STM JOURNALS

Journal of
Open Source Developments

ISSN: 2395-6704
Volume 10, Issue 2, 2023
DOI (Journal): 10.37591/JoOSD

http://computers.stmjournals.com/index.php?journal=JoOSD&page=index

Review JoOSD

Accelerating Cross-platform Development with Flutter Framework

Arif Md. Sattar[1,*], Preetam Soni[2], Mritunjay Kr. Ranjan[3], Amar Kumar[4], Chandrahas Sahu[5], Shilpi Saxena[6], Prafulla Chaudhari[7]

Abstract

The need for businesses to spread the word about their wares to a larger audience has led to the rise in popularity of cross-platform mobile application development in recent years. Google created the open-source Flutter framework for creating mobile apps. It has rapidly replaced other options as the method of choice for developing high-quality, cross-platform software. Flutter's cross-platform compatibility across macOS, Linux, and Windows is only one of its many advantages. The framework can also be used to build web apps and desktop software in addition to mobile apps. The purpose of this study is to examine the pros and cons of utilizing Flutter to create cross-platform mobile applications. We will examine the pros and cons of utilizing Flutter to create cross-platform mobile, web, and desktop applications in contrast to other popular cross-platform development frameworks like Xamarin and React Native. We'll dive into Flutter's architecture, features, and distinctions from existing cross-platform development frameworks in the context of building applications for several platforms. Case studies of many Flutter-built apps will help us weigh the benefits and drawbacks of this framework for building apps that run on multiple platforms. There will be mobile, online, and desktop apps available, all built for many platforms. We'll be rating these applications based on a number of criteria, including how well they operate and how satisfying their users find the whole experience to be. The ultimate goal of this article is to provide readers with an in-depth understanding of the benefits and drawbacks associated with developing cross-platform applications using Flutter. The app's portability between platforms will be part of this comprehension. With this study, we aim to equip developers with the knowledge they need to make informed decisions about the development framework they will employ for future projects by comparing Flutter to other cross-platform development frameworks and investigating the capabilities it offers for a variety of application categories.

***Author for Correspondence**
Arif Md. Sattar
E-mail: amsattargaya@gmail.com

[1]Assistant Professor, Department of Computer Science & Information Technology, Anugrah Memorial Collège, Gaya, Bihar, India
[2,4,5]Student, Department of Computer science and application, Atal Bihari Vajpayee Vishwavidyalaya Bilaspur, Chhattisgarh, India
[3,6]Assistant Professor, School of Computer Sciences and Engineering, Sandip University Nashik, Maharashtra, India
[7]Assistant Professor, Department of Electronics and Telecommunication Engineering, Sandip Institute of Technology and Research Centre, India

Received Date: June 19, 2023
Accepted Date: June 30, 2023
Published Date: August 14, 2023

Citation: Arif Md. Sattar, Preetam Soni, Mritunjay Kumarr Ranjan, Amar Kumar, Chandrahas Sahu, Shilpi Saxena, Prafulla Chaudhari. Accelerating Cross-platform Development with Flutter Framework. Journal of Open Source Developments. 2023; 10(2): 1–11p.

Keywords: Flutter, Cross-Platform Development, Google, Windows, Xamarin, Mobile Systems.

INTRODUCTION

The need for companies and developers to provide mobile apps that are compatible with a wide variety of platforms has led to a rise in the popularity of cross-platform mobile app development. Flutter, a mobile development framework created by Google, is one of the cutting-edge innovations in this space since it enables the creation of natively built apps for mobile, web, and desktop platforms all from a single codebase. Due to its novel design and features, such as the hot reload option that enables fast iteration and rapid development and the widgets that give a rich and

versatile set of possibilities for designing bespoke user interfaces, Flutter has quickly acquired popularity [1]. Flutter's ability to compile to native code for iOS, Android, and other platforms has also made it a popular option among enterprises. This article will investigate the pros and cons of using Flutter for cross-platform development, including the influence it has on efficiency, the quality of the end-user experience, and the overall performance of the programme. We will also cover the limits and downsides of Flutter, as well as the best practises and techniques for creating cross-platform applications with it [2]. The overarching goal of this study is to shed light on the viability of Flutter as a framework for creating cross-platform mobile apps of the highest quality, and to provide direction to developers and enterprises interested in making use of this tool.

BENEFITS OF USING FLUTTER FOR CROSS-PLATFORM APP DEVELOPMENT

1. Flutter is a framework that may be used to create cross-platform applications with a number of benefits. Developers may make changes to the code and observe the effects instantly thanks to the hot reload capability. This function speeds up development and decreases wait times [3].
2. Flutter's ability to share a common codebase for Android and iOS app development is another plus. As a consequence, the development process is simplified and costs are saved by not having to maintain several codebases [4].
3. Flutter also gives programmers access to a wide variety of pre-made widgets and components for designing a unified UI across devices. This guarantees that the app provides the same functionality and visual experience across all devices. In addition, the app operates smoothly on both Android and iOS thanks to Flutter's native speed, which is achieved via the usage of a compiled code approach [5].
4. Finally, Flutter's extensive feature set makes it an excellent platform for developing aesthetically stunning and engaging applications, since it supports animation, graphics, and video [6].
5. Flutter is widely used because of its many advantages for creating cross-platform apps. It's a great framework for making high-quality cross-platform programs since it can boost productivity, cut down on mistakes, and provide native performance [7].

CHALLENGES OF CROSS-PLATFORM APP DEVELOPMENT WITH FLUTTER

There are a number of factors that developers must take into account while working on cross-platform programs [8]. As each operating system has its own set of features and needs, it may be difficult to ensure compatibility across all of them. This makes it harder to provide a consistent user experience across different platforms and might cause compatibility concerns. Another difficulty arises when trying to build a unified UI across many platforms. Developers need to make sure that the app's design is consistent with the app's overall design and branding while also adhering to the platform's design standards and guidelines [9].

When developing a cross-platform app, performance is also crucial. Users that want quick and responsive performance may be put off by cross-platform programs due to their reliance on external plugins and frameworks.

When creating a cross-platform programme, it's also important to take into account the platform's native capabilities. Developers need to make sure their app works with each platform's native features to provide their users the greatest possible experience.

Last but not least, it might be difficult and time consuming to test the software on several platforms. It might be difficult to detect and address problems when they only manifest on one platform.

If they want their cross-platform programme to perform well on all platforms and live up to users' expectations, developers need to give serious thought to these issues. Developers may build high-quality cross-platform applications that deliver a consistent user experience and benefit consumers on several platforms if they overcome these obstacles.

COMPARING FLUTTER TO OTHER CROSS-PLATFORM DEVELOPMENT FRAMEWORKS

Here, we'll compare Flutter, React Native, and Xamarin in terms of how quickly and efficiently they can create cross-platform apps—two of the most important criteria [10].

Ease of Development

With Flutter's quick reload functionality, developers can view their work immediately without having to restart the app to see the effects of their changes. Although React Native supports hot reload, it may not perform as quickly as Flutter. However, Xamarin might slow down the development process since changes need to be compiled after each one.

Code Sharing

Flutter's single codebase makes it possible to develop applications for Android, iOS, and the web while maximizing code reuse. Even though React Native allows code exchange, certain tweaks may be needed for each platform. Xamarin has a similar strategy of a common codebase, however it may call for additional platform-specific code.

Development Tools and Libraries

Faster app creation is possible with Flutter because to its extensive collection of pre-built UI widgets and frameworks. There is a rich ecosystem of third-party libraries available for use with React Native, developed and maintained by its active community. There is a vast selection of libraries available for Xamarin as well, albeit they may need extra bindings to work with platform-specific components.

Learning Curve

Dart, the language used by Flutter, is simple to pick up for programmers with experience in other Java-related languages. Since React Native is written in JavaScript, a language already well-known to online developers, it facilitates their move into mobile app development. For developers who aren't already fluent in C#, Xamarin may need some further study.

Platform-Specific

To facilitate the development of platform-specific UI components, Flutter provides a very flexible UI framework. Although more settings may be needed, React Native also supports platform-specific modifications. Xamarin also offers platform-specific UI components, albeit additional platform-specific code may be necessary.

Flutter's single codebase and quick reload capability may help speed up the development process, and both Flutter and React Native are well-known for their rapid pace and efficiency. Xamarin provides effective development in a similar vein, although it may slow down development in certain cases due to the need for platform-specific code and compilation stages. The needs of the project and the expertise of the development team will determine which framework will be the most effective in terms of development speed and efficiency as shown in Figure 1.

When compared to other cross-platform frameworks, Flutter stands out for using its own custom, high-performance rendering engine. Flutter uses the Skia 2D graphics library for efficient and quick rendering across several platforms, as opposed to depending on native rendering engines like React Native or Xamarin.

Flutter uses its own rendering engine, avoiding the performance hit that would be taken by using a platform's native engine. This means that Flutter apps run as smoothly as native ones do, with no hiccups in animation or functionality.

The robust hot reload functionality of the framework is in part due to the Flutter rendering engine, which allows for quick iteration and real-time debugging. When a developer makes a modification to

the code, they may see the results without having to restart the programme. This greatly improves productivity and speeds up the design and creation process.

The rendering engine of Flutter is driven by Skia, which provides a high-performance and consistent rendering solution, giving it a significant edge. Because of this, Flutter is a great option for developing aesthetically pleasing and responsive apps for a variety of devices, with the added benefits of superb performance, smooth animations, and a streamlined development process as shown in Figure 2.

Due to their relative youth, both the Flutter and React Native frameworks have massive community support and dominate their respective communities to almost the same extent. According to Google Trends, 81% of developers are considering learning Flutter, whereas 61% like React Native [11].

Development Speed and Efficiency in Flutter

Flutter is a widely-used cross-platform mobile app development framework, and it is often praised for the ways in which it streamlines the app creation process. Here we'll explore how Flutter's rapid and efficient development may be achieved in more detail [12].

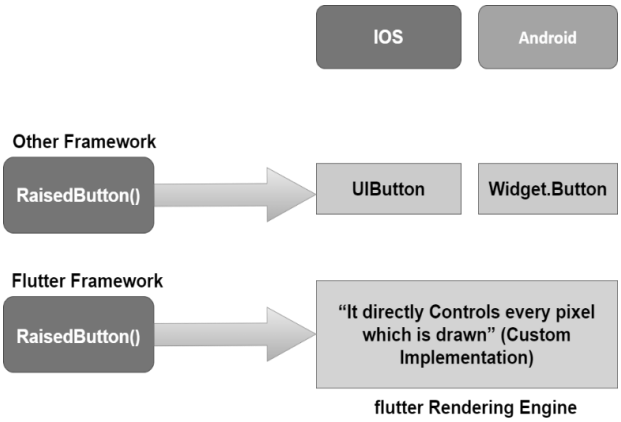

Figure 1. Flutter frame work Comparison.

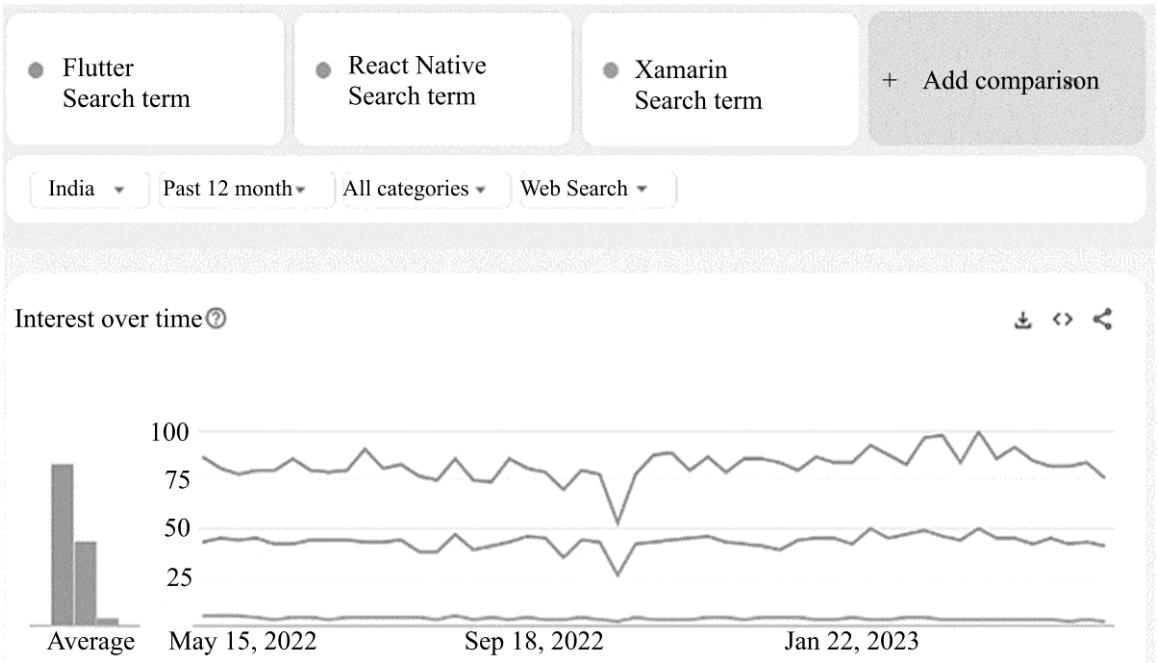

Figure 2. Graphical Form of Flutter frame work Comparison.
Image Source: trends.google.com

1. ***Hot Reload:*** Hot reload is a feature of Flutter that enables developers to observe the effects of their changes to the app's UI and functionality without having to restart the app. Accelerating development via fast iterations and quick modifications made possible by a real-time feedback loop.
2. ***Single Codebase:*** With Flutter, developers no longer have to generate and manage several codebases for Android and iOS app development. By reducing wasted time and lowering the likelihood of mistakes, this method of code sharing speeds up the development process.
3. ***Rich Widget-Based UI Framework:*** Widgets are the building blocks of Flutter's UI framework. Widgets are UI components that can be easily modified and reused. Flutter's rich collection of ready-made widgets simplifies the creation of even the most sophisticated user interface designs.
4. ***Productive Development Environment:*** The official integrated development environment (IDE) for Flutter is Visual Studio Code (VSCode), which includes several time-saving tools like as code autocompletion, debugging, and built-in emulators. This more simplified setting for development helps boost efficiency and output.
5. ***Testing and Debugging:*** In order to help developers find and fix errors as early as possible in the development process, Flutter includes tools for writing and running tests for their applications. Furthermore, Flutter's powerful debugging features allow for speedy problem resolution, since the UI can be inspected and modified in real time.
6. ***Active Community and Ecosystem:*** There is a large and productive group of programmers working on Flutter, therefore there is a plethora of information available in the form of tutorials, guides, and third-party libraries. Because of its size, this ecosystem can provide ready-made solutions and cut down on the requirement for bespoke development, which speeds up the development process.
7. ***Continuous Integration and Deployment (CI/CD) Support:*** Developers can automate the construction, testing, and deployment of their apps using Flutter's built-in support for continuous integration and continuous delivery. This contributes to a streamlined development process, which in turn speeds up the rollout of improved software versions.

All of Flutter's features—including its quick reload functionality, unified codebase, widget-based UI framework, productive development environment, testing/debugging tools, active community, and continuous integration/continuous deployment (CI/CD) support—contribute to its rapid development speed and efficiency. It's worth noting, however, that the complexity of the software, the experience level of the development team, and other variables may affect the real pace and efficiency of development.

Performance and Rendering Engine in Flutter

Flutter's rendering engine is vital in providing fast and fluid user experiences, a crucial part of mobile app development. Here, we'll examine Flutter's and its rendering engine's speed and efficiency in further detail [10].

1. ***Skia Rendering Engine:*** Flutter's rendering engine is the powerful and efficient 2D graphics engine Skia from the Skia graphics package, which has hardware-accelerated rendering. Skia allows Flutter to render UI components, animations, and images quickly and efficiently, leading to smooth and fluid UIs.
2. ***Dart Language and Just-In-Time (JIT) Compilation:*** Dart, the language used in the creation of Flutter applications, has capabilities such as AOT and JIT compilation. In particular, just-in-time (JIT) compilation facilitates speedy development cycles by facilitating hot-reloading and rapid recompilation during development, hence shortening development time.
3. ***Widgets and Reactive Framework:*** The reactive programming methodology and widget-based UI framework used by Flutter make it easy to quickly draw and change UI elements. The "Everything is a Widget" philosophy behind Flutter's design allows for fine-grained control over the user interface and quick widget rebuilding, which in turn reduces the burden of superfluous UI changes and boosts speed.

4. ***Hardware-Accelerated Animation:*** Flutter provides a wide variety of APIs for accelerating animations in hardware, making use of the device's graphics processing unit (GPU). The resulting animations are fluid and aesthetically pleasing, while having a negligible effect on the app's overall speed.
5. ***Platform-Specific Performance Optimization:*** To provide the best possible performance on each platform, Flutter permits platform-specific performance optimizations such as the use of platform-specific plugins and access to native APIs. As a consequence, developers are able to fine-tune performance for diverse target devices and their unique setups.
6. ***Performance Profiling and Optimization Tools:*** The Flutter Dev Tools suite is only one of several performance measurement and optimization tools provided by Flutter that help developers locate and eliminate app slowdowns. These resources help make Flutter applications run as smoothly and efficiently as possible.
7. ***Continuous Performance Improvements:*** The developers behind Flutter are always hard at work, improving the framework with new releases and tweaks to improve its already impressive speed. Flutter applications can rest certain that they are built on a rock-solid foundation and will provide users with a high-performance experience thanks to our dedication to constant performance advancements.

In conclusion, Flutter's strong performance capabilities come from its Skia rendering engine, Dart language and JIT compilation, widget-based UI framework, hardware-accelerated animation, platform-specific performance optimization, performance profiling and optimization tools, and continuous performance improvements. It is worth noting, however, that other variables, including as the complexity of the app, the device hardware, and the optimization strategies chosen by the development team, may also affect the real speed of a Flutter app.

Platform Support and Compatibility in Flutter

Cross-platform development frameworks, such as Flutter, are essential because they provide the platform support and compatibility necessary for developers to construct programs that can operate natively on many platforms. In this article, we'll explore how Flutter's platform compatibility and support contribute to the framework's flexibility as a cross-platform tool [13, 14].

1. ***Comprehensive Platform Support:*** With Flutter, you can build cross-platform apps that run on iOS, Android, the web, and desktop operating systems including Windows, macOS, and Linux. Flutter is a flexible framework for cross-platform app development because to its adaptability, which enables developers to create applications that operate natively on a broad variety of devices and platforms.
2. ***Platform-Specific Widgets and Material Design:*** With Flutter, developers have access to a wide variety of platform-specific widgets, allowing them to design user interface elements that are visually consistent with the native style of the supported platform. In addition, Flutter is compatible with Google's design framework, Material Design, which guarantees a uniform and attractive user interface across devices.
3. ***Platform-Specific APIs and Plugins:*** With Flutter, programmers can access platform-specific APIs and plugins, unlocking the full potential of the underlying platform. Users are given an experience as close to native as possible because to the app's ability to seamlessly integrate with the platform's camera, location, and sensors.
4. ***Hot Reload and Fast Iteration:*** By allowing for instantaneous feedback on code changes, Flutter facilitates rapid iteration and development cycles. In addition to speeding up development, this also guarantees that the app's design and behavior will be identical across all supported devices.
5. ***Compatibility with Existing Code and Libraries:*** To reuse their codebase and incorporate with popular libraries from the Flutter ecosystem, developers can rest certain that Flutter is compatible with their current code and resources. This makes Flutter developer-friendly for cross-platform app development by encouraging efficient code exchange and reducing development effort.
6. ***Device Emulators and Simulators:*** Developers can test their applications without real devices thanks to Flutter's built-in device emulators and simulators for iOS, Android, and the web. As a result, it's easier to test how well an app will run on several platforms before releasing it.

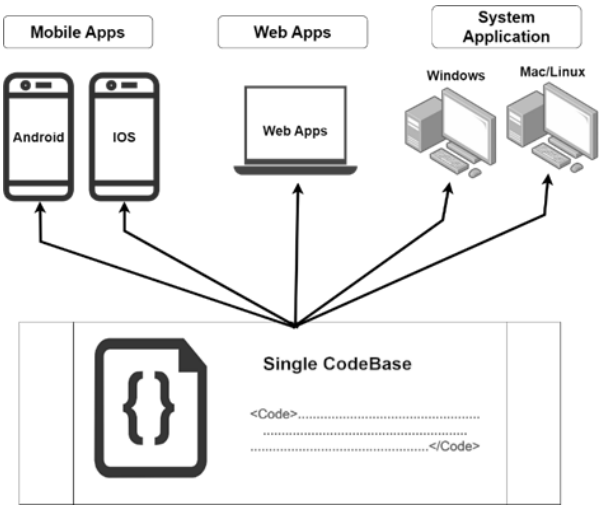

Figure 3. Cross Platform Model.

Flutter has a large developer community, and Google, the framework's inventor, offers regular updates and support to keep it current and compatible with new versions of supported platforms. This guarantees that Flutter applications will continue to be supported and updated as platform needs change as shown in Figure 3.

In conclusion, Flutter is a highly compatible and versatile framework for cross-platform app development due to its wide range of supported platforms, compatibility with existing code and libraries, platform-specific widgets and APIs, hot reload feature, device emulators and simulators, and continuous updates and support. This allows programmers to build programs that are native to various platforms, each of which can then provide its users with the same excellent experience.

LIMITATIONS AND CONSTRAINTS OF FLUTTER FOR CROSS-PLATFORM APP DEVELOPMENT

It is essential to point out that while Flutter does have some restrictions, it also has a number of key advantages, such as the ability to reuse code, rapid development cycles, and capabilities for hot reloading [15]. The choice to utilise Flutter for the creation of cross-platform apps need to be arrived at after an in-depth analysis of the requirements of the project, thought-through examination of potential trade-offs, as well as the knowledge and preferences of the development team.

1. **Limited Native Access:** Flutter provides a wide variety of platform-specific widgets and plugins, although there may be limits to the app's interaction with native platform capabilities if some native functionality or third-party libraries are not made available.
2. **Learning Curve:** Those who are used to working with other programming languages may find it difficult to learn Dart, the programming language used by Flutter, and it may take some time to become proficient in the syntax and concepts unique to Flutter and Dart.
3. **App Size:** Because the Flutter engine and framework are embedded in the app, Flutter applications could be bulkier than their native counterparts. This may increase the amount of space used on the device and slow down the process of downloading and installing apps, especially in places with limited internet access.
4. **Performance Limitations:** However, complicated and resource-intensive applications may have performance limits, especially on older or lower-end devices, despite Flutter's reputation for high-speed UI rendering capabilities. Apps that use a lot of graphics, animations, or calculations may need optimizations and speed adjustments.
5. **Limited Ecosystem:** While Flutter's ecosystem of plugins and modules is expanding, it still lacks the community support and ecosystem maturity of older frameworks like React Native. When compared to more established frameworks, this might mean less resources, documentation, and community support are accessible.

6. **Platform-Specific Design Constraints:** The UI components may not always be in perfect harmony with the native design patterns and rules of each platform, despite the fact that Flutter provides platform-specific widgets and adheres to Material Design. This may reduce the app's natural appearance and feel, which might negatively affect its reception.
7. **Continuous Updates and Changes:** Being a young framework, Flutter goes through numerous upgrades and changes, some of which may cause incompatibilities or need apps to be updated. This may increase the time and effort needed to keep the app updated to the most recent version of Flutter, which might be problematic for maintenance and continuing support.

While there are many benefits to using Flutter for building cross-platform apps, there are also some drawbacks that developers should be aware of. These include: a steeper learning curve, larger apps, lower performance, a less-mature ecosystem, platform-specific design constraints, and frequent updates and revisions. When deciding on Flutter as a cross-platform development framework and planning app development initiatives, it is necessary to give careful thought to these constraints [16].

BEST PRACTICES FOR CROSS-PLATFORM APP DEVELOPMENT WITH FLUTTER

There are a number of recommended practises that we should adhere to in order to make sure that the process of designing cross-platform apps using Flutter goes as smoothly and quickly as possible. The following are some best practises that are advised for developing cross-platform applications with Flutter [17]:

1. **Adhere to Flutter's UI and UX Guidelines:** Flutter offers user interface widgets that adhere to the Material Design rules for Android and the Cupertino Design guidelines for iOS. This ensures that users will have a consistent and native-like experience regardless of the platform they are using.
2. **Optimize App Performance:** Improve the efficiency of your app by optimizing the UI rendering, reducing the number of needless widget rebuilds, and optimizing the retrieval and processing of data. Utilize the performance profiling tools that Flutter provides in order to identify and eliminate performance bottlenecks.
3. **Utilize Platform-Specific Widgets and Plugins**: Utilize the widgets and plugins that are platform-specific to Flutter in order to customize the appearance and feel of the application for each platform, so assuring a native-like experience on both Android and iOS.
4. **Plan for Different Screen Sizes and Orientations**: For a seamless user experience across a variety of devices and orientations, the app should be designed to automatically adjust to varied screen sizes and positions. Examine the app on a variety of devices to check for compatibility issues.
5. **Handle Platform-Specific Edge Cases:** Be careful to take into account platform-specific edge situations in order to guarantee that your application will work correctly on each platform. For example, you should handle platform-specific permissions and alerts and integrate with platform-specific services.
6. **Use Flutter's Testing and Debugging Tools**: Make use of the testing and debugging tools provided by Flutter, such as widget testing, integration testing, and the Developer Tools, in order to test and debug the app in great detail to ensure that it is accurate and reliable.
7. **Keep Dependencies Updated:** Updates to Flutter and its accompanying plugins and libraries should be installed on a regular basis in order to maintain the app's stability and security and to take advantage of new features and bug fixes.
8. **Follow Code Organization and Best Practices:** Maintainable and scalable code may be achieved by adhering to Flutter's suggested best practises and organizational patterns for code. These patterns include state management patterns, the separation of UI functionality from business logic, and adherence to the DRY principle.
9. **Test on Real Devices:** Because emulators and simulators do not always adequately represent real-world performance, it is important to test the application on actual Android and iOS devices in order to get an accurate assessment of how it behaves on a variety of hardware combinations and operating system versions.

10. **Follow Material Design and Cupertino Design Guidelines:** For an experience that is consistent and feels natural across Android and iOS platforms, adhering to the rules established by Material Design and Cupertino Design in terms of UI/UX design, typography, color palettes, and iconography is essential.

In conclusion, sticking to best practises for cross-platform app development with Flutter guarantees the construction of an app of the highest possible quality, which is also highly performant and user-friendly, and which runs faultlessly on the Android platform as well as the iOS platform. You will be able to provide a top-notch cross-platform mobile app using Flutter if you adhere to the rules provided by Flutter, optimize performance, make use of platform-specific widgets and plugins, and undertake extensive testing on the app.

CASE STUDY

Reflectly created a cross-platform app with a consistent and high-quality user experience on iOS and Android using Flutter. The software earned great reviews, a large user base, and excellent app store ratings. Reflectly's experience with Flutter showed its potential for developing feature-rich, visually attractive, and performant cross-platform apps. Flutter helped Reflectly create a strong and engaging personal diary app with native-like speed, cross-platform development, and a stunning user experience [18]. It showcases Flutter's ability to develop high-quality mobile apps quickly.

1. **Objective:** A patient will be able to plan appointments with their healthcare providers on mobile, web, and desktop platforms using an appointment app that is being developed with the goal of creating a user experience that is both simple and straightforward.
2. **Background:** There has been an upsurge in the demand for digital appointment booking systems as a direct result of the COVID-19 epidemic. Appointment applications may offer a secure and simple means for people to communicate with their healthcare professionals, especially with the growth of telemedicine in recent years.
3. **Development process:** The development process for the Flutter appointment app involved the following steps:
    - ***i. Planning:*** The group came up with a strategy for the app, which included its features, functioning, and the components of its design. In addition to this, they investigated a variety of different appointment applications to determine industry standards and user preferences.
    - ***ii. Design:*** The user interface of the app was built by the team utilizing the widgets and design components that are customizable in Flutter. The team's primary objective was on developing a streamlined and user-friendly interface that would be maintained across all platforms.
    - ***iii. Development:*** Throughout the whole of the app's development, both Flutter and the programming language known as Dart were used. The team worked hard to integrate the app with a number of application programming interfaces (APIs), such as a scheduling API and a payment gateway API, in order to ease the process of booking appointments and processing payments via the app.
    - ***iv. Testing:*** In order to ensure that the application had a quality that was up to par, it was subjected to rigorous testing, which consisted of both human and automated testing techniques.
    - ***v. Deployment:*** Because the application was released on a number of different platforms, such as mobile app stores, web browsers, and desktop operating systems, users are now able to access it using whatever device they choose.
    - ***vi. Outcome:*** The Flutter appointment app has gotten great comments from customers, who praised its streamlined and easy-to-understand user experience as well as its capability to plan appointments from any device. The development team is constantly making enhancements to the app based on the comments and suggestions provided by users, as well as introducing new features and capabilities.
    - ***vii. Conclusion:*** The creation of the Flutter appointment app highlights the advantages of adopting Flutter for the development of cross-platform apps. The team was able to produce

a high-quality app that is uniform across all platforms and offers a user experience that is both smooth and straightforward with the assistance of Flutter's fully adjustable widgets and design features, as well as Flutter's quick development process. Cross-platform appointment applications may be a helpful and secure method for consumers to communicate with their healthcare providers in this age of increasing interest in telemedicine and digital health as shown in Table 1.

Table 1. Comparison of different cross platform application

Tech	Platform	App Speed	Development Cost	Development Time
Flutter	Code Once Deploy multiple	Avg	Low	Less
React Native	Code Once Deploy multiple	Avg	Low	Less
Native (Kotlin, Swift, C++ etc)	Develop platform specific app	Fast	High	More

CONCLUSION

Flutter is a sophisticated cross-platform mobile app development framework that has garnered a large amount of popularity recently. It offers advantages such as speedy development, efficient UI rendering, and a single codebase that can be used for both Android and iOS. On the other hand, it does have certain restrictions and difficulties, such as platform-specific limits and the problem of optimising performance.

FUTURE DIRECTIONS

Several directions may be taken to make Flutter better in the future. This includes bolstering platform-specific support, honing performance and the rendering engine, growing the ecosystem of plugins and libraries, upgrading tools and DevOps support, fostering more acceptance and community support, and maintaining tight interaction with Google's tools. It is anticipated that these enhancements will make Flutter an even more flexible and powerful platform for building cross-platform apps. Cross-platform mobile app quality, user friendliness, and performance may all benefit from developers keeping up with the newest Flutter improvements and taking use of the framework's benefits. Version 3.7 of Flutter, released in the start of 2023, offers a plethora of new features and improvements. New rendering engine built from the ground up for iOS, better support for Material 3 and iOS-style widgets, updated internationalization support, and better background processing are just some of the major changes.

Updates to the developer tools have also made it easier and more intuitive to create high-quality cross-platform applications using Flutter. These updates and enhancements have helped solidify Flutter's position as a leading framework for creating apps that run across many platforms.

REFERENCES

1. V. Yakovyna and B. Uhrynovskyi, "Aging of Native and Flutter Applications in Android OS in Various Usage Scenarios," 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT), LVIV, Ukraine, 2021, pp. 313–316, doi: 10.1109/ CSIT52700.2021.9648777.
2. N. Garg, Palak and N. Goel, "Real-time deployment and test set analysis of automatic colonoscopy polyp identification architecture," 2023 10th International Conference on Signal Processing and Integrated Networks (SPIN), Noida, India, 2023, pp. 346-350, doi: 10.1109/SPIN57001.2023.10116179.
3. M. A. Faiz, D. S. Kusumo and M. J. Alibasa, "Flutter Framework Code Portability Measurement on Multiplatform Applications with ISO 9126," 2022 1st International Conference on Software Engineering and Information Technology (ICoSEIT), Bandung, Indonesia, 2022, pp. 36–40, doi: 10.1109/ICoSEIT55604.2022.10030045.
4. S. Durai, C. Shyamalakumari and T. Sujithra, "Cloud Computing based Multipurpose E-Service Application using Flutter," 2022 6th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2022, pp. 1122–1126, doi: 10.1109/ICCMC53470. 2022.9753968.

5. S. Dovgyi, O. Trofymchuk, O. Lebid, I. Kaliukh, V. Berchun and Y. Berchun, "Aeroelastic Flutter Oscilations of Distributed Systems," 2022 IEEE 3rd KhPI Week on Advanced Technology (KhPIWeek), Kharkiv, Ukraine, 2022, pp. 1–5, doi: 10.1109/KhPIWeek57572.2022.9916499.
6. M. -D. Pop and A. -R. Stoia, "Improving the Tourists Experiences: Application of Firebase and Flutter Technologies in Mobile Applications Development Process," 2021 International Conference Engineering Technologies and Computer Science (EnT), Moscow, Russian Federation, 2021, pp. 146–151, doi: 10.1109/EnT52731.2021.00033.
7. P. Bhangale, B. Bhatt, M. Nandu and P. Chavda, "MeetUp: An Appointment Booking System using Flutter and Django Framework : MeetUp: Meetings Made Easy," 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2021, pp. 1425–1431, doi: 10.1109/ICOSEC51865.2021.9591786.
8. G. Luongo et al., "Non-Invasive Characterization of Atrial Flutter Mechanisms Using Recurrence Quantification Analysis on the ECG: A Computational Study," in IEEE Transactions on Biomedical Engineering, vol. 68, no. 3, pp. 914–925, March 2021, doi: 10.1109/TBME.2020.2990655.
9. 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.
10. K. Kishore, S. Khare, V. Uniyal and S. Verma, "Performance and stability Comparison of React and Flutter: Cross-platform Application Development," 2022 International Conference on Cyber Resilience (ICCR), Dubai, United Arab Emirates, 2022, pp. 1–4, doi: 10.1109/ICCR56254. 2022.9996039.
11. M. J. Aziz, S. Hussain, M. Z. Bashir and S. Bilal, "Flutter analysis of ARW-2 wing using doublet lattice," 2018 15th International Bhurban Conference on Applied Sciences and Technology (IBCAST), Islamabad, Pakistan, 2018, pp. 579–584, doi: 10.1109/IBCAST.2018.8312283.
12. W. C. Wilson, J. P. Moore and K. R. Brinker, "The use of a Reflectometer as a Monostatic Radar for Measuring Aircraft Structural Flutter," 2019 IEEE National Aerospace and Electronics Conference (NAECON), Dayton, OH, USA, 2019, pp. 702-706, doi: 10.1109/NAECON46414. 2019.9058150.
13. M. H. Kamarul Azman, O. Meste, D. G. Latcu and K. Kadir, "Non-Invasive Localization of Atrial Flutter Circuit Using Recurrence Quantification Analysis and Machine Learning," 2019 Computing in Cardiology (CinC), Singapore, 2019, pp. Page 1-Page 4, doi: 10.22489/CinC.2019.222.
14. M. U. Gul, K. Kadir and M. H. Kamarul Azman, "Data Augmentation for Discrimination of Atrial Flutter Mechanism Using 12-Lead Surface Electrocardiogram," 2021 Computing in Cardiology (CinC), Brno, Czech Republic, 2021, pp. 1–4, doi: 10.23919/CinC53138.2021.9662957.
15. S. M. Hari Krishna et al., "Trip Planner and Recommender using Flutter and Tensor Flow," 2022 IEEE 7th International conference for Convergence in Technology (I2CT), Mumbai, India, 2022, pp. 1–7, doi: 10.1109/I2CT54291.2022.9824468.
16. S. Sharma, S. Khare, V. Unival and S. Verma, "Hybrid Development in Flutter and its Widgits," 2022 International Conference on Cyber Resilience (ICCR), Dubai, United Arab Emirates, 2022, pp. 1–4, doi: 10.1109/ICCR56254.2022.9995973.
17. T. Fatkhulin, R. Alshawi, A. Kulikova, A. Mokin and A. Timofeyeva, "Analysis of Software Tools Allowing the Development of Cross-Platform Applications for Mobile Devices," 2023 Systems of Signals Generating and Processing in the Field of on Board Communications, Moscow, Russian Federation, 2023, pp. 1–5, doi: 10.1109/IEEECONF56737.2023.10092148.
18. S. Gowri, C. Kanmani Pappa, T. Tamilvizhi, L. Nelson and R. Surendran, "Intelligent Analysis on Frameworks for Mobile App Development," 2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2023, pp. 1506–1512, doi: 10.1109/ICSSIT55814.2023.10060902.

