CASE STUDY
ON
FLUTTER

By Joyson Fernandis
Indira college Of Commerce And Science
TyB. Sc -Computer Science
Email: joysonfdis@gmail.com

Content

Abstract

Cross-platform mobile application development is the pressing priority in today's world and generation. Developers are enforced to either construct the same application numerous times for various OS (operating systems) or accept a low-quality similar solution that trades native speed and accuracy for portability. Flutter is an open-source SDK for developing high-performance and more reliable mobile applications for operating systems like iOS and Android. Significant features of the Flutter are Just-in-time compilation which executes the computer code that encompasses compiling during program execution at run time rather than preceding execution. More frequently, this comprises of bytecode translation lesser known as source code to machine code, which is unswervingly executed. AOT compilation (Ahead-of-time compilation) compiles a high-level programming language such as C or C++, or an intermediary representation such as Java bytecode or NET Framework Common Intermediate Language (CIL) code, into native system-dependent machine code so that the subsequent binary file can execute natively. Flutter has a feature called as hot reload which helps you easily experiment, build UIs, add features, and fix bugs. Hot reload works by inserting updated source code files into the running Dart Virtual Machine (VM). After the VM updates classes with the new versions of fields and functions, the Flutter framework automatically reconstructs the widget tree, permitting you to rapidly view the special effects of your changes. Flutter targets the top mobile operating systems like Android and iOS, it gives you a solution for GPU rendering and UI, powered by native ARM code.

KEYWORDS: Cross-Platform Mobile application development, IDE, Android development, iOS development, Flutter, Dart. I. I

1. INTRODUCTION

1.1 Flutter

Flutter was unveiled at the 2015 Dart Developer Summit under the name Sky. Eric Seidel

React serves as inspiration for the latest framework used to create Flutter widgets. The main concept is that widgets are used to construct user interfaces. Widgets specify how their current configuration and status should appear in their display. A widget rebuilds its description as its state changes, and the framework compares it to the old description to find the minimum modifications required in the render tree underneath to move from one state to the next.

In general, developing a mobile application is a complex and challenging task. There are many frameworks available to develop a mobile application. Android provides a native framework based on Java language and iOS provides a native framework based on Objective-C / Shift language. However, to develop an application supporting both the OSs, we need to code in two different languages using two different frameworks. To help overcome this complexity, there exists mobile frameworks supporting both OS. These frameworks range from simple HTML based hybrid mobile application framework (which uses HTML for User Interface and JavaScript for application logic) to complex language specific framework (which do the heavy lifting of converting code to native code). Irrespective of their simplicity or complexity, these frameworks always have many disadvantages, one of the main drawback being their slow performance. In this scenario, Flutter – a simple and high performance framework based on Dart language, provides high performance by rendering the UI directly in the operating system's canvas rather than through native framework. Flutter also offers many ready to use widgets (UI) to create a modern application. These widgets are optimized for mobile environment and designing the application using widgets is as simple as designing HTML. To be specific, Flutter application is itself a widget. Flutter widgets also supports animations and gestures. The application logic is based on reactive programming. Widget may optionally have a state. By changing the state of the widget, Flutter will automatically (reactive programming) compare the widget's state (old and new) and render the widget with only the necessary changes instead of re-rendering the whole widget.

1.2 About Flutter

Flutter is a cross-platform framework that targets developing high-performance mobile applications. Flutter was publicly released in 2015 by Google. Besides running on Android and iOS flutter applications also run on Fuschia. Flutter is chosen as Google's application-level framework for its next-generation operating system. Flutter is exceptional because it is dependent on the device's OEM widgets rather than consuming web views. Flutter uses a high-performance rendering engine to render each view component using its own. This provides a chance to build applications that are as high-performance as native applications can be. In view of architecture, the engine's C or C++ code involves compilation with Android's NDK and LLVM for iOS respectively, and during the compilation process, the Dart code is compiled into native code. Hot reload feature in Flutter is called as Stateful hot reload and it is a major factor for boosting the development cycle. Flutter supports it during development. Stateful hot reload is implemented by sending the updated source code into the running Dart Virtual Machine (Dart VM) without changing the inner structure of the application, therefore the transitions and actions of the application will be well-preserved after hot reloading.

Flutter is a cross-platform framework that targets developing high-performance mobile applications. Flutter was publicly released in 2016 by Google. Besides running on Android and iOS flutter applications also run on Fuschia. Flutter is chosen as Google's application-level framework for its next-generation operating system. Flutter is exceptional because it is dependent on the device's OEM widgets rather than consuming web views. Flutter uses a high-performance rendering engine to render each view component using its own. This provides a chance to build applications that are as high-performance as native applications can be. In view of architecture, the engine's C or C++ code involves compilation with Android's NDK and LLVM for iOS respectively, and during the compilation process, the Dart code is compiled into native code. Hot reload feature in Flutter is called as Stateful hot reload and it is a major factor for boosting the development cycle. Flutter supports it during development. Stateful hot reload is implemented by sending the updated source code into the running Dart Virtual Machine (Dart VM) without changing the inner structure of the application, therefore the transitions and actions of the application will be well-preserved after hot reloading.

2. LITERATURE SURVEY

2.1. Literature Review

As we know that there are lots of mobile applications which are used nowadays. To develop these applications the developers, work their best to provide a good experience but they also face a lot of difficulties. One of the major problems faced by the developers is to select the OS which is either Android and iOS. For instance, a developer wants to develop an application then the choice that is to be made is that if the application is to be developed for both the OS or for only one. Majorly, the application is developed for both the OS.

Now to develop the application for different OS, the code should also be written in a different language. The code for Android is written in java and for iOS, the code is written in Swift language. This is a bit difficult for the developer to learn 2 different languages and use them to the full extent to develop the same application but for a different OS. It is very time consuming, as code is to be written in java and swift, this development of application in the different platform is known as cross-platform development.

To overcome the cross-development problem, a software that is developed by google known as flutter is used. Now what flutter does is that it allows the developers to use a single codebase and develop the application in both Android and iOS platforms, without making any changed to the code. This is done by deploying the flutter code written in a dart language. Dart is the language that is used to develop the application through flutter.

When a code is written in a dart language then the code is deployed in the respective OS IDE (Integrated Development Environment), which is a platform that supports the code of a language. For instance, the IDE of Android is Android Studio and iOS is XCode. As soon as the code is deployed, the IDE automatically makes the judgment according to the OS for any necessary changes and allows the application to be developed successfully.

The experiment is conducted, that a basic program of hello world is created. This is a basic application in which when the user opens the application, the text “Hello World”, will be displayed on the screen. Now the program is written in different languages like Java for Android, Swift for iOS, and in dart for flutter application.

The code written in different languages is compared with each other. By doing the analysis, it is observed that the code written in Android is very complex, different files need to be used for a different purpose. For the user interface, the layout file is used and for text, a string file is used and another main file is also used to run the application. In swift language, all code can be written in one file but the code is too complex to understand and learn. But in a flutter, there is no need to use a separate file for a separate purpose, all the text, display widgets used can be coded in one file, which is the main file. The code is so simple to understand, easy to learn, moreover, this code can be used for both the platform and it takes less time to work in dart as compare to the other two languages, which means that code in the flutter combines in about 2–3 seconds.

The results from the experiment conclude that flutter development, the application developed through flutter can be used to solve the cross-development problem which is being faced by the developers.

3. PROBLEM STATEMENT

- **Performance Optimization**: Ensure efficient memory management, minimize app size, and optimize rendering for smooth performance on various devices and screen sizes.

- **Data Management**: Implement robust data management techniques, including local 9data storage (using SQLite or shared preferences) and integration with remote APIs (RESTful services or GraphQL).

- **User Authentication and Authorization**: Incorporate secure user authentication methods such as OAuth, JWT, or Firebase Auth, along with role-based access control (RBAC) for authorization.

- **Offline Support**: Provide offline capabilities by implementing data caching and synchronization mechanisms, allowing users to access certain features and content even without an active internet connection.

- **Internationalization and Localization**: Support multiple languages and regions by implementing internationalization (i18n) and localization (l10n) features to make the app accessible to a global audience.

- **Accessibility**: Ensure that the app is accessible to users with disabilities by following accessibility best practices, including support for screen readers, high contrast modes, and customizable text sizes.

- **Push Notifications**: Enable push notifications to engage users and keep them informed about important updates, leveraging platforms such as Firebase Cloud Messaging (FCM) or OneSignal.

- **Security**: Implement security measures to protect user data and prevent common vulnerabilities such as SQL injection, cross-site scripting (XSS), and insecure data transmission.

- **Testing and Quality Assurance**: Develop comprehensive test suites, including unit tests, integration tests, and UI tests, using tools like Flutter's built-in testing framework, Dart's test library, and external testing frameworks like Flutter Driver or Appium.

- **Continuous Integration and Deployment (CI/CD)**: Set up automated build, testing, and deployment pipelines using CI/CD tools like Jenkins, Travis CI, or GitHub Actions to streamline the development process and ensure consistent quality across releases.

- **Feedback and Analytics**: Integrate feedback mechanisms and analytics tools (such as Firebase Analytics or Google Analytics) to gather user insights, track app performance, and make data-driven decisions for future iterations.

- **Scalability and Extensibility**: Design the app architecture with scalability and extensibility in mind, allowing for easy integration of new features, modules, and third-party libraries as the app evolves over time.

- **Documentation and Support**: Provide comprehensive documentation, including code comments, API references, and user guides, to facilitate future maintenance and support activities for developers and end-users alike.

4. OBJECTIVE AND SCOPE

4.1 Objective

 Mastering Flutter Framework: Aim to become proficient in utilizing Flutter's framework, including its widgets, layout system, and state management techniques, to efficiently develop cross-platform mobile applications.

 Creating Engaging User Experiences: Focus on designing and implementing captivating user interfaces and experiences using Flutter's rich set of customizable widgets and animations, ensuring that apps are visually appealing and intuitive to use.

 Optimizing Performance: Strive to optimize app performance by implementing best practices in code structure, resource management, and rendering techniques, ensuring smooth animations and fast response times across various devices and platforms.

 Embracing Continuous Learning: Commit to staying updated with the latest Flutter advancements, tools, and best practices through ongoing learning, attending conferences, participating in community events, and engaging with fellow developers.

 Collaborating Effectively: Work collaboratively with multidisciplinary teams, including designers, developers, and stakeholders, to translate project requirements into successful Flutter applications, fostering effective communication, teamwork, and synergy.

4.2 Scope

Despite the several Flutter advantages and disadvantages, the UI framework promises a bright future. It is bound to grow and evolve. Here are some areas that promise Flutter scope in future in the coming years. These are:

- **Expanding Platform Support:** Flutter has already extended its Despite the several Flutter advantages and disadvantages, the UI support to other platforms beyond the mobile domain, such as web and desktop. In the future, we can further expand the language, enabling developers to create applications for a range of devices.
- **Improved Performance:** As Flutter evolves with time, we can anticipate better performance optimizations. This will make it capable of enhancing its capabilities to handle resource-intensive applications and graphics.
- **Wider Adoption:** The different advantages of using Flutter will help it grow its user base and community support. This will lead to more developers and organizations adopting it for their projects.
- **Enhanced Native Integration:** Flutter will likely continue improving its native integration. This will make it easier for developers to access and use features of Flutter that are platform-specific and its APIs.
- **App Store Acceptance:** Over time, as Flutter becomes more established, the concern regarding approval from the app store will most likely diminish. It will provide Flutter apps with more familiarity and reduce potential hurdles in the approval process.

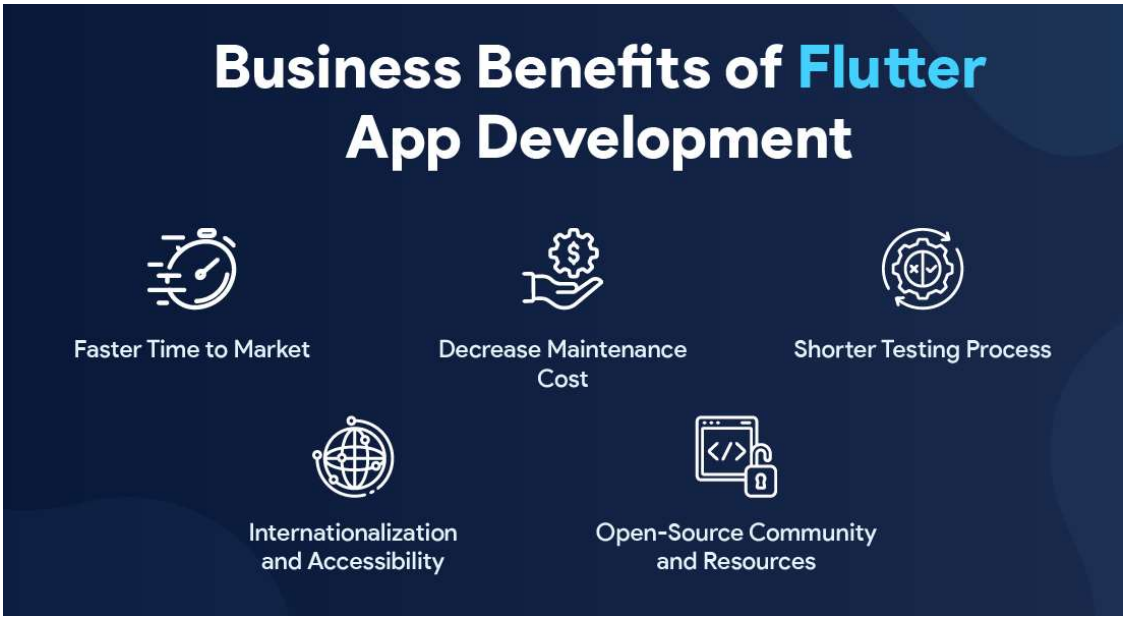
- **Stability and Long-term Support:** The active support from the community and Google will contribute to its long-term stability and support. It will also alleviate the concerns related to the framework's future.

- **Education Training:** Educational institutions have started to include Flutter in their online curriculum. It will produce a new generation of developers that are proficient in Flutter, and it will most likely enable its growth.

- **Enterprise Adoption:** There are several advantages of adopting Flutter, as discussed earlier, such as faster development, lower cost, and consistent user experience, etc. This makes this language an attractive choice for enterprises. As Flutter becomes more mainstream, it will garner larger traction in the corporate world.

4.3 Benefits of Flutter

Flutter has gained a stellar reputation in the app development community due to its numerous benefits and advantages. Here are some of the top reasons why developers and businesses alike are embracing Flutter for their mobile app development needs:



4.3.1 Benefits of Flutter

Cross-Platform Development

One of the most significant advantages of Flutter is its ability to enable cross-platform app development. With Flutter, developers can write code once and deploy it seamlessly across multiple platforms, including iOS, Android, web, and desktop. This cross-platform capability not only saves time and resources but also ensures a consistent user experience across different devices and operating systems.

Hot Reload for Faster Development

Flutter's hot reload feature is a game-changer in terms of accelerating the development process and enhancing productivity. With hot reloading, developers can instantly see the changes they make to their code reflected in the running application without the need for a full restart or recompilation.

Customizable and Flexible UI

Flutter's approach to user interface (UI) design offers developers a high degree of customization and flexibility. Unlike traditional mobile app development frameworks that rely heavily on platform-specific UI components, Flutter provides a comprehensive set of customizable widgets that can be tailored to match the desired look and feel of the application.

Comprehensive Widget Library

Flutter comes equipped with a vast and comprehensive widget library that offers a wide range of pre-built UI components and building blocks. This extensive collection of widgets covers various UI elements, such as buttons, text fields, sliders, menus, and navigation components, among others.

Robust Performance

When it comes to mobile app development, performance is a critical factor that can make or break the user experience. Flutter excels in this regard, delivering exceptional performance and ensuring a smooth, responsive, and efficient application across various platforms and devices.

Growing Community and Support

Flutter's popularity has been steadily rising, thanks to its powerful features and the backing of tech giant Google. As a result, a vibrant and thriving community of developers has emerged, actively contributing to the framework's growth and development. This active community offers a wealth of resources, including documentation, tutorials, forums, and open-source libraries and packages. Developers can leverage these resources to learn, share knowledge, and find solutions to common challenges, accelerating their development process and enhancing their skills.

5. ADVANTAGES AND DISADVANTAGES

5.1 Advantages

Flutter comes with beautiful and customizable widgets for high performance and outstanding mobile application. It fulfills all the custom needs and requirements. Besides these, Flutter offers many more advantages as mentioned below:

- Dart has a large repository of software packages which lets you to extend the capabilities of your application.

- Developers need to write just a single code base for both applications (both Android and iOS platforms).

- Flutter may to be extended to other platform as well in the future. Flutter needs lesser testing. Because of its single code base, it is sufficient if we write automated tests once for both the platforms.

- Flutter's simplicity makes it a good candidate for fast development. Its customization capability and extendibility makes it even more powerful.

- With Flutter, developers has full control over the widgets and its layout. Flutter offers great developer tools, with amazing hot reload.

- **Fast Development**: Flutter's hot reload feature allows developers to see changes in the code almost instantly reflected on the app, speeding up the development and debugging process significantly.

- **Beautiful UIs**: Flutter offers a rich set of customizable widgets and a flexible design system that allows developers to create visually appealing and highly interactive user interfaces that look and feel native on both iOS and Android.

- **Open Source and Backed by Google**: Flutter is an open-source framework backed by Google, which means it benefits from regular updates, contributions from the community, and ongoing support and development efforts from Google engineers.

5.2 Disadvantages

Despite its many advantages, flutter has the following drawbacks in it:

- Since it is coded in Dart language, a developer needs to learn new language (though it is easy to learn).
- Modern framework tries to separate logic and UI as much as possible but, in Flutter, user interface and logic is intermixed.
- We can overcome this using smart coding and using high level module to separate user interface and logic.
- Flutter is yet another framework to create mobile application.
- Developers are having a hard time in choosing the right development tools in hugely populated segment.
- **Large App Size**: Flutter apps tend to have larger file sizes compared to native apps because they include the Flutter engine and framework. This can be a concern, especially for apps targeting regions with slower internet connections or devices with limited storage.
- **Community Support**: While Flutter's community is growing rapidly, it may still not be as large or mature as other frameworks like React Native or native platforms. This could mean fewer resources, tutorials, or community support for developers encountering challenges.

6. APPLICATION

Flutter is a versatile framework that can be applied to various types of mobile applications across different industries. Some common applications of Flutter include:

1. **E-commerce Apps**: Flutter is well-suited for building feature-rich e-commerce applications with attractive user interfaces, smooth animations, and seamless payment integrations. Examples include shopping apps, delivery services, and online marketplaces.

2. **Social Networking Apps**: Flutter enables the development of social networking applications with interactive user interfaces, real-time messaging features, and social media integrations. Examples include chat apps, social media platforms, and community forums.

3. **Travel and Tourism Apps**: Flutter can be used to create travel and tourism applications with immersive experiences, location-based services, and booking functionalities for flights, hotels, and activities. Examples include travel guides, booking platforms, and navigation apps.

4. **Health and Fitness Apps**: Flutter is suitable for developing health and fitness applications with personalized workout plans, activity tracking features, and integration with wearable devices. Examples include fitness trackers, diet planners, and meditation apps.

5. **Financial Apps**: Flutter can be utilized to build financial applications with secure transactions, budget management tools, and real-time market data. Examples include banking apps, investment platforms, and cryptocurrency wallets.

6. **Education and E-learning Apps**: Flutter enables the creation of interactive educational applications with multimedia content, quizzes, and progress tracking features. Examples include e-learning platforms, language learning apps, and educational games.

7. **Media and Entertainment Apps**: Flutter can be used to develop media and entertainment applications with streaming capabilities, multimedia content, and personalized recommendations. Examples include music streaming apps, video-on-demand platforms, and gaming apps.

8. **Productivity and Utility Apps**: Flutter is suitable for building productivity and utility applications with task management features, note-taking capabilities, and productivity tools. Examples include to-do list apps, calendar apps, and document scanners.

9. **Event Management Apps**: Flutter enables the development of event management applications with event listings, ticket booking functionalities, and event reminders. Examples include event planning apps, conference apps, and ticketing platforms.

10. **IoT Apps**: Flutter can be applied to develop Internet of Things (IoT) applications with remote control capabilities, sensor monitoring features, and IoT device integrations. Examples include home automation apps, smart home controllers, and IoT dashboards.

7.CONCLUSION

Points to the success of a mobile-driven reward system are helping the retailers and small to medium shop owners to attract new customers, retain existing ones, and motivate increased purchase among current consumers. Offering loyalty programs attract the customers to invest in your brand and also leaves its imprints on their mind and brings your brand in the limelight.

REFERENCE

[1] React Native vs Flutter, Cross-Platform Mobile Application Framework, Thesis March 2018- Wenhau Wu.

[2] A clean approach to Flutter Development through the Flutter Clean architecture package, IEEE 2019, Shady Boukhary, Eduardo Colemenares.

[3] Exploring end user's perception of Flutter mobile apps, Malmo University Nov 2019- Dahl, Ola.

[4] Flutter for Cross-Platform App and SDK Development, Metropolia University Thesis May 2019- Lucas Dagne.

[5] Cross-Platform Framework comparison- Flutter vs React Native.

[6] Flutter Native Performance and Expressive UX/UI, paper 2019- Tran Thanh