

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and
Technology (Deemed to be University)

Submitted in partial fulfillment of the requirements for
the award of Bachelor of Technology degree in Information
Technology

By
B.VAMSHI
39120130



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND
TECHNOLOGY

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

NOVEMBER 2021



SATHYABAMA
INSTITUTE OF SCIENCE AND
TECHNOLOGY
(DEEMED TO BE
UNIVERSITY)



Accredited with Grade
“A” by NAAC

(Established under Section 3 of UGC Act,
1956)

JEPPIAAR NAGAR, RAJIV GANDHI
SALAI

CHENNAI– 600119

www.sathyabama.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafideonafide work of **B.VAMSHI (39120130)** who carried out the project entitled **“Android mobile application with flutter frame work”** under my supervision from June 2021 to November 2021.

Internal Guide

Head of the Department

Submitted for Viva voce Examination held on _____

InternalExaminer

ExternalExaminer

DECLARATION

I **B.VAMSHI** hereby declare that the project report entitled “**Android mobile application with flutter frame work**” done by me under the guidance of **Dr. Mary Gladence**.is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in information technology

DATE:

PLACE:

SIGNATURE OF CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. SUBHASHINI, M.E., Ph.D., Heads of the Department** of **INFORMATION TECHNOLOGY** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.G.MATHIVANAN** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of INFORMATION TECHNOLOGY** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE

CIN: U74999TG2016PTC112234



TechWebLabs

CERTIFICATE OF COURSE COMPLETED

THIS CERTIFICATE IS PRESENTED TO:

B. Vamshi

Flutter Development Course

09th November 2021

Krishna Sai

TRAINER'S SIGNATURE

CH. Sharath Chand

MANAGING DIRECTOR

Floor, Spline Arcade, Flat no 102, Plot no 1208, Ayyappa Society, Madhapur, Telangana 500081

ABSTRACT

A mobile application has to be able to keep up with heavy demands to compete with all the new applications that are developed each day. Good performance and nice visuals are base requirements for the development of mobile applications. There are many options for tools when developing and one of these choices is a native application, which is said to have better performance and suitability to the mobile environment. Another choice is a tool which requires only one code base for multiple platforms and is therefore easier to maintain. Flutter is an open-source User Interface (UI) toolkit created by Google that can create cross-platform applications with one code base while said to maintain the aspects of looking native. This paper explores how Flutter compares to native applications, which are currently seen as superior in mobile behaviour and performance. An experiment was conducted to test how Flutter as a cross-compiler compared to two native applications made of kotlin and Android studio and swift and XCode, in terms of CPU performance. A survey was created to see if there was a difference in the perception of users with regards to appearance and animations. A literature study was conducted to strengthen the results from the experiment and survey and to give a background to the subject. Flutter is a new tool and it continues to grow incredibly fast. Conclusions are drawn that a Flutter application can compete with a native application when it comes to CPU performance, but is not as developed in the animation area. Flutter does not require complex code for creating a simple application and uses significantly less lines of code in development compared to native. The final conclusion is that Flutter is best to use when building smaller to medium-sized applications, but has a potential to grow to overcome its current drawbacks in the animation department. Further examination of the areas examined in this paper is needed in order to ensure and strengthen the results.

(i)

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No
	ABSTRACT	i
	LIST OF FIGURES	
1	INTRODUCTION	1
	1.1	2
	1.2	3
2.	AIM AND SCOPE OF THE PRESENT INVESTIGATION	5
	2.1	5
	2.2	8
	2.3	11
	3. EXPERIMENTAL OR MATERIALS AND METHODS, ALGORITHMS USED	9
	3.1	9
	3.2	10
	3.3	11
	3.4	12
4.	RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS	13
	4.2	9
	7	
	7	

5	SUMMARY AND CONCLUSIONS	14
	REFERENCES	15
	•	
	•	
	•	
	Related images	16

INTRODUCTION

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 / Swift language.

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 catalog.
- Runs same UI for multiple platforms.
- High performance application.

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.

We shall discuss the complete architecture in the coming chapters.

In this section, let us see how to install *Flutter SDK* and its requirement in a windows system.

Step 1: Go to URL, <https://flutter.dev/docs/get-started/install/windows> and download the latest *Flutter SDK*. As of April 2019, the version is 1.2.1 and the file is flutter_windows_v1.2.1-stable.zip.

Step 2: Unzip the zip archive in a folder, say C:\flutter\

Step 3: Update the system path to include flutter bin directory.

Step 4: Flutter provides a tool, flutter doctor to check that all the requirement of flutter development is met.

```
flutter doctor
```

```
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, v1.2.1, on Microsoft Windows
[Version10.0.17134.706], locale en-US)
[✓] Android toolchain - develop for Android devices (Android SDK
version28.0.3)
[✓] Android Studio (version 3.2)
[✓] VS Code, 64-bit edition (version 1.29.1)
[!] Connected device
    ! No devices available
```

The report says that all development tools are available but the device is not connected. We can fix this by connecting an android device through USB or starting an android emulator.

Step 6: Install the latest Android SDK, if reported by flutter doctor

Step 7: Install the latest Android Studio, if reported by flutter doctor

Step 8: Start an android emulator or connect a real android device to the system.

Step 9: Install Flutter and Dart plugin for Android Studio. It provides startup template to create new Flutter application, an option to run and debug Flutter application in the Android studio itself, etc..

- Open Android Studio.
- Click File > Settings > Plugins.
- Select the Flutter plugin and click Install.
- Click Yes when prompted to install the Dart plugin.
- Restart Android studio.

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 catalog.
- Runs same UI for multiple platforms.

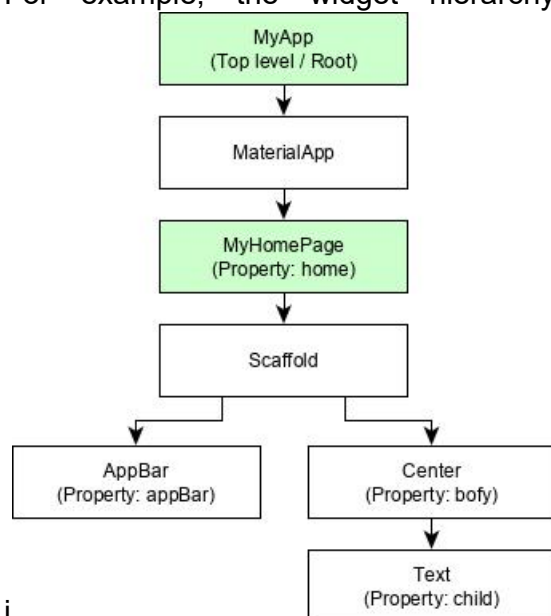
High performance application

Widgets

The core concept of the Flutter framework is **In Flutter, Everything is a widget**. Widgets are basically user interface components used to create the user interface of the application.

In *Flutter*, the application is itself a widget. The application is the top-level widget and its UI is built using one or more children (widgets), which again build using its children widgets. This **composability** feature helps us to create a user interface of any complexity.

For example, the widget hierarchy of the hello world application (created



i chapter) is as specified in the following diagram:

n previous

Layouts in Flutter:

The core of Flutter's layout mechanism is widgets. In Flutter, almost everything is a widget—even layout models are widgets. The images, icons, and text that you see in a Flutter app are all widgets. But things you don't see are also widgets, such as the rows, columns, and grids that arrange, constrain, and align the visible widgets.

AppBar class:

A material design app bar.

An app bar consists of a toolbar and potentially other widgets, such as a [TabBar](#) and a [FlexibleSpaceBar](#). App bars typically expose one or more common [actions](#) with [IconButton](#)s which are optionally followed by a [PopupMenuButton](#) for less common operations (sometimes called the "overflow menu").

App bars are typically used in the [Scaffold.appBar](#) property, which places the app bar as a fixed-height widget at the top of the screen. For a scrollable app bar, see [SliverAppBar](#), which embeds an [AppBar](#) in a sliver for use in a [CustomScrollView](#).

Flutter Buttons

Buttons are the graphical control element that **provides a user to trigger an event** such as taking actions, making choices, searching things, and many more. They can be placed anywhere in our UI like dialogs, forms, cards, toolbars, etc.

Buttons are the Flutter widgets, which is a part of the material design library. Flutter provides several types of buttons that have different shapes, styles, and features.

3

Drawer:

In apps that use Material Design, there are two primary options for navigation: tabs and drawers. When there is insufficient space to support tabs, drawers provide a handy alternative.

Flutter – Carousel Slider:

Carousel Slider is one of the most popular image slider used nowadays in most apps. These Carousel Sliders are mostly seen in various eCommerce sites such as Amazon, Flipkart, Myntra, and many more. Displaying the images in a slider gives an attractive User Experience. As these sliders are automated you can get to see various types of images and content in them.

List View class:

[ListView](#) is the most commonly used scrolling widget. It displays its children one after another in the scroll direction. In the cross axis, the children are required to fill the [ListView](#).

If non-null, the [itemExtent](#) forces the children to have the given extent in the scroll direction.

If non-null, the [prototypeItem](#) forces the children to have the same extent as the given widget in the scroll direction.

Grid list:

In some cases, you might want to display your items as a grid rather than a normal list of items that come one after the next. For this task, use the [GridView](#) widget.

The simplest way to get started using grids is by using the [GridView.count\(\)](#) constructor, because it allows you to specify how many rows or columns you'd like.

Card class:

A material design card: a panel with slightly rounded corners and an elevation shadow.

A card is a sheet of [Material](#) used to represent some related information, for example an album, a geographical location, a meal, contact details, etc.

Bottom Navigation Bar class:

A material widget that's displayed at the bottom of an app for selecting among a small number of views, typically between three and five.

The bottom navigation bar consists of multiple items in the form of text labels, icons, or both, laid out on top of a piece of material. It provides quick navigation between the top-level views of an app. For larger screens, side navigation may be a better fit.

A bottom navigation bar is usually used in conjunction with a [Scaffold](#), where it is provided as the [Scaffold.bottomNavigationBar](#) argument.

The bottom navigation bar's [type](#) changes how its [items](#) are displayed. If not specified, then it's automatically set to [BottomNavigationBarType.fixed](#) when there are less than four items, and [BottomNavigationBarType.shifting](#) otherwise.

Text field:

Text fields allow users to type text into an app. They are used to build forms, send messages, create search experiences, and more. In this recipe, explore how to create and style text fields.

2.AIM AND SCOPE OF THE PRESENT INVESTIGATIONS:

What is Flutter?

Flutter is a free and open-source mobile UI framework created by Google and released in May 2017. In a few words, it allows you to create a native mobile application with only one codebase. This means that you can use one programming language and one codebase to create two different apps (for iOS and Android).

Flutter consists of two important parts:

- An SDK (Software Development Kit): A collection of tools that are going to help you develop your applications. This includes tools to compile your code into native machine code (code for iOS and Android).
- A Framework (UI Library based on widgets): A collection of reusable UI elements (buttons, text inputs, sliders, and so on) that you can personalize for your own needs.

To develop with Flutter, you will use a programming language called Dart. The language was created by Google in October 2011, but it has improved a lot over these past years.

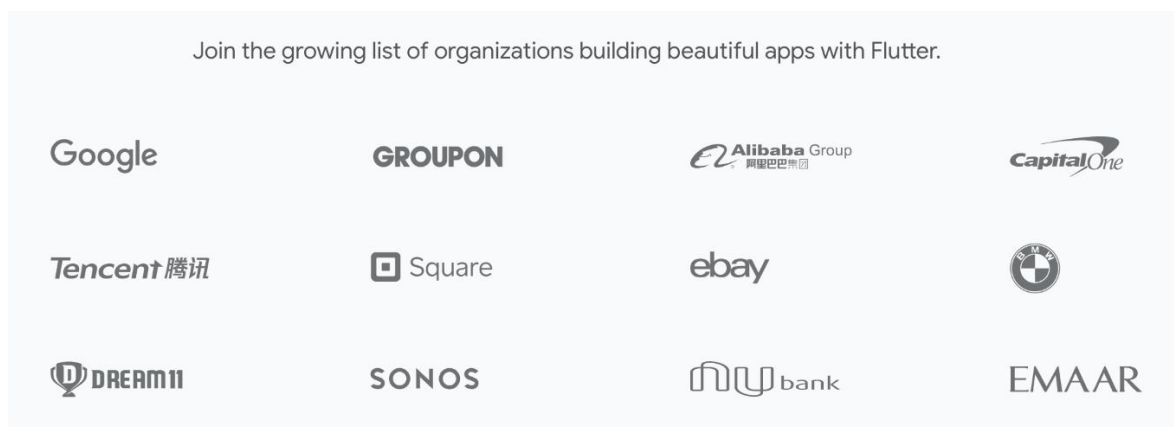
Dart focuses on front-end development, and you can use it to create mobile and web applications.

If you know a bit of programming, Dart is a typed object programming language. You can compare Dart's syntax to JavaScript.

“Flutter is Google’s UI toolkit for building beautiful, natively compiled applications for mobile, web, and desktop from a single codebase.” - Google, flutter.dev.

Why you should learn Flutter?

I selected some of the reasons why I like Flutter and why I want to use it next year. I will give you details and my feedback below.



Companies using

Flutter

Simple to learn and use:

Flutter is a modern framework, and you can feel it! It's way simpler to create mobile applications with it. If you have used Java, Swift, or React Native, you'll notice how Flutter is different.

I personally never liked mobile application development before I started using Flutter.

What I love about Flutter is that you can create a real native application without a bunch of code.

Quick compilation: maximum productivity:

Thanks to Flutter, you can change your code and see the results in real-time. It's called Hot-Reload. It only takes a short amount of time after you save to update the application itself.

Significant modifications force you to reload the app. But if you do work like design, for example, and change the size of an element, it's in real-time!

Ideal for startup MVPs:

If you want to show your product to investors as soon as possible, Flutter is a good choice.

Here are my top 4 reasons to use it for your MVP:

- It's cheaper to develop a mobile application with Flutter because you don't need to create and maintain two mobile apps (one for iOS and one for Android).
- One developer is all you need to create your MVP.
- It's performant – you won't notice the difference between a native application and a Flutter app.
- It's beautiful – you can easily use widgets provided by Flutter and personalize it to create a valuable UI for your customers (you can find examples of applications made with Flutter below).

Good documentation:

It's important for new technology to have good documentation. But it's not always the case that it has it!

You can learn a lot from Flutter's documentation, and everything is very detailed with easy examples for basic use cases. Each time I've had a problem with one of my widgets in my code, I have been able to check the documentation and the answer was there.

A growing community:

Flutter has a robust community, and it's only the beginning!

As you may know, I love to share my knowledge and useful content on programming on my website. I need to know I'm working on a technology full of potential with a lot of backers.

When I started using Flutter, the first thing I did was search for communities, and to my surprise... there are a considerable number of places to exchange info on Flutter.

I will give you some examples of places I love to check daily. Feel free to send me a message on Twitter with your suggestions.

- Flutter Awesome: An awesome list that curates the best Flutter libraries and tools. This website publishes daily content with lots of examples, application templates, advice, and so on.
- Awesome Flutter: A GitHub repository (linked to Flutter Awesome) with a list of articles, videos, components, utilities, and so on.
- It's all widgets!: An open list of apps built with Flutter.
- Flutter Community: A Medium publication where you can find articles, tutorials, and much more.

Supported by Android Studio and VS Code:

Flutter is available on different IDEs. The two main code editors for developing with this technology are Android Studio (IntelliJ) and VS Code.

Android Studio is a complete software with everything already integrated. You have to download Flutter and Dart plugins to start.

VS Code is a lightweight tool, and everything is configurable through plugins from the marketplace.

I use Android Studio because I don't need to configure a lot of things to work.

You are free to choose your preferred IDE!

Freelance:

If you want to start doing some freelance work, you should think about using Flutter.

In 2020, I believe that this technology is going to explode. And that means a lot of people are going to search for developers who know how to use it.

The biggest platform for freelancers in France, called Malt, recently published the tech trends of this year. Flutter has grown by +303% on this platform between 2018 and 2019.

Advantages of Flutter:

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.

Disadvantages of Flutter:

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

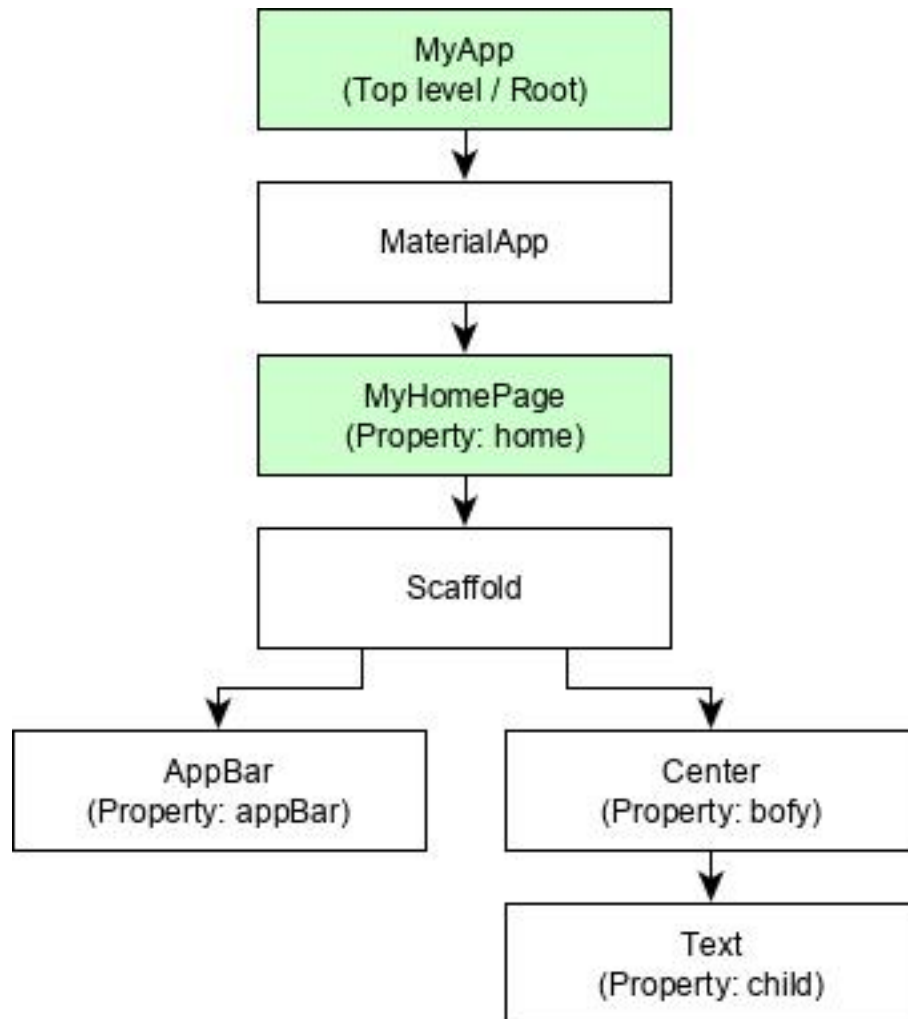
- Since it is coded in Dart language, a developer needs to learn new language (though it is easy to learn).
- Modern framework tries to separate logic and UI as much as possible but, in Flutter, user interface and logic is intermixed. We can overcome this using smart coding and using high level module 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.

3. EXPERIMENTAL OR MATERIALS AND METHODS, ALGORITHMS uSED:

Widgets: The core concept of the Flutter framework is **In Flutter, Everything is a widget**. Widgets are basically user interface components used to create the user interface of the application.

In *Flutter*, the application is itself a widget. The application is the top- level widget and its UI is build using one or more children (widgets), which again build using its children widgets. This **composability** feature helps us to create a user interface of any complexity.

For example, the widget hierarchy of the hello world application (created in previous chapter) is as specified in the following diagram:



Here the following points are worth notable:

- *MyApp* is the user created widget and it is build using the Flutter native widget, *MaterialApp*.
- *MaterialApp* has a *home* property to specify the user interface of the home page, which is again a user created widget, *MyHomePage*.
- *MyHomePage* is build using another flutter native widget, *Scaffold*.
- *Scaffold* has two properties – *body* and *appBar*.
- *body* is used to specify its main user interface and *appBar* is used to specify its header user interface.
- *Header UI* is build using flutter native widget, *AppBar* and *Body UI* is build using *Center* widget.
- The *Center* widget has a property, *Child*, which refers the actual content and it is build using *Text* widget.

Gestures

Flutter widgets support interaction through a special widget, *GestureDetector*. *GestureDetector* is an invisible widget having the ability to capture user interactions such as tapping, dragging, etc., of its child widget. Many native widgets of Flutter support interaction through the use of *GestureDetector*. We can also incorporate interactive feature into the existing widget by composing it with the *GestureDetector* widget. We will learn the gestures separately in the upcoming chapters.

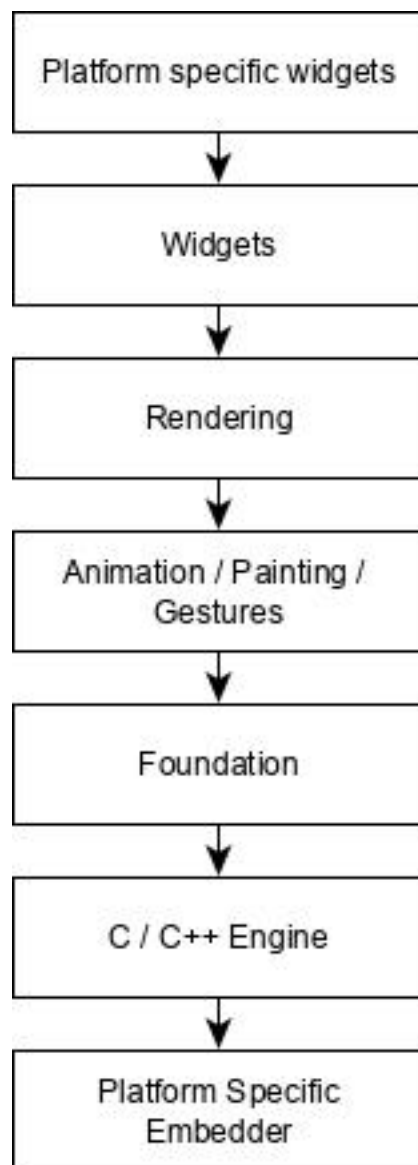
Concept of State

Flutter widgets support *State maintenance* by providing a special widget, *StatefulWidget*. Widget needs to be derived from *StatefulWidget* widget to support state maintenance and all other widget should be derived from *StatelessWidget*. Flutter widgets are **reactive** in nature. This is similar to *reactjs* and *StatefulWidget* will be auto re- rendered whenever its internal state is changed. The re-rendering is optimized by finding the difference between old and new widget UI and rendering only the necessary changes.

Layers

The most important concept of Flutter framework is that the framework is grouped into multiple category in terms of complexity and clearly arranged in layers of decreasing complexity. A layer is build using its immediate next level layer. The top most layer is widget specific to *Android* and *iOS*. The next layer has all flutter native widgets. The next layer is *Rendering* layer, which is low level renderer component and renders everything in the flutter app. Layers goes down to core platform specific code.

The general overview of a layer in Flutter is specified in the below diagram:



The following points summarize the architecture of Flutter:

- In Flutter, everything is a widget and a complex widget is composed of already existing widgets.
- Interactive features can be incorporated whenever necessary using *GestureDetector* widget.
- The state of a widget can be maintained whenever necessary using *StatefulWidget*.
- Flutter offers layered design so that any layer can be programmed depending on the complexity of the task.

We will discuss all these concepts in detail in the upcoming chapters.

RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS:

This section displays the results that were discovered through the experiment and literature review

Looking at the results as a whole, Flutter wins the majority of most categories in the development area. There are however some differences that are interesting to take note of when comparing Flutter to native builds.

A part of answering Q3 is comparing the code complexity of the development code of each of the applications. The part of the application that was chosen for this was the creating of the notification list. It was chosen in particular because it contained the most code that was written and because the other view only featured an image and a title. The code that is shown in this section is only a part of the application code bases. As shown in the results for the first question of the survey, the majority had used or had owned an android at least once. This was positive because android phones were used in the examples of the survey. The results that revolves around looks show that even though most of the people thought that both looked like native, there was a slightly bigger percentage that thought app A looked more native over app B and a small amount that found none of the applications as native. The survey takers found no difference to a small difference between the applications. This shows that Flutter can mimic the looks of a native application without affecting the users too much. An ideal addition to the survey, would have been to use the same application in iOS to compliment the survey that only used the android applications, but this had to be omitted due to time constraints and delayed access to equipment. Individual answers reinforced the results by motivating these differences below. Individual answers that had nothing to do with these were small issues with centering and size that could be mitigated with resources Flutter can supply, hence the other answers were sorted out. • Not using full screen entirely • Standard native font and native spacing An issue that could be a problem regarding the images and videos that were used in the survey is that differences in padding between the applications are different. This could have created confusion around the amount of notification bars in the list. However, looking at the individual answers regarding looks, results show that the survey takers notice the differences in padding but do not mention the possibility of different numbers of notifications. Looking at the other part of the survey that revolves around animation and behaviour, there were similar answers to the first question regarding looks.

SUMMARY AND CONCLUSIONS:

Flutter is a useful toolkit that enables easy ways of creating new applications. It has gotten more and more popular recently and is talked about in the application development industry as a possible replacement of React Native and how it can be compared with native applications. The experiment of this report revealed that there is a small difference between the CPU performance of Flutter iOS and native Swift iOS respective Flutter android and flutter native. There seem to be a difference between the performance of iOS and android but when it comes to how well Flutter can perform in CPU usage compared to native applications, there is barely a difference. The summery for this is that Flutter can perform up to par with a native application for the type of application size that was tested. To verify these results and determine that Flutter can keep up with native, further testing needs to be carried out. There are multiple other metrics that are needed in order to fully display how Flutter compares to native applications in performance. When it came down to code size, the Flutter application had a low amount of code needed (125 lines) compared to iOS (363 lines) and android (217 lines). Flutter do not split its functionality code and visual code which lowers the amount of code and files. The Flutter code had similarities to web languages and often only required replicating to create new parts. The complexity of code that is needed for flutter, is simpler than the code that was needed for the native builds. This can however, not be fully concluded since additional methods of measuring code complexity has to be carried out for a better answer regarding the complexity and code size of all the code bases. Appearance wise, Flutter and native seem to differentiate little to a majority of users. It is able to mimic the native looks and behaviour to a certain point. The biggest difference showed to be around the behaviour and animations of the application where the difference was noticeable on items such as list, menu, default spacing and font. Animations are however something that can be added according their documentation[40] although it requires more of the developer and application as it i

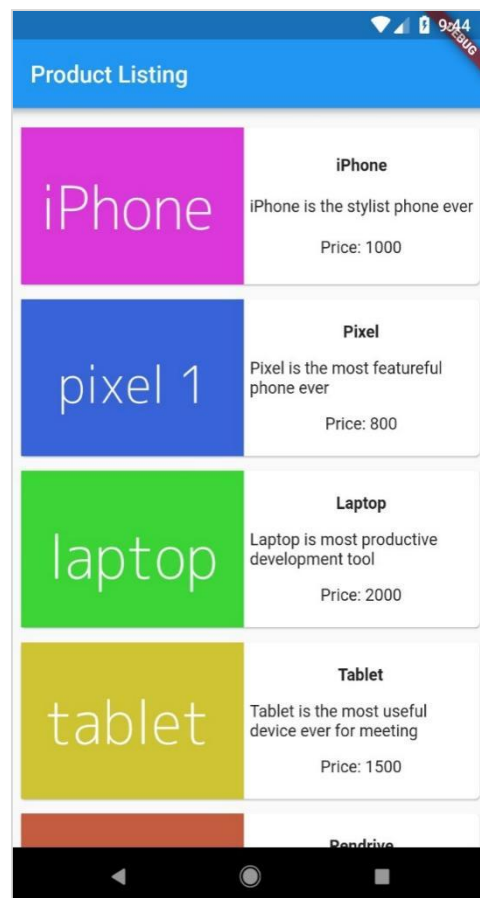
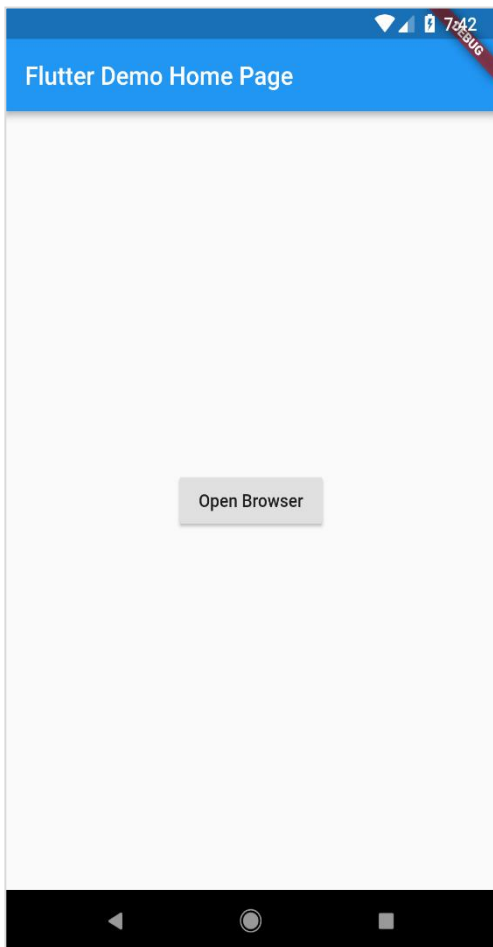
References

REFERENCES:

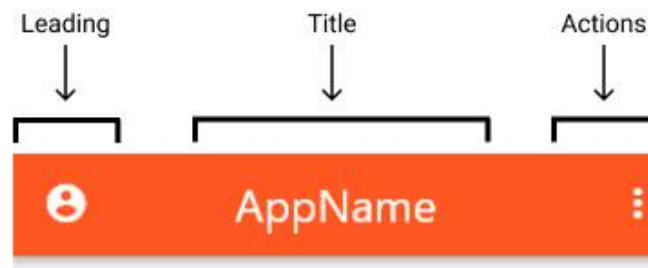
- [1] O. Lifh and P. Lindholm, Recreating a native application in react native, 2018. doi: diva2:1213248.
- [2] A. Smith. (2019). Why businesses should start focusing on google's flutter and fuchsia, [Online]. Available: [https://medium.com/swlh/whybusinesses - should - start - focusing - on - googles - flutter - and - fuchsia-48e16820f2a9](https://medium.com/swlh/whybusinesses-should-start-focusing-on-googles-flutter-and-fuchsia-48e16820f2a9) (visited on 01/21/2019).
- [3] M. Palmieri, I. Singh, and A. Cicchetti, "Comparison of cross-platform mobile development tools," 2012 16th International Conference on Intelligence in Next Generation Networks, Researchgate.net, 2012, p. 8.
- [4] S. Xanthopoulos and S. Xinogalos, "A comparative analysis of crossplatform developement approaches for mobile applications," University of Macedonia, academia.edu, 2013, p. 7. [Online]. Available: <https://dl.acm.org/doi/abs/10.1145/2490257.2490292>.
- [5] Google. (2019). Google flutter, [Online]. Available: <https://flutter.dev/> (visited on 12/03/2019).
- [6] B. D. Coninck. (2019). Flutter versus other development frameworks: A ui and performance experiment, [Online]. Available: [https : / / blog . codemagic . io / flutter - vs - ios - android - reactnative - xamarin/](https://blog.codemagic.io/flutter-vs-ios-android-reactnative-xamarin/) (visited on 02/06/2020).
- [7] M. Bellinaso. (2018). Flutter: The good, the bad and the ugly, [Online]. Available: <https://medium.com/asos-techblog/flutter-vs-reactnative-for-ios-android-app-development-c41b4e038db9> (visited on 02/06/2020).
- [8] H. Heitkötter, S. Hanschke, and T. A. Majchrzak, "Comparing crossplatform development approaches for mobile applications," Scitpress.org, <https://www.scitepress.org/Papers/2012/39045/39045.pdf>: Department of Information systems, University of Münster, 2012, p. 13.
- [9] L. Corral, A. Sillitti, and G. Succi, "Mobile multiplatform development: An experiment for performance analysis," Free University of Bozen-Bolzano, Piazza Domenicani 3, 39100 Bozen-Bolzano, Italy, diva2:664680: www.sciencedirect.com, 2011, p. 8. [10] S. Amatyia, "Cross-platform mobile development: An alternative to native mobile development," Linnéuniversitetet, Fakulteten för teknik (FTK), Institutionen för datavetenskap (DV), diva2:664680: Diva, 2013, p. 67.s not something that is accompanied by the default in Flutter.

Related images:

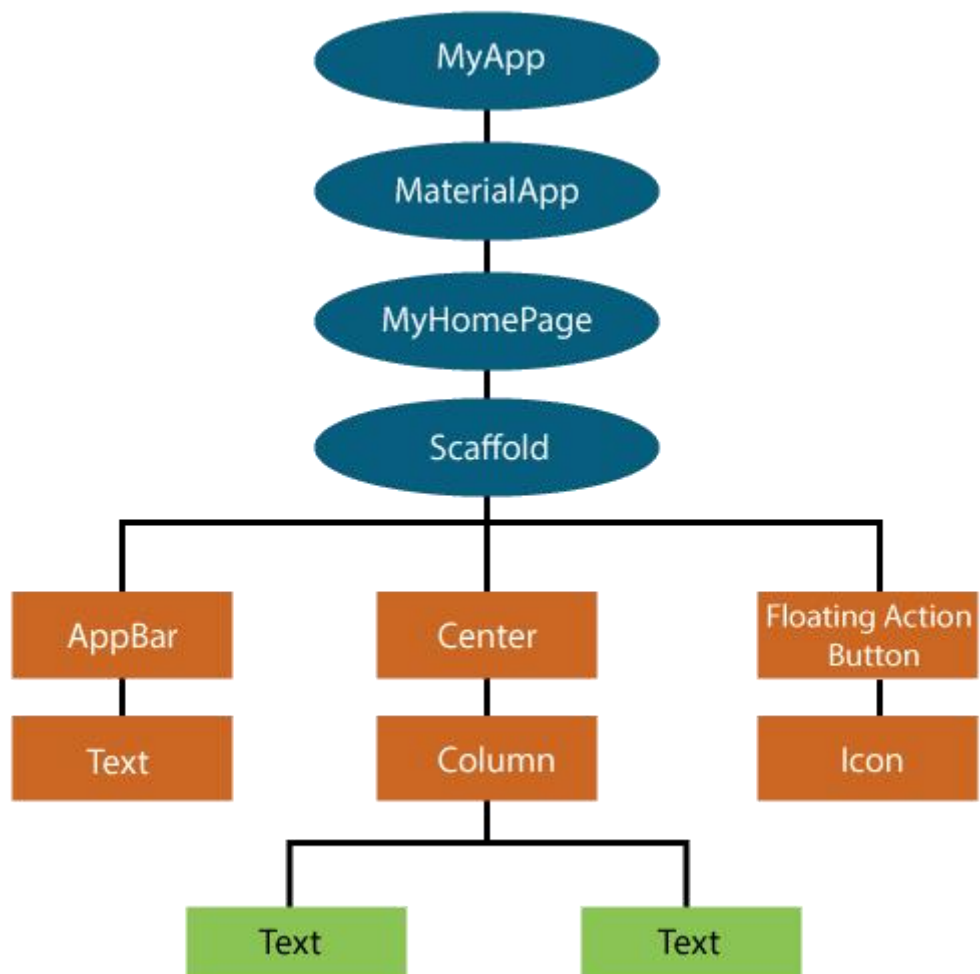
this is related to button function,



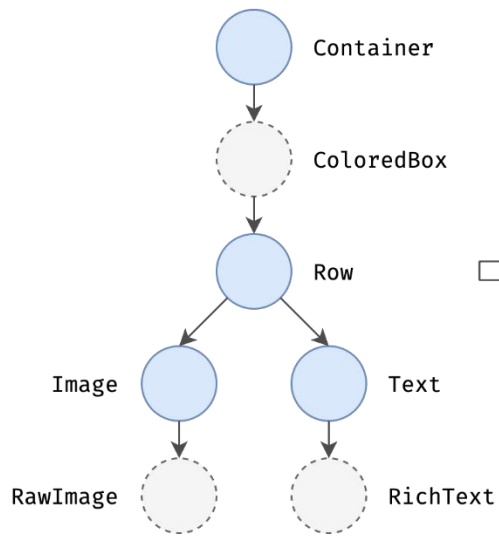
This is related for list view example:
This is related for app bar



example:



Widgets



Element Tree

