

Comparative analysis of cross-platform development methodologies: a comprehensive study

Raiymbek Jangassiyev¹, Zhanat Umarova¹, Aisaule Ussenova¹, Zlikha Makhanova¹, Nurlybek Zhumatayev², Manat Amirov³, Gulzhan Koishibekova¹

¹Department of Information Systems and Modeling, Information Technology and Energetic Scientific School, Auezov University, Shymkent, Kazakhstan

²Department of Computing systems and Software, Information Technology and Energetic Scientific School, Auezov University, Shymkent, Kazakhstan

³Department of Information Communication Technologies, Information Technology and Energetic Scientific School, Auezov University, Shymkent, Kazakhstan

Article Info

Article history:

Received May 3, 2024

Revised Nov 6, 2024

Accepted Nov 26, 2024

Keywords:

.NET MAUI

Cross-platform development frameworks

Flutter

GUI

React Native

Xamarin

ABSTRACT

In an era marked by the proliferation of devices and operating systems, delivering native-feeling applications across platforms has become indispensable. This paper scrutinizes native development through the lens of cross-platform frameworks, investigating their merits, major contenders such as React Native, Flutter, Xamarin, and the nascent .NET MAUI, and their practical implementations. By dissecting the distinct strengths and considerations of each framework, we provide developers with insights to make judicious decisions commensurate with their requirements and proficiencies. This inquiry underscores how cross-platform frameworks empower developers to broaden their audience reach while upholding native performance standards, thereby shaping the trajectory of app development through sustained innovation and integration with emergent technologies.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Zhanat Umarova

Department of Information Systems and Modeling, Auezov University

Tauke khan Avenue, 5, Shymkent, Kazakhstan

Email: Zhanat-u@mail.ru

1. INTRODUCTION

In a rapidly evolving technological landscape, the demand for seamless and intuitive user experiences across a multitude of devices has never been greater. Gone are the days when mobile app development was solely focused on smartphones and occasional forays into tablet applications. Today, the spectrum of app-enabled devices spans from internet of things (IoT) gadgets to autonomous vehicles [1], underscoring the need for adaptable and versatile development solutions.

Traditionally, native app development has been the cornerstone of mobile application creation, offering unparalleled performance and user experience tailored to specific platforms. However, the native approach is not without its limitations. Chief among these is the inherent platform dependency of native app SDKs, which necessitates separate development efforts for each platform. For instance, iOS and Android SDKs are distinct entities, making it technically challenging to create a single app that seamlessly runs across multiple platforms [2].

In response to these challenges, the emergence of cross-platform development frameworks has revolutionized the app development landscape. These frameworks, such as React Native, Flutter, Xamarin, and the innovative .NET MAUI, offer developers a compelling alternative by bridging the gap between native performance and code recyclability. By enabling developers to write code once and deploy it across

various platforms, cross-platform frameworks empower them to sculpt unparalleled user experiences while maximizing development efficiency [3], [4].

This paper embarks on an in-depth exploration of cross-platform development using frameworks, delving into the strengths and considerations inherent in each approach. Through comparative analysis, real-world examples, and prognostication of future trends, we unravel the pivotal role of cross-platform frameworks in shaping the future of app development. Join us on this journey as we navigate the realm of native development using cross-platform frameworks, unlocking their potential to drive broader reach, superior performance, and continuous innovation in the ever-evolving landscape of app development [5].

a. Comparative analysis of app development types

Cross-platform development encompasses diverse methodologies aimed at creating applications that seamlessly operate across multiple operating systems. The selection of an appropriate approach is contingent upon specific project requirements, the proficiency of the development team, and the desired performance standards. This paper provides a comprehensive breakdown of the principal types of cross-platform development [6]. Table 1 presents a comparative analysis of various types of app development, including Native app development, Hybrid app development, Progressive Web apps (PWAs), Cloud-Based app development, and low-code/no-code platforms.

Table 1. Comparative analysis of app development types

Feature	Native app development	Hybrid app development	PWAs	Cloud-based app development	Low-code/no-code platforms
Performance	Near-native	Moderate	Lower	Native-like (depends on implementation)	Variable
User experience	Native-like	Native-like with potential limitations	Native-like browser experience	Depends on platform implementation	Variable
Development time	Faster due to code reuse	Faster initial development	Fastest	Moderate	Fastest
Development cost	Can be higher due to expertise needed	Lower	Lower	Variable	Lower
Code reusability	High	Moderate	Limited	High	Limited
Native feature access	High	Limited	Limited	Native-like (depends on platform implementation)	Limited
Offline functionality	Limited	Variable	Available	Yes, with caching	Variable
Platform support	All major platforms	All major platforms	Web browsers	All platforms	Most platforms
Development expertise	Requires familiarity with frameworks and native development	Moderate web development skills	Web development skills	Backend and cloud development skills	Limited coding experience

Table 1 compares the performance, user experience, development time, and other features of different types of app development. Native app development achieves near-native performance and user experience, but requires code reuse and expertise. Hybrid app development provides moderate performance and user experience with some limitations, and has a faster initial development time. Progressive web apps have lower performance and user experience that depend on their implementation, and are the fastest option to develop. Cloud-based app development has variable performance and user experience that depend on the platform, and has a moderate development time. Low-code/no-code platforms have variable performance and user experience, and offer the fastest development time. Table 1 can help developers choose a suitable framework for app development.

b. Survey of leading and emerging frameworks

Let us now embark on an exploration of specific exemplars that underscore the potency of cross-platform development facilitated by frameworks. React Native: endorsed by Facebook, React Native harnesses the prowess of JavaScript and React, thereby endowing a sense of familiarity for web developers alongside access to an expansive community. Noteworthy applications developed with React Native include Facebook, Instagram, and Bloomberg. Figure 1 architectures of Flutter and Xamarin; Figure 1(a) flowchart of React Native architecture and Figure 1(b) flowchart of Xamarin architecture shows a flowchart illustrating the basic overview of React Native architecture between native modules and JavaScript during application execution [7], [8].

Figure 1(a) illustrates the architecture shared by Flutter and Xamarin, consisting of three key sections: Native, Bridge, and JavaScript. In this structure, the Native segment manages platform-specific UI and events, while the Bridge facilitates seamless communication between Native and JavaScript components. The JavaScript section houses the business logic and React components, orchestrating event handling, data transfer, processing, and UI updates across the application [9], [10].

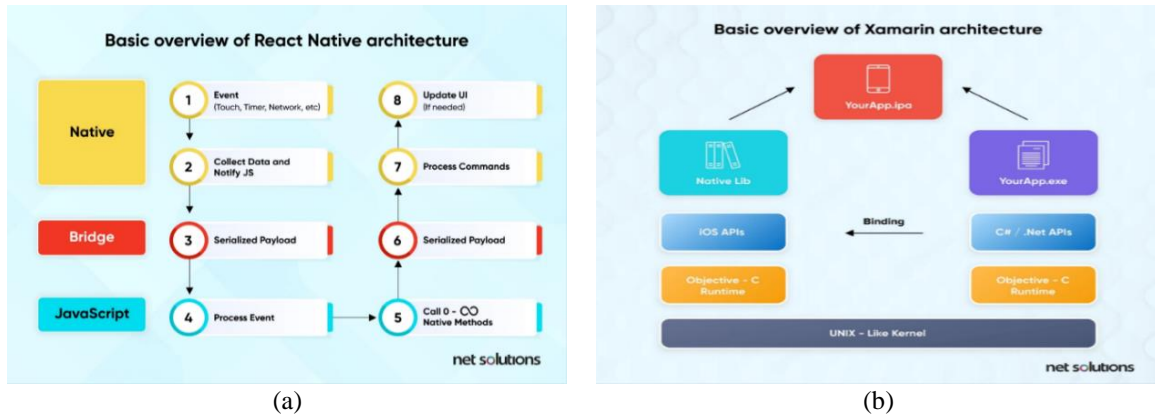


Figure 1. Architectures of Flutter and Xamarin: (a) flowchart of React Native architecture and (b) flowchart of Xamarin architecture

Flutter, developed by Google, utilizes Dart for rapid development and integrates hot reloading functionality, making it conducive for complex projects. Its extensive widget library and robust performance have garnered favor from notable companies like Alibaba, Reflectly, and Google Ads. Xamarin, owned by Microsoft, leverages C# and integrates seamlessly with Visual Studio, appealing particularly to .NET developers. Pinterest, Uber Eats, and Microsoft Office Mobile are among its distinguished users. The Xamarin architecture, depicted in Figure 1(b), showcases how it combines native libraries and C#/.NET APIs, allowing developers to write code in a single language while achieving near-native performance and accessing native features. .NET MAUI, spearheaded by Microsoft, utilizes C# and XAML to extend its reach across Android, iOS, macOS, and Windows through a unified codebase. Benefiting from the strengths of the .NET and Xamarin communities, .NET MAUI promises native-level performance, appealing to both existing Xamarin developers and new ventures alike [10].

When evaluating these frameworks, the importance of open-source considerations should not be underestimated, as reliance on third-party libraries can lead to technical debt and increased maintenance complexity. The quantity and quality of third-party components and packages may vary between frameworks, influencing long-term project viability [11], [12]. Figure 2 shows the usage of cross-platform mobile frameworks by software developers worldwide from 2019 to 2022 in a bar graph. The y-axis displays the percentage of developers using each framework, while the x-axis lists the different frameworks.

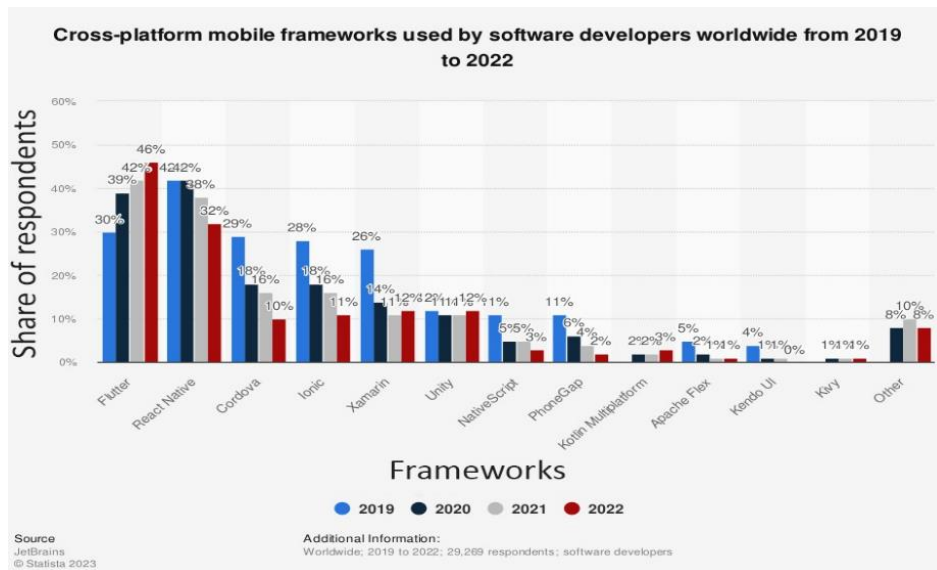


Figure 2. Usage according to Statista Developer Survey (Statista, 2024) [13]

These are the main observations from Figure 3, Flutter usage has consistently increased from 30% in 2019 to 46% in 2022. React Native has also increased, but not as significantly, from 42% in 2019 to 48% in 2022. Cordova and Ionic have declined over the years. Xamarin has remained relatively stable with slight fluctuations, while Unity, Native Script, and other less common frameworks have lower percentages but are still included in the data. It is imperative to recognize that the selection of the “optimal” framework hinges upon the precise requisites of your project, the proficiencies of your team, and the target platforms. It behooves one to meticulously scrutinize each option, evaluating its strengths, weaknesses, and the support it garners from the community, prior to arriving at an informed decision.

2. METHOD

2.1. Comparative analysis of frameworks. utilization and performance assesment of cross-platform frameworks

Identifying the ideal framework for your project hinges on its unique requirements, but understanding usage trends and performance can provide valuable insight. Here’s a comparative analysis of the discussed frameworks based on available data:

a. Utilization metrics:

- React Native: with the largest developer community and ecosystem, React Native leads in popularity (42.1% utilization rate according to Statista’s Developer Survey 2023), offering abundant resources and strong community support.
- Flutter: despite being newer, Flutter has rapidly gained traction, especially in enterprise adoption, thanks to its feature-rich widget library and hot reloading functionality. Google Trends shows a rising interest in Flutter compared to React Native.
- Xamarin: while still popular, particularly among .NET developers, Xamarin’s growth lags behind Flutter. However, its robust toolset and integration with Visual Studio remain attractive.
- .NET MAUI: as a newcomer, comprehensive usage statistics are still emerging. However, its potential within the .NET community and focus on native performance bode well for its future.

b. Performance evaluations:

Comparative assessments like the 2023 State of JavaScript report highlight performance differences, but these are often minor for typical applications. Framework optimizations aim to further minimize these differences over time.

Figure 3 shows a horizontal bar graph titled ‘PERFORMANCE BENCHMARKS’ sourced from [HTTPS://JSPERF.APP/](https://jsperf.app/). It compares the start times (in seconds) of applications built with three different frameworks: Native, Flutter, and React Native on Android devices. Native apps start the fastest (0.6 s), followed by Flutter (0.8 s) and React Native (1.2 s). This shows that Native apps launch faster than the other two frameworks.

Figure 4 shows a horizontal bar graph representing the steady-state performance in frames per second (FPS) of three different mobile application development frameworks: Native, Flutter, and React Native. The FPS is averaged across devices. Native apps have the highest FPS (63), followed by Flutter (62) and React Native (60). This shows that Native apps perform slightly better than Flutter and significantly better than React Native.

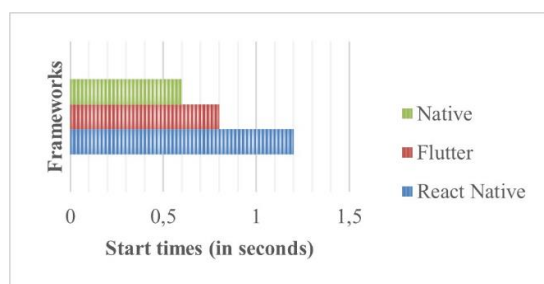


Figure 3. Start time comparison of app development frameworks (jsperf.app) [14]

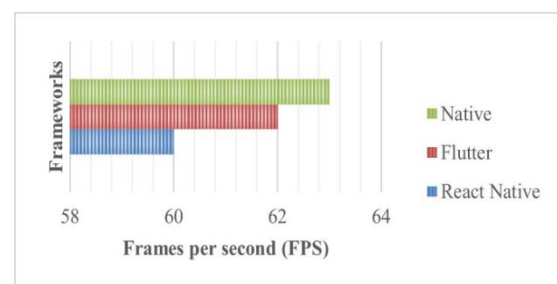


Figure 4. Steady-state performance of app development frameworks (jsperf.app) [14]

Performance profiles may vary depending on the specific usage scenarios and device configurations. Significant differences can be observed in benchmarks such as cold start times, with Flutter demonstrating faster initialization, although it may be slightly slower in stable operational scenarios compared to React Native [15]. In terms of development steps, native and Xamarin.iOS are very close. Features can be implemented similarly

in both, and UI design is facilitated by a what you see is what you get (WYSIWYG) visual tool [16]. Table 2 presents a comparative analysis of four mobile app development frameworks: React Native, Flutter, Xamarin, and .NET MAUI.

Table 2. Battle of the frameworks: a comparative study

Feature	React Native	Flutter	Xamarin	.NET MAUI (NEW)
Usage rank	1	2	3	Not established
Developer community	Largest	Large & growing	Established	Large .NET community
Key languages	JavaScript	Dart	C#	C# & XAML
Performance compared to Native	Near Native	Near Native	Near Native	Potential for high performance
Popular apps	Facebook, Instagram, Bloomberg	Alibaba, Reflectly, Google Ads	Pinterest, Uber Eats, Microsoft Office Mobile	New - No Major Apps Yet

In Table 2, React Native is ranked first in terms of usage, followed by Flutter in second place, and Xamarin in third place. .NET MAUI has not yet established a usage rank. React Native has the largest developer community, while Flutter has a large and growing community. Xamarin has an established community, while .NET MAUI is a newer technology with a developing community. It is worth noting that these technologies are supported by the large .NET community. React Native uses JavaScript, Flutter uses Dart, Xamarin uses C#, and .NET MAUI uses both C# and XAML.

When comparing CPU time usage by application, Flutter technology outperformed React Native. The average usage was around 97%. The React Native technology used about 130% of the CPU time (i.e., it used more than one core) and the total amount of memory used by the application for the Flutter software framework varied between 50 MB and 120 MB. For an application developed using the React Native development framework, the value of this memory fluctuated around 75 MB [16].

3. RESULTS AND DISCUSSION

3.1. Performance analysis of .NET MAUI and Xamarin frameworks

The .NET MAUI platform enables developers to create mobile and desktop applications using a single interface. It offers ample opportunities to organize the structure and select user interface controls. Separating the GUI definition from the program logic allows for more efficient development and maintenance of applications. Additionally, MAUI's deep integration with other .NET tools and services ensures high application performance.

Figure 5 is an implementation of the Game of Life for .NET Multi-platform App UI (.NET MAUI). Life is a cellular automaton invented by mathematician John Conway in 1970 and popularized in Scientific American [17], [18]. Using the information provided in Figure 5, this analysis examines the performance of the .NET MAUI framework. The memory usage remains stable at approximately 230 MB, indicating that the Game of Life program does not consume excessive memory and is well-optimized in this aspect.

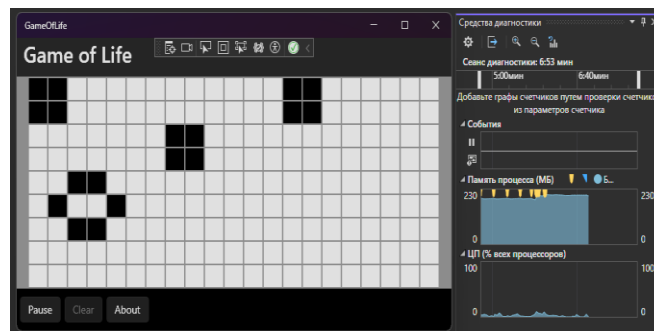


Figure 5. Debugging the Game of Life on Windows (.NET MAUI)

The CPU usage shows minimal spikes, suggesting that the .NET MAUI framework handles processing efficiently, ensuring smooth performance without overburdening the CPU. In conclusion, the Game of Life program demonstrates that the .NET MAUI framework utilizes memory and CPU resources

efficiently. This suggests that .NET MAUI could be a reliable choice for developers seeking optimized performance from their applications. To enhance the study's informativeness, the application was executed on the Android platform. Figure 6 illustrates the performance demonstration.

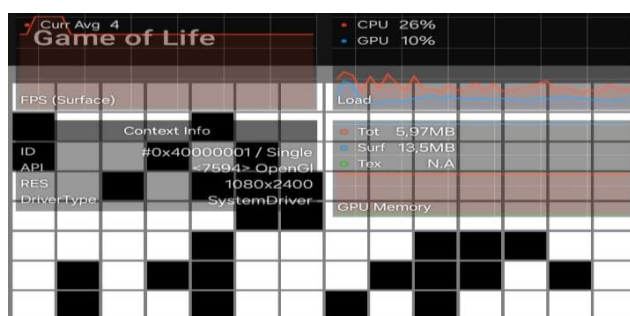


Figure 6. Debugging the Game of Life on Android (.NET MAUI)

This analysis presents the performance of the .NET MAUI framework on the Samsung Galaxy S21+ device, based on the information provided in Figure 5. The CPU usage is relatively low, at 26%, suggesting that the Game of Life program is not overly taxing on the device's processing capabilities when run on the .NET MAUI framework. The GPU usage is also low, at 10%, indicating that the application is not demanding in terms of graphical processing. This could be due to the efficient rendering or minimal graphical content in the application. The application uses only 5.97MB of memory, which is relatively low and indicates that it is not consuming excessive resources. This suggests that the .NET MAUI framework is an efficient option for developers who want to optimize their application's performance on mobile devices. However, it is important to note that performance may be influenced by other factors, such as code efficiency and overall system configuration. Therefore, a comprehensive performance analysis should take these factors into account. Figure 7 is an implementation of the Game of Life for Xamarin.Forms [19].

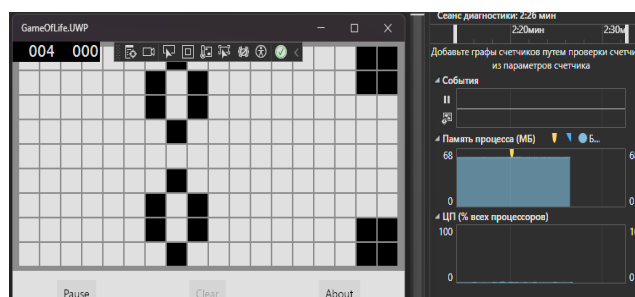


Figure 7. Debugging the Game of Life on Windows (Xamarin)

Based on the data presented in Figure 7, the performance of the Xamarin framework can be analyzed as follows: The program consumes 68 MB of memory, indicating that it is well-optimized and does not consume excessive memory; The CPU usage is 0%, indicating that the Xamarin framework handles processing efficiently, ensuring smooth performance without overburdening the CPU.

The Game of Life program demonstrates that the Xamarin framework uses memory and CPU efficiently, indicating that it could be a dependable option for developers seeking optimized application performance. However, a more comprehensive analysis is required to compare Xamarin with other development tools. When comparing the memory and CPU usage of the same program on a Ryzen 9 5900X processor, it was observed that the .NET MAUI framework uses approximately 230 MB of memory and has minimal CPU usage. On the S21+ device, the memory usage is around 5.97 MB. Therefore, the selection between Xamarin and .NET MAUI may depend on the specific requirements of the application and the hardware on which it is intended to run. Performance analysis should consider factors such as code efficiency and system configuration, as they can influence performance. To improve the study's comprehensibility, we executed the application on the Android platform. The performance demonstration is illustrated in Figure 8.

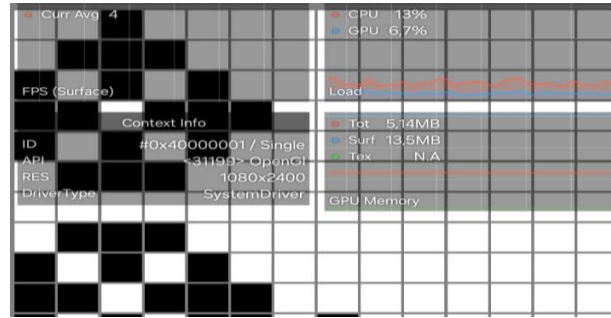


Figure 8. Debugging the Game of Life on Android (Xamarin)

Figure 8 presents data on the performance of the Xamarin framework on the Samsung Galaxy S21+ device. The CPU usage was found to be relatively low at 13%, indicating that the Game of Life program does not heavily tax the device’s processing capabilities when run on the Xamarin framework. The GPU usage was at 6.7%, suggesting that the application is not graphically demanding. This may be a result of efficient rendering or minimal graphical content in the application. The application’s total memory usage is 5.14 MB, which is relatively low and indicates that it is not consuming excessive resources. This is a positive sign of optimization. Table 3 displays the test results for the Samsung Galaxy S21+ device.

Table 3. Comparison of resource usage in .NET MAUI and XAMARIN frameworks

Framework	CPU usage (%)	GPU usage (%)	GPU memory (MB)
.NET MAUI	26	10	5.97
Xamarin	13	6.7	5.14

This text compares the performance of the Game of Life program on the .NET MAUI and Xamarin frameworks using data from Table 3. The CPU usage of .NET MAUI is higher (26%) than that of Xamarin (13%), which suggests that .NET MAUI is performing more computations or processes in the background. The GPU usage is not mentioned in this fragment of text. However, it is worth noting that the GPU usage of .NET MAUI is higher (10%) than that of Xamarin (6.7%), which may be due to more intensive graphical rendering tasks. Additionally, the memory usage of .NET MAUI is slightly higher (5.97 MB) than that of Xamarin (5.14 MB), indicating that .NET MAUI may be using more resources for its operations [20]-[30].

Mobile applications are vital for our daily lives, as they offer various functions. With the growing number of applications, it is crucial to build successful, trouble-free, and easy-to-use applications [31]-[40]. Cross-platform development frameworks enable real-world applications to achieve this goal. Table 4 shows some examples of renowned apps built with each framework, and their capabilities across diverse domains.

Table 4. Examples and capabilities of applications built with various frameworks

Framework	Example apps	Capabilities
React Native	Facebook, Instagram, Bloomberg	Scalability, user interaction, engagement, dynamic user experience, real-time data and news, uniform and responsive interface
Flutter	Alibaba, Reflectly, Google Ads	E-commerce mechanisms, seamless animations, captivating visual designs, interactive and data-centric experiences
Xamarin	Uber Eats, Pinterest, Microsoft Office Mobile	Location-centric services, mobile commerce, visually-arresting and immersive experience, productivity tools, enterprise-grade applications
.NET MAUI	N/A	Native performance, migration from Xamarin, potential within .NET developer community

Finally, this analysis has evaluated four frameworks for app development: React Native, Flutter, Xamarin, and .NET MAUI [41]-[51]. Table 4 illustrates their differences in terms of performance, user experience, development time, and other features, using examples of well-known apps built with each framework. The table also demonstrates that each framework has its advantages and disadvantages, and that there is no ideal solution for app development. Developers should select a framework that meets their requirements and preferences. Generally speaking, “that’s the only way we know how”, “that’s the cheapest way”, or “I don’t really know” should be considered with caution. Further studies or practices can explore new trends and developments in app development frameworks and their impact on the app industry and users. Both the .NET MAUI and Xamarin frameworks demonstrate efficient resource usage. However, it

appears that .NET MAUI may have higher resource usage compared to Xamarin, possibly due to additional features or optimizations, such as support for dark themes. The decision to use either .NET MAUI or Xamarin should be based on the application's specific requirements and the intended hardware. It is worth noting that both frameworks are part of the .NET ecosystem and share many features and capabilities, which could facilitate transitioning between them if needed [52]-[55].

4. CONCLUSION

The horizon of cross-platform development shines with promise. Offering boundless opportunities to craft exemplary user experiences across a myriad of platforms. Through the assimilation of novel technologies, cultivation of community collaboration, and unwavering emphasis on performance, security, and scalability. Developers stand poised to harness the transformative potential of cross-platform frameworks, ushering in the dawn of innovative applications.

Our exploration into the .NET MAUI, Xamarin, React Native, and Flutter frameworks has provided valuable insights into their performance characteristics. All these frameworks exhibit efficient memory and CPU usage, with .NET MAUI showing more optimization for mobile devices. The frame rate of 4 FPS for both .NET MAUI and Xamarin frameworks, as observed in the Game of Life program, is by design and does not indicate performance issues.

In today's interconnected realm, the imperative of engaging diverse audiences across multiple platforms underscores the quintessence of app success. Cross-platform development, epitomized by frameworks such as React Native, Flutter, Xamarin, and the fledgling .NET MAUI, emerges as a compelling solution, empowering developers to manifest the ethos of "write once, run anywhere" with a semblance of near-native performance and efficacy.

Our exploration traversed the myriad avenues of cross-platform development, delineating the merits and nuances of each approach. We ventured into the realms of popular frameworks, illuminating their unique strengths, utilization metrics, performance evaluations, and real-world triumphs witnessed in the likes of Facebook, Alibaba, and Uber Eats. Furthermore, we cast a gaze into the vista of tomorrow, envisioning strides in performance optimization, integration with nascent technologies such as artificial intelligence (AI) and IoT, and the perpetual democratization of development through the refinement of tools. The addition of our findings from the .NET MAUI, Xamarin, React Native, and Flutter frameworks enriches this exploration. Understanding the contemporary landscape, the latent potential of each framework, and the exhilarating vistas on the horizon equips developers to orchestrate judicious decisions and harness this technology to fashion high-caliber, adaptable applications that resonate with broader audiences and engender superlative user experiences.

Cross-platform development transcends the realm of transient trends; it emerges as an omnipotent tool sculpting the trajectory of app creation. As we move forward, the choice between .NET MAUI, Xamarin, React Native, and Flutter will depend on the specific requirements of the application and the hardware it is intended to run on. All these frameworks, being part of the broader ecosystem of cross-platform development, share many features and capabilities, which could make transitioning between them easier if necessary. This flexibility and adaptability are what make cross-platform development a powerful tool in the hands of developers.

REFERENCES




- [1] C. Rieger and T. A. Majchrzak, "Towards the definitive evaluation framework for cross-platform app development approaches," *Journal of Systems and Software*, vol. 153, pp. 175–199, 2019, doi: 10.1016/j.jss.2019.04.001.
- [2] M. Q. Huynh, P. Ghimire, and D. Truong, "Hybrid App Approach: Could It Mark the End of Native App Domination?," *Issues in Informing Science and Information Technology*, vol. 14, pp. 049–065, 2017, doi: 10.28945/3723.
- [3] T. A. Majchrzak, A. Bjørn-Hansen, and T.-M. Grønli, "Progressive Web Apps: The Definite Approach to Cross-Platform Development?" *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2018, doi: 10.24251/hicss.2018.718
- [4] A. Y. Pchelkin and N. F. Gusarova, "Cross-Platform Development Based on Web Technologies to Support Solutions in Problem-Oriented Management Systems," *Economics Law Innovation*, 1, pp. 41–47, 2022, doi: 10.17586/2713-1874-2022-1-41-47
- [5] T. Guo, "Cloud-Based or On-Device: An Empirical Study of Mobile Deep Inference," *2018 IEEE International Conference on Cloud Engineering (IC2E)*, Orlando, FL, USA, 2018, pp. 184–190, doi: 10.1109/IC2E.2018.00042.
- [6] A. C. Bock and U. Frank, "Low-Code Platform," *Business & Information Systems Engineering*, vol. 63, no. 6, pp. 733–740, 2021, doi: 10.1007/s12599-021-00726-8
- [7] A. Manchanda, "The Ultimate Guide to Cross Platform App Development Frameworks in 2024," Net Solutions, <https://www.netsolutions.com/insights/cross-platform-app-frameworks-in-2019/>, (Accessed September 7, 2023).
- [8] Microsoft. (2024). Xamarin - Build cross-platform apps with .NET. <https://dotnet.microsoft.com/en-us/apps/xamarin>.
- [9] K. Shah, H. Sinha and P. Mishra, "Analysis of Cross-Platform Mobile App Development Tools," *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, Bombay, India, 2019, pp. 1-7, doi: 10.1109/I2CT45611.2019.9033872.
- [10] T. Majchrzak and T.-M. Grønli, "Comprehensive Analysis of Innovative Cross-Platform App Development Frameworks," *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2017, doi: 10.24251/hicss.2017.745.

- [11] Statista, "Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2022," 2024, Available: <https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/>.
- [12] M. Kovács and Z.C. Johanyák, "Comparative Analysis of Native and Cross-Platform iOS Application Development," *Műszaki Tudományos Közlemények*, vol. 15, no. 1, pp. 61–64, 2021, doi: 10.33894/mtk-2021.15.12.
- [13] M. Markowski and J. Smolka, "A comparative analysis of the Flutter and React Native frameworks," *Journal of Computer Sciences Institute*, vol. 29, 346–351, 2023, doi: 10.35784/jcsi.3794.
- [14] I. V. Ponomarev, "Features of the .NET MAUI framework for creating a cross-platform applications," *System Technologies*, vol. 1, no. 144, pp. 51–57, 2023, doi: 10.34185/1562-9945-1-144-2023-07.
- [15] Microsoft. (2024, February 13). .NET MAUI - Game of Life [Code Sample]. [Online]. Available: <https://learn.microsoft.com/ru-ru/samples/dotnet/maui-samples/apps-gameoflife/>.
- [16] P. R. Hiwale, A. A. Kalsait, K. Y. Choukade, A. S. Puri, and P. V. Shirbhate, "Review On Cross-Platform Mobile Application Development," *International Journal for Research in Applied Science and Engineering Technology*, vol. 10, no. 1, pp. 1433–1439, 2022, doi: 10.22214/ijraset.2022.40004.
- [17] Microsoft. (2024, February 16). What is .NET MAUI? <https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-7.0>.
- [18] J. White, "Going native (or not): Five questions to ask mobile application developers," *Australasian Medical Journal*, vol. 6, no. 1, pp. 7–14, 2013, doi: 10.4066/amj.2013.1576.
- [19] I. H. Sarker, M. M. Hoque, Md. K. Uddin, and T. Alsanoosy, "Mobile Data Science and Intelligent Apps: Concepts, AI-Based Modeling and Research Directions," *Mobile Networks and Applications*, vol. 26, no. 1, pp. 285–303, 2020, doi: 10.1007/s11036-020-01650-z.
- [20] Y. K. Dwivedi *et al.*, "Metaverse beyond the hype: Multidisciplinary perspectives on emerging challenges, opportunities, and agenda for research, practice and policy," *International Journal of Information Management*, vol. 66, 2022, doi: 10.1016/j.ijinfomgt.2022.102542.
- [21] X. Qiao, P. Ren, S. Dustdar, and L. Liu, "Web AR: A Promising Future for Mobile Augmented Reality—State of the Art, Challenges, and Insights," in *Proceedings of the IEEE*, vol. 107, no. 4, pp. 651–666, April 2019, doi: 10.1109/JPROC.2019.2895105.
- [22] S. Xanthopoulos and S. Xinogalos, "A Comparative Analysis of Cross-platform Development Approaches for Mobile Applications," *ACM International Conference Proceeding Series*, 2013, pp. 213–220, doi: 10.1145/2490257.2490292.
- [23] D. You and M. Hu, "A Comparative Study of Cross-platform Mobile Application Development," *12th Annual Conference of Computing and Information Technology Research and Education*, New Zealand, 2021.
- [24] A. Bjørn-Hansen, T.-M. Grønli, G. Ghinea, and S. Alouneh, "An Empirical Study of Cross-Platform Mobile Development in Industry," *Wireless Communications and Mobile Computing*, pp. 1–12, 2019, doi: 10.1155/2019/5743892.
- [25] A. Ahmad, K. Li, C. Feng, A. Syed, A. Yousif, and S. Ge, "An Empirical Study of Investigating Mobile Applications Development Challenges," in *IEEE Access*, vol. 6, pp. 17711–17728, 2018, doi: 10.1109/ACCESS.2018.2818724.
- [26] M. Martinez, "Two Datasets of Questions and Answers for Studying the Development of Cross-Platform Mobile Applications using Xamarin Framework," *2019 IEEE/ACM 6th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, Montreal, QC, Canada, 2019, pp. 162–173, doi: 10.1109/MOBIRESOFT.2019.00032.
- [27] M. Abdal, T. Mohamed, S. Jan, F. Khan, and A. Khattak, "A Comparative Analysis of Mobile Application Development Approaches," *Proceedings of the Pakistan Academy of Sciences*, vol. 58, pp. 35–45, 2021, doi: 10.53560/PPASA(58-1)717.
- [28] M. Işitan and M. Koklu, "Comparison and Evaluation of Cross Platform Mobile Application Development Tools," *International Journal of Applied Mathematics Electronics and Computers*, vol. 8, no. 4, pp. 273–281, 2020, doi: 10.18100/ijamec.832673.
- [29] C. Ferreira *et al.*, "An Evaluation of Cross-Platform Frameworks for Multimedia Mobile Applications Development," in *IEEE Latin America Transactions*, vol. 16, no. 4, pp. 1206–1212, Apr. 2018, doi: 10.1109/TLA.2018.8362158.
- [30] P. Nawrocki, K. Wrona, M. Marczak, and B. Śnieżyński, "A Comparison of Native and Cross-Platform Frameworks for Mobile Applications," *Computer*, vol. 54, pp. 18–27, 2021, doi: 10.1109/MC.2020.2983893.
- [31] L. P. Barros, F. Medeiros, E. Moraes, and A. F. Júnior, "Analyzing the Performance of Apps Developed by using Cross-Platform and Native Technologies," *International Conference on Software Engineering and Knowledge Engineering*, 2020, doi: 10.18293/SEKE2020-122.
- [32] A. Javed, "Performance Optimization Techniques for ReactJS," *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, Coimbatore, India, 2019, pp. 1–5, doi: 10.1109/ICECCT.2019.8869134.
- [33] X. Jia, A. Ebone, and T. Yongshan, "A performance evaluation of cross-platform mobile application development approaches" *MOBILESoft '18: Proceedings of the 5th International Conference on Mobile Software Engineering and Systems*, 2018, pp. 92–93, doi: 10.1145/3197231.3197252.
- [34] H. Zahra and S. Samer, "A Systematic Comparison Between Flutter and React Native from Automation Testing Perspective," *2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, Ankara, Turkey, 2022, pp. 6–12, doi: 10.1109/ISMSIT56059.2022.9932749.
- [35] T. Zohud and S. Zein, "Cross-Platform Mobile App Development in Industry: A Multiple Case-Study," *International Journal of Computing*, pp. 46–54, 2012, doi: 10.47839/ijc.20.1.2091.
- [36] H. V. Gamido and M. V. Gamido, "Comparative Review of the Features of Automated Software Testing Tools," *International Journal of Electrical and Computer Engineering*, vol. 9, no. 5, pp. 4473–4478, 2019, doi: 10.11591/ijece.v9i5.pp4473-4478.
- [37] P. Meirelles, C. Rocha, F. Assis, R. Siqueira, and A. Goldman, "A Students' Perspective of Native and Cross-Platform Approaches for Mobile Application Development," *Computational Science and Its Applications – ICCSA*, pp. 586–601, 2019, doi: 10.1007/978-3-030-24308-1_47.
- [38] M. Mahendra and B. Anggorojati, "Evaluating the performance of Android based Cross-Platform App Development Frameworks," *ICCIP '20: Proceedings of the 6th International Conference on Communication and Information Processing*, 2020, pp. 32–37, doi: 10.1145/3442555.3442561.
- [39] P. K. Aggarwal, P. S. Grover, and L. Ahuja, "A Performance Evaluation Model for Mobile Applications," *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*, Ghaziabad, India, 2019, pp. 1–3, doi: 10.1109/IoT-SIU.2019.8777497.
- [40] D. T. Bui *et al.*, "New Hybrids of ANFIS with Several Optimization Algorithms for Flood Susceptibility Modeling," *Water*, vol. 10, no. 9, 2019, doi: 10.3390/w10091210.
- [41] I. C. Morgado and A. C. R. Paiva, "The iMPAcT Tool for Android Testing," *Proceedings of the ACM on Human-Computer Interaction*, vol. 3, pp. 1–23, 2019, doi: 10.1145/3300963.




- [42] C. M. Pinto and C. Coutinho, "From Native to Cross-platform Hybrid Development," *2018 International Conference on Intelligent Systems (IS)*, Funchal, Portugal, 2018, pp. 669-676, doi: 10.1109/IS.2018.8710545.
- [43] A. Alkhalifah, "Predicting Mobile Cross-Platform Adaptation Using a Hybrid Sem-ANN Approach," *Computer Systems Science and Engineering*, vol. 42, no. 2, pp. 639-658, 2022, doi: 10.32604/csse.2022.022519.
- [44] K. Majrashi, M. Hamilton, A. L. Uitdenbogerd, and S. Al-Megren, "Cross-Platform Usability Model Evaluation," *Multimodal Technologies and Interaction*, vol. 4, no. 4, 2022, doi: 10.3390/mti4040080.
- [45] C. Rieger and T. A. Majchrzak, "Towards the Definitive Evaluation Framework for Cross-Platform App Development Approaches," *Journal of Systems and Software*, vol. 153, pp. 175-199, 2019, doi: 10.1016/j.jss.2019.04.001.
- [46] W. Niemiec, R. Borges, and E. Cota, "Mobilex: a generic framework for cross-platform mobile development based on web language," *SBES '22: Proceedings of the XXXVI Brazilian Symposium on Software Engineering*, pp. 347-352, 2022, doi: 10.1145/3555228.3555274.
- [47] Y. Cheon and C. Chavez, "Converting Android Native Apps to Flutter Cross-Platform Apps," *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, NV, USA, 2021, pp. 1898-1904, doi: 10.1109/CSCI54926.2021.00355.
- [48] F. Brudy *et al.*, "Cross-Device Taxonomy: Survey, Opportunities and Challenges of Interactions Spanning Across Multiple Devices," *CHI '19: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1-28, 2019, doi: 10.1145/3290605.3300792.
- [49] A. S. A. Alwabel and X.-J. Zeng, "Data-driven modeling of technology acceptance: A machine learning perspective," *Expert Systems with Applications*, vol. 185, p. 115584, 2021, doi: 10.1016/j.eswa.2021.115584.
- [50] P. Kr. Chopdar, N. Korfiatis, V. J. Sivakumar, and M. D. Lytras, "Mobile shopping apps adoption and perceived risks: A cross-country perspective utilizing the Unified Theory of Acceptance and Use of Technology," *Computers in Human Behavior*, vol. 86, pp. 109-128, 2018, doi: 10.1016/j.chb.2018.04.017.
- [51] C. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning, with Applications in R*, in Springer Texts in Statistics (STS), Latest edition, Springer, 2021, doi: 10.1007/978-1-0716-1418-1.
- [52] K. Sohn and O. Kwon, "Technology acceptance theories and factors influencing artificial Intelligence-based intelligent products," *Telematics and Informatics*, vol. 47, p. 101324, 2020, doi: 10.1016/j.tele.2019.101324.
- [53] Z. Kalinić, V. Marinković, L. Kalinić, and F. Liébana-Cabanillas, "Neural Network Modeling of Consumer Satisfaction in Mobile Commerce: An Empirical Analysis," *Expert Systems with Applications*, vol. 175, p. 114803, 2021, doi: 10.1016/j.eswa.2021.114803.
- [54] H. Rafique, A. O. Almagrabi, A. Shamim, F. Anwar, and A. K. Bashir, "Investigating the Acceptance of Mobile Library Applications with an Extended Technology Acceptance Model (TAM)," *Computers & Education*, vol. 145, p. 103732, 2020, doi: 10.1016/j.compedu.2019.103732.
- [55] M. Ahmad, "Analysis of cross platform mobile application development frameworks," Ph.D. thesis, Institute of Applied Computer Systems, Riga Technical University, Rīga, Latvia, 2023.

BIOGRAPHIES OF AUTHORS






Raiymbek Jangassiyev    received the bachelor of technic and technologies degree in Information Systems in 2023. Currently he is a master's student in the Department of Information Systems and Modeling at Auezov University. His research interests include information systems, web programming, mobile applications, and IoT. He can be contacted at email: jangasiev@gmail.com.






Zhanat Umarova    received her Ph.D. degree in Informatics, Computer Engineering and Control from Ministry of Education and Science, in 2013. Currently she works as an Associated Professor of Information Systems and Modeling Department at M. Auezov South Kazakhstan University. She has supervised and co-supervised more than 20 masters' students. She has authored or coauthored more than 100 publications. Her research interests include mathematical modeling, computer simulation, information security and data protection in information systems, and mobile technologies. She can be contacted at email: Zhanat-u@mail.ru.






Aisaule Usenova    received a bachelor's degree in Mathematics and teaching in 1988. Currently, he works as a senior lecturer at the Department of Information Systems and Modeling at the M. Auezov South Kazakhstan University. She is the author or co-author of more than 200 publications. Her research interests include mathematical modeling, computer simulation, and information communication technologies, mobile technologies. She can be contacted at email: Ais_usen@mail.ru.






Zlikha Makhanova    received the degree of Candidate of Pedagogical Sciences by the decision of the Committee of the Knowledge and Science Industry of the Republic of Kazakhstan on June 14, 2011. Currently she works as an Associated Professor of Information Systems and Modeling Department at the Auezov University. She has authored or coauthored more than 160 publications. Her research interests include mathematical modeling, information communication technologies, mobile technologies, and web technologies. She can be contacted at email: zlikha70@bk.ru.






Nurlybek Zhumatayev    received a Ph.D. in Computer Science, Computer Engineering and Management in 2012. Currently, he works as an associate professor of the Department of Computer Engineering and Software at the Auezov University. He is the author or co-author of more than 75 publications. His research interests include computer graphics, computer modeling, information security, and data protection. He can be contacted at email: nuralmiras@mail.ru.



Manat Amirov    received his master degree in Computer Science in 2020. Currently, he works as a senior lecturer of the Department of Information and Communication Technologies at the Auezov University. He is the author or co-author of more than 25 publications. His research interests include computer science, and information communication technologies. He can be contacted at email: manat_amirov@mail.ru.



Gulzhan Koishibekova    received her master degree in Computer Science in 2016. Currently, she works as a lecturer of the Department of Information Systems and Modeling at the Auezov University. She is the author or co-author of more than 25 publications. Her research interests include computer science, information systems, and mobile technologies. She can be contacted at email: koyshybekovag@mail.ru.