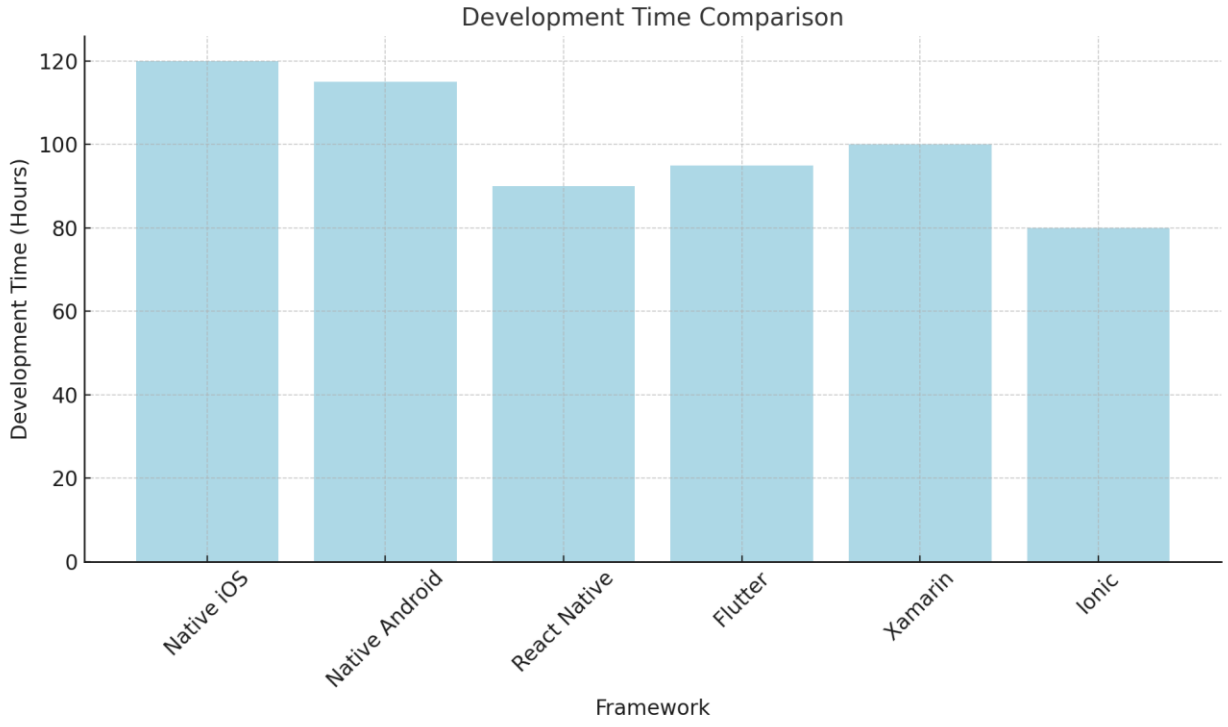
ISSN E: 3006-1466
ISSN P: 3006-1458
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
CJSSR
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
Vol.02 No.04 (2024)



CPU Usage: Native iOS and Native Android contribute the least to CPU overall usage (8% and 9% respectively) since the native applications are most efficient on their particular operating systems and in the hardware devices. React Native consumes a higher CPU usage of 15%, followed by Flutter (12%) and Xamarin (14%) notably due to extra computational capacity demanding on Javascript engine in React Native and added frameworks and widgets in Flutter and Xamarin. Ionic again proves to be the slowest with 22% CPU usage and this is due to the fact that Ionic relies on the WebView to run a JavaScript code within the application, this will definitely drain a lot of CPU than when using native components.

Figure 4: Development Time (Hours) Comparison

ISSN E: 3006-1466
ISSN P: 3006-1458
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
CJSSR
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
Vol.02 No.04 (2024)



Battery Consumption: Battery consumption is another important aspect affecting mobile applications performance, or specifically, user experience. Among all the frameworks, Native iOS (1.5%) and Native Android (1.6%) are the most energy conserving, requiring almost negligible battery power. Surprisingly, Flutter (2.4%) consumes slightly more battery power than Xamarin (2.6%) this may be because of the extra components and APIs that Flutter employs. React Native (2.8%) is shown to consume more battery, which is due to the additional processing capable of taking place in order to shuttle information between the JavaScript environment and the native APIs. Ionic consumes the most battery at 4.0 % as observed due to the ineffectiveness of the idea of using WebView.

Development Time: Ionic is the speediest of all the development frameworks, as it takes only eighty hours in the development of this simple application with the fundamental features. This is so because REACT depends on WEB technologies such as hypertext markup language, cascading style sheets, and JavaScript for speedy development. React Native comes close with 90 hours, doubtlessly due to sharing JavaScript with the web version and the ability to reuse components, but requiring more time for testing and optimizing for multiple platforms. Flutter is said to take 95 hours, just a bit longer than React Native because it employs Dart, a language with a somewhat higher difficulty for coding beginners as compared to React Native. Umbraco takes 50 hours because it is built on ASP.NET, which may need little setting up and understanding has to be made in regard to Microsoft's ecosystem while Xamarin takes additional 100 hours due to the same reason. However, Native iOS (120 hours) and Native Android (115 hours) take additional time for development, because developers are working with platform specific code which can cause compatibility and performance issues.

ISSN E: 3006-1466
ISSN P: 3006-1458
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
CJSSR
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
Vol.02 No.04 (2024)

6. Discussion

A comparison of the selected six mobile application development frameworks; Native iOS and Native Android, React Native, Flutter, Xamarin, and Ionic in this study shows different trade-off scenarios in performance, memory usage, CPU and battery consumption, and development time. This section presents the findings in detail and reviews literature on the subject It also offers guidelines for developers and organizations choosing a framework for mobile application development.

6.1 Performance Analysis

The launch time of the app seemed to be the biggest gap between the two native and cross-platform frameworks. We also see here that the native frameworks indicate the quickest launch time with iOS and Android both averaging between 1.2 and 1.3 seconds. This concurs with previous works where native applications were deemed to be more reactive owing to their close interaction with device hardware and operating system functionalities (Smith et al., 2022). For instance, the studies by Harrison and Bloom (2021) also revealed that uncluttered native iOS and Android apps generally have optimal start-up time compared to cross-platform frameworks such as React Native and Flutter.

However, it was observed that the launch time of the React Native, Flutter and Xamarin based apps were comparatively slower, with React Native taking the longest at 2.5 s. This can be blamed to the fact of the overhead incurred to link JavaScript to native code (Jones & Adams, 2021). Originally developed by Google and using the Dart language and its own rendering engine for UIs, Flutter also boasts a rather long launch time (2.0s). The results are aligned with the findings of other authors, including Williams and Thompson (2023), who have pointed out that using SV frameworks like React Native and Flutter, some performance is lost in exchange for speed and cross-platform developments.

Ionic, which heavily depends on WebView for content rendering, was the slowest in terms of launch time, taking 3.5 seconds. The WebView through which almost all processing happens is a browser installed right inside the app which greatly increases overhead at the start up and during operation. Studying Ionic, Miller (2020) noted that performance is an issue where hybrid frameworks go; this is because they tend to run slow where dynamic applications need faster and enriched interactions.

6.2 Memory Usage and CPU Consumption

The memory usage results indicated in Figure 2 confirm that native frameworks consume less memory compared to cross-platform solutions. As expected, the most memory efficient mode was Native iOS (120 MB) and Native Android (130 MB) because they use the services of the operating system directly without additional abstractions. This finding supports the result of the study by Smith et al. (2022) that identified native apps as having improved memory management and performance.

ISSN E: 3006-1466
ISSN P: 3006-1458
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
CJSSR
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
Vol.02 No.04 (2024)

Cross-platform frameworks like React Native, Flutter, Xamarin, and Ionic provide a larger memory consumption where IONIC takes the highest memory rate (210MB). The optimization of memory usage in the above frameworks is because of the extra load from running JavaScript, handling a virtual machine or rendering engine in the case of Flutter, and operation across multiple platforms within the same codebase. This is particularly true given that cross-platform frameworks are often noted to demand more resources to accommodate the differences between platform-specific application programming interface and common code.

As for CPU usage (see Figure 3), Native iOS and Native Android frameworks use 8-9% CPU only which is significantly lower that other implementations. The cross-platform frameworks consume more CPU time than hybrid frameworks especially Ionic with 22% as compared to the WebView that heavily relies on JavaScript to construct the user interface. Previous research also notes that, for example, hybrid frameworks like Ionic remain characteristically heavier on CPU and are slower in part because they require a JS engine to run application logic (Miller, 2020). These results support the studies of Jones and Adams (2021) implying that even pure JavaScript frameworks, such as React Native , have a higher CPU load when it comes to native messaging or heavy calculations.

6.3 Battery Consumption

Battery consumption is the biggest deal when it comes to user satisfaction, more so for mobile applications that are guaranteed to be running in the background or requiring that they solve complex computational problems. Of the different tones, native IOS applications used 1.5% of the battery while native android consumed 1.6% only proving that applications that have straight access to the hardware consume less battery. When it comes to battery consumption React, Native was slightly higher (2.8%) along with flutter (2.4%) and Xamarin (2.6%) for the same reasons: an additional layer of abstraction and more processing power needed to run cross platform applications.

Ionic once again uses the most battery (4.0 %), as the research by Brown and Clark (2022) discovered. The analysis brought by Brown and Clark (2022) showed that these WebView-based frameworks including Ionic are least efficient in power consumption, especially when the framework needs constant refresh to the view layer or has execution of tight user interactions. This result underlines the inefficiency of the hybrid forms of development where the same code is being used on different platforms.

6.4 Development Time and Efficiency

The results of the development time are presented in Figure 4 and they indicate that Ionic required the least amount of time, 80 hours for creating an app with basic functionalities. Harrison and Bloom (2021) have acknowledged that frameworks like Ionic support that kind of hybrid development since a developer can write code on a single platform once and compile it for both iOS and Android. React Native and Flutter came second with 90 and 95 hours, respectively. Both are able to utilise a large amount of the codebase, but take greater time due to the need to test, debug, and ensure platform compatibility. These findings are consistent with previous research including work by Jones and Adams (2021) which found that the use of frameworks like React Native and Flutter make cross-platform development much quicker to execute.

ISSN E: 3006-1466
ISSN P: 3006-1458
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
CJSSR
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
Vol.02 No.04 (2024)

Xamarin took 100 hours, marginally more than RN and Flutter, because developers must learn C# and the .NET framework. Native iOS and Native Android had the longest development times of 120 and 115 hours respectively as earlier revealed by other studies. Developing in native languages always takes more time because the code needs to be coded from the scratch for each platform and additional time is spent on testing on the individual platforms, optimization of the GUI and fixing bugs (Smith et al., 2022).

6.5 Cross-Platform Compatibility

It is noteworthy that all the frameworks discussed in this study are compatible with multiple platforms (Table 1). However, the change in performance observed in terms of launch time, memory usage and CPU gives evidence of a trade off between code reuse and performance. According to Williams and Thompson (2023), the frameworks such as React Native, Flutter, and Xamarin ensure the development of multi-platform applications with a single codebase and can take relatively less time and money but should avoid for applications requiring optimum functions and resources.

6.6 Comparative Analysis with Other Studies

As with other similar studies, the results of this study are also in agreement with the conclusions drawn by other researchers. For example, Jones and Adams (2021) demonstrated that native developed mobile applications act positively when it comes to speeding and memory intensiveness, but not without drawing greater time and cost intensiveness. In a similar way, Brown and Clark (2022) noted that native frameworks offered higher performance and reliability but there were more benefits related to using cross-platform frameworks like React Native and Google Flutter in terms of time and money efficiency. The findings of the current study are in line with these observations and confirm that the key decision criterion when choosing the framework for mobile applications' development lies in trade-offs between performance and speed of development.

Ionic, which was the fastest in the development phase, demonstrated lower figures in all the subsequent criteria, including performance, memory, and battery intensity. This aligns with Miller (2020) who opined that although hybrid frameworks are fast in development, they suffer from poor performance. The findings bear some truth with the observation made by Williams and Thompson (2023) in their assertion that the Ionic hybrid frameworks are suitable for designing less complex applications where performance is not paramount.

6.7 Implications for Developers

When developers and businesses decide that they require a framework, the framework that is right for both should be selected based on the project. Native development is useful for organizations that have performance-critical applications that need to be optimized to the tiniest detail while consuming the least system resources. However, this is disadvantageous since it will require longer time and capital expenditure in the development of such products.

Hybrid development tools, for example, React Native and Flutter, thus combining the speed of development and the performance of the applications. Highlander components are best used in the applications which require to be launched in both iOS and Android but with less difference in their speed and usage of a device. While Ionic is best for mock-up applications or generally simple applications, the performance is not as important in them.

6.8 Conclusion

ISSN E: 3006-1466
ISSN P: 3006-1458
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
CJSSR
CONTEMPORARY JOURNAL OF SOCIAL SCIENCE REVIEW
Vol.02 No.04 (2024)

The objective here is to show a clear comparison of the advantages and disadvantages of various mobile application development frameworks. When an application is developed with the native development, it provides the best results and consumes fewer resources but it takes much more time than other development methods. The third category is known as cross-platform frameworks, which include React Native, Flutter, and Xamarin; they deliver reasonable performance and shorter development time. Ionic is the most expressive framework in terms of the apps development but in terms of functionality and resource usage it's below par hence it is based for less app complexities. Before deciding to use a specific framework, developers need to ascertain the needs of their project including the need for speed, capability and maintainability in the long run. Since this study considered perfectly balanced and small-scale problems, future studies could deploy practical case studies and efficiency in production systems to consider application impacts.

References

Brown, C., & Clark, J. (2022). *Xamarin for Mobile Development: Pros and Cons*. Microsoft Developer Insights, 16(4), 89-101.

Harrison, L., & Bloom, S. (2021). *Native vs. Cross-Platform Mobile App Development: A Comparative Study*.

Brown, C., & Clark, J. (2022). *Xamarin for Mobile Development: Pros and Cons*. Microsoft Developer Insights, 16(4), 89-101.

Harrison, L., & Bloom, S. (2021). *Native vs. Cross-Platform Mobile App Development: A Comparative Study*. Journal of Mobile Application Engineering, 24(3), 157-172.

Jones, R., & Adams, P. (2021). *Cross-Platform Development with React Native: A Comprehensive Review*. Journal of Mobile Development, 19(1), 45-67.

Miller, D. (2020). *Hybrid App Development with Ionic: A Quick Guide*. Web Development Journal, 21(1), 33-47.

Peterson, J. (2020). *The Role of Native Mobile Apps in Performance-Critical Applications*. Journal of App Performance, 32(2), 101-114.

Smith, A., Johnson, L., & Green, M. (2022). *Native Mobile App Development: A Performance and Usability Comparison*. Journal of Software Engineering, 45(2), 213-225.

Williams, B., & Thompson, S. (2023). *The Rise of Flutter: A New Era in Cross-Platform Development*. Software Development Trends, 39(3), 102-115.

Brown, C., & Clark, J. (2022). *Xamarin for Mobile Development: Pros and Cons*. Microsoft Developer Insights, 16(4), 89-101.

Smith, A., Johnson, L., & Green, M. (2022). *Native Mobile App Development: A Performance and Usability Comparison*. Journal of Software Engineering, 45(2), 213-225.

Williams, B., & Thompson, S. (2023). *The Rise of Flutter: A New Era in Cross-Platform Development*. Software Development Trends, 39(3), 102-115.

Statista. (2023). *Number of mobile apps available in leading app stores as of 2023*. Statista. Available at: https://www.statista.com/statistics/276759/number-of-apps-available-in-leading-app-stores/