MITRAIS
MEMBER OF CAC HOLDINGS GROUP

continuous commitment

Flutter: A Comparative Analysis

Flutter VS Xamarin VS React Native VS Ionic VS Cordova VS Kotlin Analytics

FEB 2022

VER.0.1

Table of Contents

1. Introduction

Flutter is a cross-platform application development toolkit developed by Google. The popularity of Flutter is imminent as it enables faster development, with a flexible UI and customization, including the creation of natively compiled applications across platforms such as mobile, web, desktop, and embedded devices with the help of a single codebase. Flutter applications are developed using Dart, which is a simple object-oriented programming language. Currently, Google uses Flutter for some of its most popular applications, namely Google Assistant, Google Home Hub UI, and much more. Nearly 50,000 Flutter-based applications are available in Google Play Store, and the numbers are increasing at a staggering rate. In addition, BMW, eBay, Alibaba Group, Capital One are some popular companies that use Flutter to give their web applications an appealing look and provide an excellent user experience.

The significance of cross-platform app development is gaining traction, and the reason behind its rising demand is faster development and quicker time-to-market, reduced costs, and the use of a single codebase for various platforms. As a result, several organizations are opting for cross-platform development frameworks and technologies, but the debate over the appropriate choice of framework continues to prevail. In recent years, various frameworks and technologies have been at the forefront of application development, but each consists of varying attributes that make them differ from one another. Therefore, evaluating the application development frameworks requires a developer's perspective to understand the effectiveness of the frameworks.

From a developer's standpoint, the critical criteria for the evaluation include:

1. License Cost
2. Long-term Feasibility
3. Extensibility
4. Learning Success
5. Documentation and Support
6. Development Effort
7. Maintainability

This paper provides a comparative analysis of how Flutter performs against some of the existing application development frameworks, emphasizing the key considerations highlighted from a developer's perspective.

2. Evaluated Platforms

No.	Platform
1.	Flutter
2.	Xamarin
3.	React Native
4.	Ionic
5.	Cordova
6.	Kotlin

Table 1 – Evaluated Platforms

3. Other Platforms

Other platforms, which are not evaluated but may be included in the future are listed below:

- N/A

4. Evaluation Criteria

The evaluation of app development frameworks is best proposed by one of the frequently cited papers on evaluation criteria by Heitkotter et al. (2013). The overview of these criteria can be understood comprehensively from the sections hereunder.

1. License Cost
   There are various costs incurred to develop and publish commercial applications using a framework. An ideal case is using an open-source framework, but it tends to be complicated for commercial and closed-source projects. Moreover, some of the licenses have a severe impact on cross-platform development. It is widely known that different licenses allow different things, which provide various restrictions on licensed projects and distribution. Therefore, the direct costs are accrued for purchasing licenses, renewal, and additional tools.

2. Long-term Feasibility
   The investment decisions on an appropriate framework are crucial as restrictions on how the source code of the applications can be used are tied to a framework. From a developer's perspective, the preferred framework must be available for the long term. In addition, a framework requires continuous updates as web technologies and browsers are rapidly evolving. Therefore, long-term feasibility is determined by popularity, updates, and the development team. For example, popularity indicates that the developers' community recognizes the framework due to possible short update cycles and bug fixes.

3. Extensibility
   The term extensibility is associated with the evolution of the framework according to the changing environment. Therefore, extensibility is crucial for developers in terms of additional functionality and plugin mechanisms to provide stability and increase its usefulness.

4. Learning Success
   The time and effort required to understand and implement a framework for development affect the suitability for commercial uses. In addition, the ease of comprehensibility to learn new concepts at the initial process determines adaptability.

5. Document and Support
   For a successful framework, the documentation highlighting the necessary guidelines, functionalities, APIs for developers to master a framework is essential. Therefore, good quality documentation with additional resources and tutorials for reference is critical to understand the usability of a framework.

6. Development Effort
   The development efforts determine the overall cost of an application. From familiarizing with the framework to implementing applications, the developers require easy syntax, reusability of codes, and additional support, alongside debugging functionalities.

7. Maintainability
   Web applications require regular updates to provide more stability to applications. Therefore, this indicator is inter-related with the comprehensibility of codes that can allow developers to maintain the performance of the applications over a longer time frame.

5. Evaluating Application Development Frameworks

5.1.1 Flutter

No.	Criteria	Evaluation
1.	License Cost	Cost-effective.
2.	Long-Term Feasibility	More popular and has more scope in the future. In addition, there is good support for new plugins
3.	Extensibility	Flutter is designed in the form of an extensible, layered system. Moreover, the widgets are centric with the potential to evolve into a robust UI creation environment. The hot reload feature is a bonus for developers with various customizations.
4.	Learning Success	Easier learning curve.
5.	Document and Support	There is clear and precise documentation with continuous growth in the developers' community.
6.	Development Effort	Reduced code development time with various features and widgets. Besides, it is easier for developers to learn the framework.
7.	Maintainability	The framework is evolving rapidly; therefore, maintenance is challenging in the long run.

Table 2 – Flutter

5.1.2 Xamarin

No.	Criteria	Evaluation
1.	License Cost	Expensive to purchase Microsoft Visual Studio IDE for commercial-scale development.
2.	Long-Term Feasibility	Better development tools for faster development, but the increased cost may hamper its feasibility. In addition, there is a lack of immediate support for third-party tools for the latest iOS and Android releases.

3.	Extensibility	Native UI components with a cost factor and longer time to adjust to new versions. However, using Xamarin. Forms built-in layouts, controls, and pages allow designing applications from a single API that is highly extensible.
4.	Learning Success	Steep learning curve.
5.	Document and Support	Good documentation with limited community support.
6.	Development Effort	More tools and sufficient documentation makes it easier for developers. However, the learning curve is a significant problem area.
7.	Maintainability	Xamarin is a more mature framework than Flutter, and it receives continuous support from Microsoft, which makes long-term maintenance more manageable.

Table 3 – Xamarin

5.1.3 React Native

No.	Criteria	Evaluation
1.	License Cost	Cost Efficient.
2.	Long-Term Feasibility	React Native is among the top mobile application development framework, and the applications built using React Native are considered stable, reliable and ensure faster development of applications. Moreover, new codes to a running app and faster debugging make it worthy of long-term consideration.
3.	Extensibility	React Native applications are currently compatible with Android and iOS, with different expectations of building compatibility with other platforms like Windows, Linux, or macOS. In addition, react Native offers specific components like React Native Bit and Navigation that allow customization and layer-based extensibility.
4.	Learning Success	Wide range of tutorials and libraries to allow quick and easy development and increases comprehensibility. The learning curve is more straightforward for developers with React or JavaScript experience.
5.	Document and Support	Good documentation and availability of support from a growing community.
6.	Development Effort	Various companies use mature frameworks; therefore, several packages are readily available, and platform inherent features are simpler to implement.
7.	Maintainability	Continuous support by Facebook with libraries, tools, UI, and a robust community ensures maintainability. In contrast, React Native lacks platform-specific modules

		and smooth navigation compared to cross-platform development, leading to application issues.

Table 4 – React Native

5.1.4 Ionic

No.	Criteria	Evaluation
1.	License Cost	Significant costing exists with various plans and charges.
2.	Long-Term Feasibility	Ionic is not suitable for building complex applications and to include intricate functionalities in the applications. Furthermore, there is a likelihood of encountering challenges in terms of adding image processing features and animations. Thus it is ideal for simple applications and may not be feasible for large-scale organizational requirements.
3.	Extensibility	The provision of ready-made components, typography, and base themes are extensible to adapt to a different platform and well-known to provide efficient scalability. Moreover, due to the uses of Angular, various modules allow construction components for the applications to support modularity.
4.	Learning Success	Developers with the knowledge of Angular and JavaScript, CLI, HTML, and CSS will adapt faster with an easy learning curve.
5.	Document and Support	Concise documentation with an increasing developer community.
6.	Development Effort	The use of a capacitor saves a significant amount of developmental time.
7.	Maintainability	No additional testing devices are required, which makes it relatively easy to maintain. On the contrary, the absence of hot reloading features, performance issues with memory-intensive apps, and security issues make the overall maintainability challenging for specific functionalities.

Table 5 – Ionic

5.1.5 Cordova

No.	Criteria	Evaluation
1.	License Cost	The unique mobile platform-related developer program enrollment is free to use, with the cost accrued for deploying applications to these platforms.
2.	Long-Term Feasibility	Cordova provides a myriad of solutions for access device features through native plugins. In addition,

		there is support for multiple third-party plugins, and developers can run their plugins. With multiple functionalities and from the cost perspective, long-term feasibility can be considered. However, specific issues require to be evaluated before finalizing this framework for a long-term purpose.
3.	Extensibility	Extensibility is well-supported across platforms and does not require reimplementation with different platform languages or toolsets.
4.	Learning Success	There is a smaller learning curve with easy-to-learn functionalities and setup.
5.	Document and Support	There is a lack of appropriate documentation and community support. As a result, the phone Gap Build has been discontinued as of 2020. However, after Adobe System has taken over Cordova, community support is improving.
6.	Development Effort	Before the takeover by Adobe, Cordova lacked proper documentation, and it has caused a significant drop in its popularity as developers found it difficult to understand the framework without proper community support and documentation. With the emergence of Flutter, Cordova supporters are waning with a possibility to decrease further.
7.	Maintainability	The performance of Cordova applications is slower than native applications and tends to crash, and challenging to debug immediately can be tricky when it comes to maintainability.

Table 7 – Cordova

5.1.6 Kotlin

No.	Criteria	Evaluation
1.	License Cost	Very expensive to build applications using Kotlin. Additional charges are applicable when using Kotlin SDK with semi-native functionalities.
2.	Long-Term Feasibility	The possibility of interoperability with Java and code-sharing functionality makes it a unique and user-friendly application development language. Moreover, additional features and automation make it one of the most feasible development languages concerning long-term consideration.
3.	Extensibility	Creating a functional application with Kotlin that supports modularity and is easy to test is more effortless. Moreover, multiplatform programming is Kotlin's key area which makes it flexible and extensible.

4.	Learning Success	Easy to learn with a simple syntax similar to other programming languages such as Java or Swift.
5.	Document and Support	Documentation is concise, but community support is limited as it is a new alternative.
6.	Development Effort	As Kotlin is new, familiarity with technical stacks and components is essential. In addition, there is a need for utilizing several native components and additional development resources, which increases the cost of development in terms of time.
7.	Maintainability	Despite the simplicity of the syntax and easy learning curve, various features are not supported, such as hot reload. Additionally, support for libraries is constantly being developed and evolving. Therefore, maintainability is tricky in the current context but has scope for improvement as Google supports it constantly grows.

Table 8 – Kotlin

Staying ahead of the competition in the fast-paced technology industry requires a business to implement every tool at its disposal to provide the best user experience. Therefore, mobile applications are critical for a business to set its footprint in the market segment. Developing a robust mobile application is thus essential with the right tools, libraries, and associated software development kits, which can be achieved with the right choice of a mobile development platform.

Mobile application development frameworks are used for developing and deploying mobile applications, which are critical assets for the business. On the one hand, while selecting a framework, scalability, security, speed, usability, cross-platform support, offline mode operability are some of the most sought features before finalizing a framework for an organization. On the other hand, the criteria catalog highlights some key areas from a business and developers' perspective, including licensing cost, long-term feasibility, documentations, support, maintainability, and extensibility of the framework to incorporate new updates without affecting the performance of the applications. However, the primary question prevails on how to select despite the various criteria available. Firstly, it begins from the managerial decisions of the business and the targeted market that defines the choice for the development framework.

There are two primary types of mobile development frameworks, namely native mobile development framework and cross-platform development framework. The development of native applications is specific to a platform, whereas cross-platform applications can work on multiple platforms and operating systems.

Why opt for an application development framework and the factors to consider?

The development frameworks generally clarify the mobile software development lifecycle (MSDLC), including planning, analysis, designing, development, testing and security, deployment, and maintenance. These frameworks assist developers during different phases of the development lifecycle. However, it is essential to understand that different frameworks have different features that benefit the developmental process to speed up the development.

Finally, the factors to consider while selecting a development framework over another includes

- Platform, which is the most crucial aspect for the choice of a framework. Typically, the industry-oriented platform is Android OS, which is home to various applications worldwide that targets smartphones, wearables, tablets, and much more. In addition, the Apple OS platform is another popular platform for various Apple products.

- Speed is critical for applications to operate efficiently without lags to create a smooth user experience. For example, gaming applications require the maintenance of a certain speed to operate without compromising performance. Thus, the developers require a framework that can provide the codes and libraries to include additional functionalities vital for the application to maintain optimal performance.

- The cost of development determines which development framework to integrate. Therefore, the cost factors need to be assessed, alongside the expected financial gains from the application, before opting for a framework.

- In the digital landscape, the risk of data breaches remains a constant threat. Therefore, the nature of applications determines the framework. For example, the applications of high-end organizations in the e-commerce platform or the banking applications with payment gateways require a robust and secure framework that can ensure a secured environment in the long term.

- When the best framework debate occurs, the scope of a mobile application development framework delves into the efficiency factor. Therefore, it is of utmost importance for the frameworks to provide code patterns, plugins, and tools that can ensure high-quality output of the end product with faster development time and market-ready applications.

- Finally, the most crucial factor is the future enhancement of the applications with security patches, upgrades, updates, and bug fixing to maintain a high-quality user experience after deployment.

6. Conclusion

The best framework debate has waned over the years. With different perspectives from developers and business leaders, choosing the appropriate framework is no longer challenging. Based on the business requirements and the target market, the framework can be chosen. However, a critical consideration is the flexibility of the framework in the future. The trends in mobile development have witnessed a paradigm using codes, scripts, integration, development methodologies, and deployment.

Furthermore, each framework offers several advantages and disadvantages that determine the feasibility in the long run. Moreover, in the era of open-source technology, the availability of excellent developer community support remains an everyday necessity across frameworks. While sure developers are more confined to a particular programming language, the choices of the frameworks are determined by which the framework supports programming language. Finally, the code reusability and more accessible learning curve with readily available documentation determine the framework's popularity. Over the years, the frameworks with a lack of support documentation have been surpassed in terms of popularity as larger

companies are taking over the development frameworks, emphasizing the readability of documentation and usability. In a nutshell, there is no particular approach on the best framework. The business requirements coupled with community support, pricing structure, and security assurance with flexibility for enhancements will remain the fundamental requirements.

Overall, mobile development is passing through an exciting transformation from native to hybrid development and web applications. However, it remains to be seen how each framework adapts to the most sought-after technologies such as artificial intelligence and the Internet of Things (IoT) to provide smart services and better automation, and improved user experience. Perhaps, the debate over the best framework will be decided based on which of the frameworks integrate these advanced technologies in the future to provide a seamless experience to the end-users.

Copyright

Copyright © 2022 Mitrais

Disclaimer

Any and all information in this document has been compiled and provided for information purposes only. The information provided herein may include information compiled from a variety of third parties. Mitrais will not be liable for any loss, damage, cost or expense incurred in relation to or arising by reason of any person relying on the information in this document or any link to any website provided herein, whether or not caused by negligence on Mitrais' part. While Mitrais endeavours to provide the information up to date and correct, Mitrais make no representations or warranties of any kind, express or implied, about the accuracy, reliability, completeness or currency of the information or its usefulness in achieving any purpose. Readers of this document are responsible for assessing its relevance and verifying the accuracy of the content. In any event, the information provided herein should not be construed as providing advice whether legal or otherwise.