A Seminar Report on

**“Cross-Platform Development With Flutter”**

submitted by

**Master Saquib Akhtar Aneesur Rahman**

(Roll No: 59, DIV: B)

*submitted in partial fulfilment of the requirement for the award of the degree of*

**Bachelor**

**in**

**COMPUTER ENGINEERING**

For the Academic Year 2020-21

*Under the guidance of*

**Prof. Choudhary Atul Shridharrao**

DEPARTMENT OF COMPUTER ENGINEERING

MET’s Institute of Engineering, Bhujbal Knowledge City,

Adgaon, Nashik-422003

**SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE**

**MET’s Institute of Engineering, Bhujbal Knowledge City,**

**CERTIFICATE**

*This is to certify that*

**Master Saquib Akhtar Aneesur Rahman (Roll no. 59)**

*Has completed the necessary Seminar work and prepared the report on*

“Cross-Platform Development With Flutter”

*In a satisfactory manner as a fulfilment of the requirement of the award of degree of Bachelor of Computer Engineering in Academic year*

*2020-2021*

**Seminar Guide H.O.D.**

Prof. Choudhary Atul Shridharrao Dr. M.U. Kharat

**Acknowledgement**

I, **Saquib Akhtar Aneesur Rahman** in all my capacity do extend my gratitude to all the people involved for all the development in the course of **“Seminar and Technical communication”.**

I hereby with a sense of thankfulness, thank our guide **Prof. Choudhary Atul Shridharrao**, of Computer department for all the guidance and all the opportunities he provided us in his capacities. I express my thanks for all the suggestions corrections and various new experiences that our guide has provided us with he has also held us in developing as a powerful professional in future by contributing his own efforts in ours.

I also thank your mentor P**rof. Manoj Suri** and our HOD **Dr. M. U. Kharat**, computer department for the cooperation and guidance for the fluent completion of the course. And lastly thanking all the people involved from our computer department for various help they have been providing for the successful completion of the course.

By

Master Saquib Akhtar Aneesur Rahman

**Contents**

1. Introduction 5

1.1 What is Flutter? . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

1.2 Features of Flutter. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

1.3 History of Flutter. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

1. Objectives 7
2. General Concept Flutter 8

3.1 Technology in which flutter is build. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

3.2 How the Flutter Works. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

3.3 Supported operating systems to build a Flutter app. . . . . . . . . . . . . . . . . . . . 8

3.4 What language is Flutter written in? . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

3.5 Why did Flutter choose to use Dart . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

1. Architecture of Flutter 15

4.1 Architecture for Different OS. . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . 10

4.2 Architecture for Web. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

1. Advantages and Disadvantages of Flutter 13

11.1 Advantages. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .13

11.2 Disadvantages. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .16

1. Conclusion 17
2. Future Scope 18
3. Bibliography 19

**1.** **Introduction to Flutter**

**1.1 What is Flutter?**

**Flutter** is an open-source UI software development kit created by Google. It is used to develop cross platform applications for Android, iOS, Linux, Mac, Windows, Google Fuchsia, and the web from a single codebase.

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 Features of Flutter**

Flutter framework offers the following features to developers:

* Modern and reactive framework.
* Uses Dart programming language and it is very easy to learn.
* Fast development.
* Beautiful and fluid user interfaces.
* Huge widget catalogue.
* Runs same UI for multiple platforms.
* High performance application.

**1.3 History of Flutter**

* In 2015 Google unveiled Flutter, a new SDK based on the Dart language, as the next platform for Android development, and in 2017 an alpha version of it (0.0.6) was released to the public for the first time.
* At I/O 2017 Google showed off using Flutter and its multi-platform capabilities, and continued promoting it at I/O 2018. Since then, Google has been investing in Flutter and recommending it as the way everyone should be developing mobile apps.
* In December 2018 Flutter 1.0 was released and made available so that developers could begin using the SDK to make app creation easier.
* At Google I/O 2019, Flutter support for desktop and web platforms was publicly announced. Tools for developing Flutter apps for Windows, macOS, Linux, and the web were released.
* By the end of 2020, Flutter support for desktop and web platforms became stable.

**2.** **Objectives**

* To make students familiar to Cross-Platform Development with Flutter.
  + Teach about the architecture and uses of framework
  + Advantages and disadvantages of framework
* To do a detailed research of the proposed activities in the scope of the seminar and technical communication course.

**3.** **General Concepts of Flutter**

**3.1 Technology in which flutter is build**

Flutter is built with C, C++, Dart, and Skia (a 2D rendering engine). For a more detailed description of the layered architecture of Flutter, read the [Architecture](https://flutter.dev/docs/resources/architectural-overview) of Flutter later in this document.

**3.2 How Flutter Works**

### **3.2.1 How does Flutter run code on Android?**

The engine’s C and C++ code are compiled with Android’s NDK. The Dart code (both the SDK’s and yours) are ahead-of-time (AOT) compiled into native, ARM, and x86 libraries. Those libraries are included in a “runner” Android project, and the whole thing is built into an .apk. When launched, the app loads the Flutter library. Any rendering, input, or event handling, and so on, is delegated to the compiled Flutter and app code. This is similar to the way many game engines work.

During debug mode, Flutter uses a virtual machine (VM) to run its code in order to enable stateful hot reload, a feature that lets you make changes to your running code without recompilation. You’ll see a “debug” banner in the top right-hand corner of your app when running in this mode, to remind you that performance is not characteristic of the finished release app.

### **3.2.2 How does Flutter run code on iOS?**

The engine’s C and C++ code are compiled with LLVM. The Dart code (both the SDK’s and yours) are ahead-of-time (AOT) compiled into a native, ARM library. That library is included in a “runner” iOS project, and the whole thing is built into an .ipa. When launched, the app loads the Flutter library. Any rendering, input or event handling, and so on, are delegated to the compiled Flutter and app code. This is similar to the way many game engines work.

During debug mode, Flutter uses a virtual machine (VM) to run its code in order to enable stateful hot reload, a feature that lets you make changes to your running code without recompilation. You’ll see a “debug” banner in the top right-hand corner of your app when running in this mode, to remind you that performance is not characteristic of the finished release app.

### **3.3 Supported operating systems to build a Flutter app**

Flutter supports development using Linux, macOS, ChromeOS, and Windows.

### **3.4 What language is Flutter written in?**

[Dart](https://dart.dev/), a fast-growing modern language optimized for client apps. The underlying graphics framework and the Dart virtual machine are implemented in C/C++.

### **3.5 Why did Flutter choose to use Dart?**

There are many reasons behind the decision made by Google to choose Dart as language for the Flutter framework. Here’s a summary of what brought them to make this choice.

3.5.1 OOP style

The vast majority of developers have object-oriented programming skills and thus Dart would be easy to learn as it adopts most of the common OOP patterns. The developer doesn’t have to deal with a completely new way of coding; he can reuse what he already knows and integrate it with the specific details of Dart.

3.5.2 Performances

In order to guarantee high performances and avoid frame dropping during the execution of the app, there’s the need of a high performance and predictable language. Dart can guarantee to be very efficient and it provides a powerful memory allocator that handles small, short-lived allocations. This is perfect for Flutter’s functional-style flow.

3.5.3 Productivity

Flutter allows developers to write Android, iOS, web and desktop apps with a single codebase keeping the same performances, aspect and feeling in each platform. A highly productive language like Dart accelerates the coding process and makes the framework more attractive.

3.5.4 Both are Developed by Google

Both Flutter and Dart are developed by Google which can freely decide what to do with them listening to the community as well. If Dart was developed by another company, Google probably wouldn’t have the same freedom of choice in implementing new features and and the language couldn’t evolve at the desired pace.

**4.** **Architecture of Flutter**

Flutter is a cross-platform UI toolkit that is designed to allow code reuse across operating systems such as iOS and Android, while also allowing applications to interface directly with underlying platform services. The goal is to enable developers to deliver high-performance apps that feel natural on different platforms, embracing differences where they exist while sharing as much code as possible.

During development, Flutter apps run in a VM that offers stateful hot reload of changes without needing a full recompile. For release, Flutter apps are compiled directly to machine code, whether Intel x64 or ARM instructions, or to JavaScript if targeting the web. The framework is open source, with a permissive BSD license, and has a thriving ecosystem of third-party packages that supplement the core library functionality.

This overview is divided into a number of sections:

1. The **layer model**: The pieces from which Flutter is constructed.
2. **Support for the web**: Concluding remarks about the characteristics of Flutter in a browser environment.

4.1 Architecture for different OS

Architectural
diagramFlutter is designed as an extensible, layered system. It exists as a series of independent libraries that each depend on the underlying layer. No layer has privileged access to the layer below, and every part of the framework level is designed to be optional and replaceable.

To the underlying operating system, Flutter applications are packaged in the same way as any other native application. A platform-specific embedder provides an entrypoint; coordinates with the underlying operating system for access to services like rendering surfaces, accessibility, and input; and manages the message event loop. The embedder is written in a language that is appropriate for the platform: currently Java and C++ for Android, Objective-C/Objective-C++ for iOS and macOS, and C++ for Windows and Linux. Using the embedder, Flutter code can be integrated into an existing application as a module, or the code may be the entire content of the application. Flutter includes a number of embedders for common target platforms, but [other embedders also exist](https://hover.build/blog/one-year-in/).

At the core of Flutter is the **Flutter engine**, which is mostly written in C++ and supports the primitives necessary to support all Flutter applications. The engine is responsible for rasterizing composited scenes whenever a new frame needs to be painted. It provides the low-level implementation of Flutter’s core API, including graphics (through [Skia](https://skia.org/)), text layout, file and network I/O, accessibility support, plugin architecture, and a Dart runtime and compile toolchain.

The engine is exposed to the Flutter framework through [dart:ui](https://github.com/flutter/engine/tree/master/lib/ui), which wraps the underlying C++ code in Dart classes. This library exposes the lowest-level primitives, such as classes for driving input, graphics, and text rendering subsystems.

4.2 Architecture for Web

While the general architectural concepts apply to all platforms that Flutter supports, there are some unique characteristics of Flutter’s web support that are worthy of comment.

Dart has been compiling to JavaScript for as long as the language has existed, with a toolchain optimized for both development and production purposes. Many important apps compile from Dart to JavaScript and run in production today, including the [advertiser tooling for Google Ads](https://ads.google.com/home/). Because the Flutter framework is written in Dart, compiling it to JavaScript was relatively straightforward.

Flutter web
architectureHowever, the Flutter engine, written in C++, is designed to interface with the underlying operating system rather than a web browser. A different approach is therefore required. On the web, Flutter provides a reimplementation of the engine on top of standard browser APIs. We currently have two options for rendering Flutter content on the web: HTML and WebGL. In HTML mode, Flutter uses HTML, CSS, Canvas, and SVG. To render to WebGL, Flutter uses a version of Skia compiled to WebAssembly called [CanvasKit](https://skia.org/user/modules/canvaskit). While HTML mode offers the best code size characteristics, CanvasKit provides the fastest path to the browser’s graphics stack, and offers somewhat higher graphical fidelity with the native mobile targets.

The web version of the architectural layer diagram is as follows:

Perhaps the most notable difference compared to other platforms on which Flutter runs is that there is no need for Flutter to provide a Dart runtime. Instead, the Flutter framework (along with any code you write) is compiled to JavaScript. It’s also worthy to note that Dart has very few language semantic differences across all its modes (JIT versus AOT, native versus web compilation), and most developers will never write a line of code that runs into such a difference.

During development time, Flutter web uses [dartdevc](https://dart.dev/tools/dartdevc), a compiler that supports incremental compilation and therefore allows hot restart (although not currently hot reload) for apps. Conversely, when you are ready to create a production app for the web, [dart2js](https://dart.dev/tools/dart2js), Dart’s highly-optimized production JavaScript compiler is used, packaging the Flutter core and framework along with your application into a minified source file that can be deployed to any web server. Code can be offered in a single file or split into multiple files through [deferred imports](https://dart.dev/guides/language/language-tour#lazily-loading-a-library).

**5.** **Advantages and Disadvantages of Flutter**

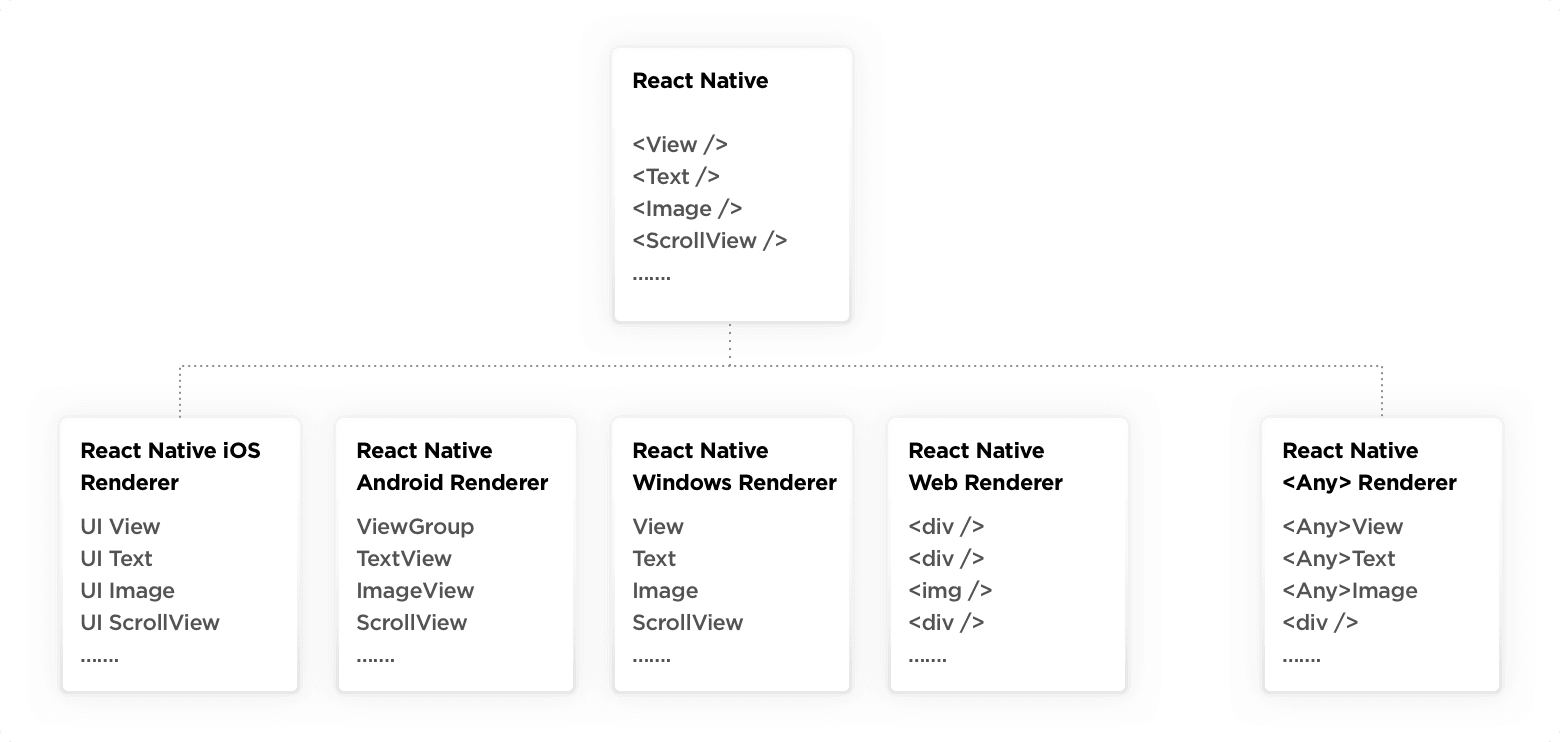
5.1 Advantages

we might assume that every cross-platform framework can perform some things and offers some features essential for the job, and we have a point. But! Flutter is a bit different from its competitors in some key aspects. Let’s take a closer look at what makes Flutter app development better than other frameworks.

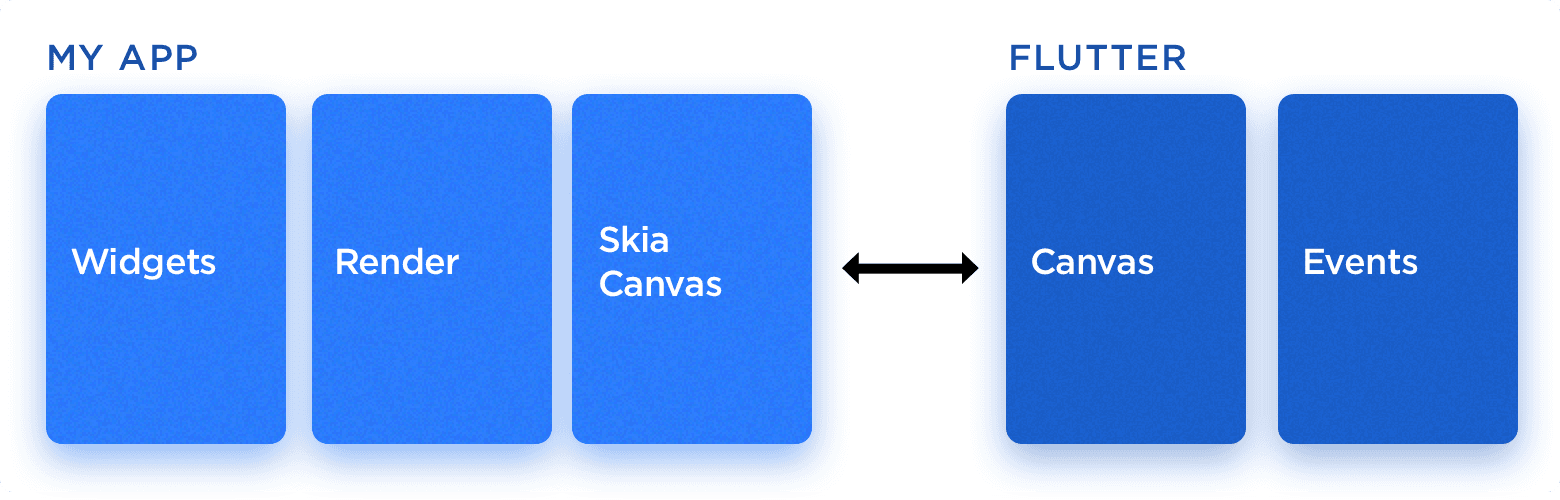
### **5.1.1 Same UI and Business Logic in All Platforms**

Experts, know that basically any cross-platform framework provides a way to share codebase between the target platforms. But there are no such application frameworks that allow sharing both the UI code the UI itself besides Flutter.

To illustrate, here’s an example of how UI rendering looks like in most cross-platform frameworks:



This kind of a rendering process makes building an app that looks native on every platform simple. But the devil’s in the details. Relying on platform-specific components for rendering provokes a need for a property mapping layer for the platform widget and a framework widget data synchronization. That’s what requires mapping every animation into a platform-specific widget call. So much more complicated than it needs to be, right?

In contrast, Flutter doesn’t need any platform-specific UI components to render its UI. The only thing Flutter needs to show the application UI is a canvas to draw onto. And here’s how it looks like: 

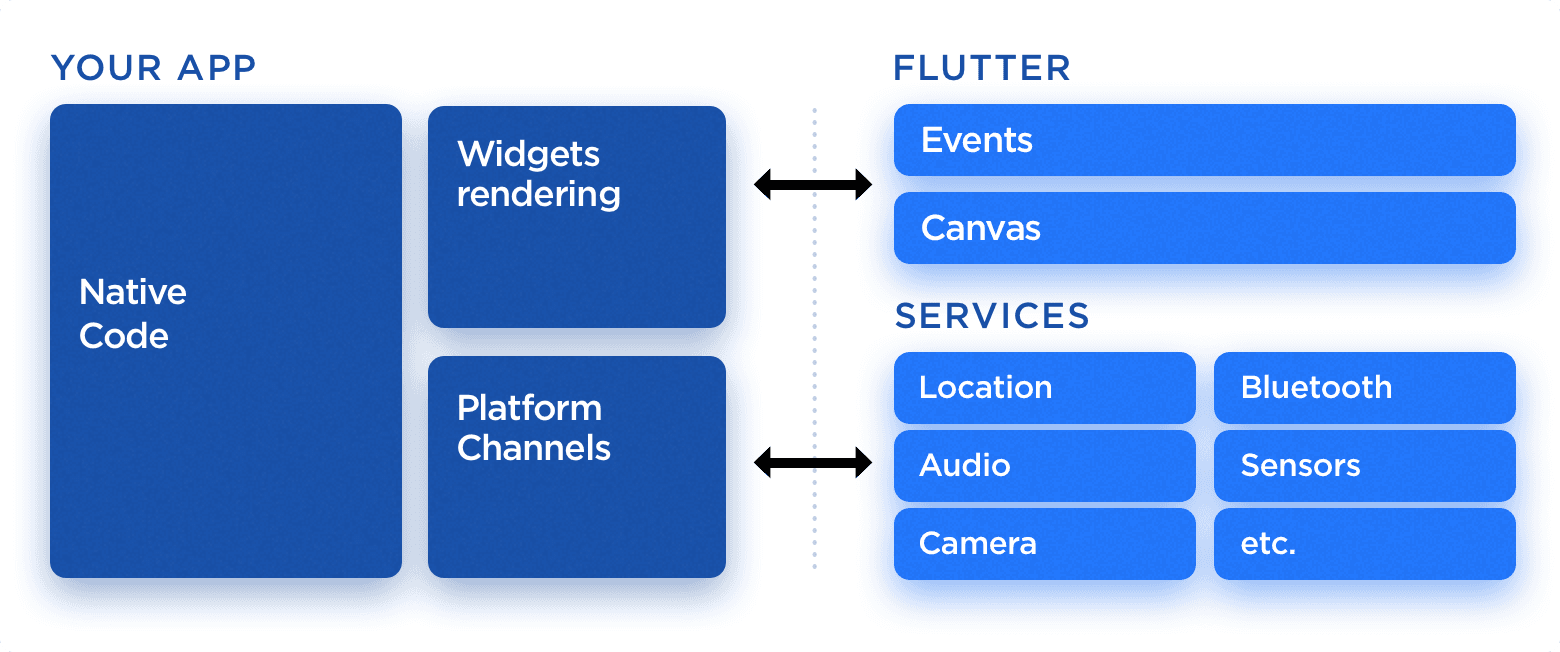
The Flutter’s way of rendering makes the framework really stand out from the crowd eliminating any worries about the UI consistency on different platforms.

In brief, sharing the UI and business logic, which is possible with Flutter, saves time, effort and the health of the developer while not affecting the performance of the end product.

### **5.1.2 Reduced Code Development Time**

Flutter’s “hot reload” feature, allows seeing the applied changes almost instantly, without even losing the current application state. And this is exactly what makes Flutter app development several times faster due to the increased development speed.

Besides, the Flutter team has put lots of effort into providing a wide variety of ready-to-use widgets. Most of them are incredibly customizable, saving your time like no other framework before. In addition to numerous core layout widgets, Flutter provides a large set of Material and Cupertino widgets that perfectly mimic the behaviour of each design language. Here’s how they work:



Altogether, this skips several heavy time-consuming steps in app development when using Flutter, which makes the entire process faster, simpler and less worrisome.

### **5.1.3 Increased Time-to-Market Speed**

Flutter development framework functions quicker than its alternatives. In most cases, you can expect a Flutter app to require at least two times fewer man-hours compared to the same app developed separately for Android and iOS. The main reason is very simple: you just don’t have to write any platform-specific code to achieve the desired visuals in your application. Any 2D-based UI can be implemented in Flutter without interacting with a native application counterpart.

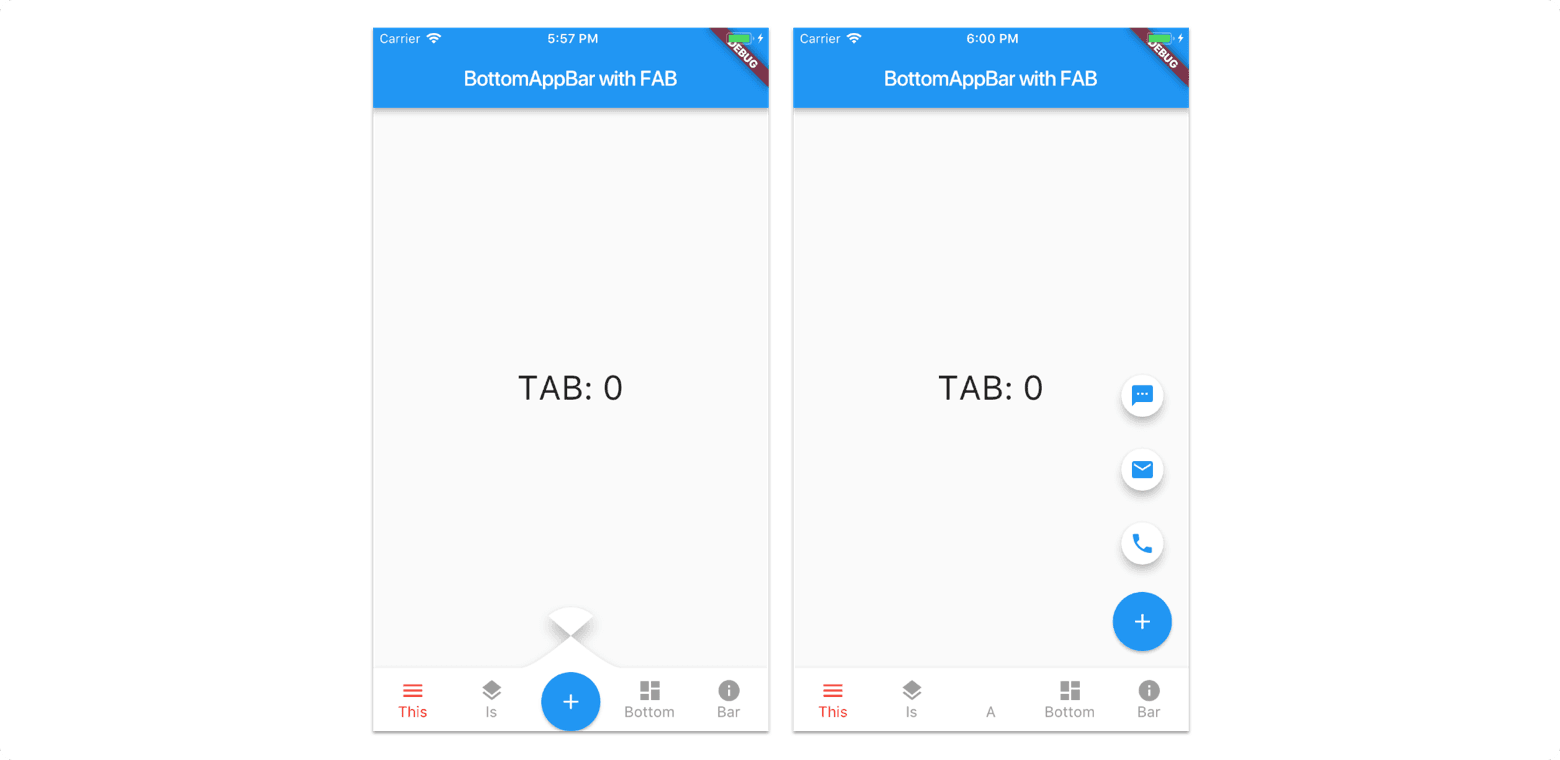
### **5.1.4 Similar to Native App Performance**

Application performance is crucial for good UX. While it’s hard to tell the exact figures, it’s safe to say that Flutter application performance in most cases will be indistinguishable from the native app and even better in complex UI animation scenarios.

Flutter doesn’t rely on any intermediate code representations or interpretation. Flutter application is built directly into the machine code, which eliminates any performance bugs of the interpretation process.

### **5.1.5 Custom, Animated UI of Any Complexity Available**

One of the biggest advantages of Flutter is the ability to customize anything you see on the screen, regardless of how complex it may be. While it’s usually possible to do a very custom UI on the native platforms as well, the amount of effort required differs by the order magnitude. Here’s an example of such simple yet custom UI:



### **5.1.6 Own Rendering Engine**

Flutter allows to do so much stuff with apps that aren’t available on other platforms. Obviously, it requires the framework to be pretty powerful. In fact, most of the points presented above wouldn’t be possible without a high-performance cross-platform rendering engine.

Flutter uses [Skia](https://skia.org/) for rendering itself onto a platform-provided canvas. Because of the engine, UI built in Flutter can be launched on virtually any platform. Putting it differently, so we no longer have to adjust UI to transfer it to a platform, which [simplifies the development process hugely](https://relevant.software/blog/7-steps-for-effective-software-product-development/https:/relevant.software/blog/7-steps-for-effective-software-product-development/).

### **5.1.7** Future Ready with Fuchsia Support

There are speculations that Google is working on a new operating system called Fuchsia which ‘could’ eventually replace android. Flutter is Fuchsia ready and you will be able to release your app of Fuchsia that day it is launched.

5.2 Disadvantages

we might assume that every cross-platform framework can perform some things and offers some features essential for the job, and we have a point. But! Flutter is a bit different from its competitors in some key aspects. Let’s take a closer look at what makes Flutter app development better than other frameworks.

5.2.1 Young Framework

Alpha released in 2017, Flutter is still an young framework It hasn’t been around for long, which is why it’s still not entirely stable. A number of more or less problematic issues remain, along with a lack of more advanced features that leverage the capabilities of operating systems.

5.2.2 Dart is Young

Dart is also pretty young. When comparing it to Swift and Kotlin, it’s basically like taking a step back - it has either fewer features or the existing ones are not exactly well-refined.

5.2.3 Generates Heavy Apps

Flutter apps are quite large and “heavy” to start with. They occupy a lot of space and take longer to download or update.

**6.** **Conclusion**

### With Flutter, the possibilities are practically endless, so even super extensive apps can be created with ease. If you develop mobile apps and have yet to give Flutter a try, I highly recommend you do.

### There are some things that pose a drastic value for a business. These are the stability of a platform, its performance, a wide talent pool – it’s relatively easy to [develop an app in Flutter](https://relevant.software/blog/how-to-hire-flutter-developers/) – and the guarantee of successful further tech/product development and improvement.

Compared to the alternative cross-platform approaches available, Flutter poses minimal risks to a business and therefore is worth to be the number one choice for businesses.

### As with the enormously growing community across the globe, it’s safe to say that it’s the best Cross-Platform Development Technology and is the future of mobile development.

**7.** **Future Scope**

We continue to see fast growth in Flutter usage, with over **two million developers having used Flutter** in the last 3 years since its release. Despite these unprecedented circumstances, in March there was 10% month-over-month growth, with nearly half a million developers now using Flutter each month.

Some other interesting statistics:

* + 60% of users are developing with Windows, 27% are using macOS, and 13% are using Linux.
  + 35% work for a startup, 26% are enterprise developers, 19% are self-employed, and 7% work for design agencies.
  + The top five territories for Flutter are India, China, the United States, the EU, and Brazil.
  + There are approximately 90,000 Flutter apps published in the Play Store, with nearly 10,000 uploaded in the last month alone.

**8.** **Bibliography**

* Flutter Complete Reference: Create Beautiful, Fast and Native Apps for Any Device - Alberto Miola
* [Flutter documentation - Flutter](https://flutter.dev/docs) (flutter.dev/docs)
* <https://www.tutorialspoint.com/flutter/index.html>
* <https://medium.com/flutterdevs>