IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

FLUTTER

[1]Prof. Navnath Ladke, [2] Prof. Dhanashri Vedpathak , [3] Prof.R.R.Saraogi

[2] Department of AI&ML , ISBM College of Engineering, Pune, Maharashtra, India.

[1,3] Department of Applied Science, ISBM College of Engineering, Pune, Maharashtra, India.

ABSTRACT

In recent years, it is difficult to develop applications for both iOs and Android with in less time. To overcome this, Google introduce a new framework called Flutter. It is a new reactive framework and platform for building high performance and beautiful mobile apps . It is useful extensively at google to build business-critical apps, and by third party developers to build popular apps .It is also used as a SDK which provides support to build beautiful mobile apps in record time. Flutter is highly customizable , which allows it to build apps that are brand centric, or with the look and feel of native Android and iOs apps froma single code base.

Keywords – Flutter, SDK, Highly customizable, Android, iOs.

I. 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. HOWEVER, TO DEVELOP AN APPLICATION SUPPORTING BOTH THE OSS, WE NEED TO CODE IN TWO DIFFERENT LANGUAGES USING TWO DIFFERENT FRAMEWORKS.

Flutter is Google's free, open-source software development kit (SDK) for cross-platform mobile application development. Using a single platform-agnostic codebase, Flutter helps developers build high-performance, *scalable applications with attractive and functional user interfaces for Android or IOS. Flutter relies on a library of pre-made widgets* that make it simple for even people with limited programming or development experience to launch their own mobile applications quickly. Created by Google in 2015 and officially launched in 2018, Flutter has quickly become the toolkit of choice developers. According to Statistics, Flutter has recently surpassed React Native to become the number one mobile app development framework.

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 WIDGETS.

II. ARCHITECTURE

THE FLUTTER FRAMEWORK IS ORGANIZED INTO A SERIES OF LAYERS, WITH EACH LAYER BUILDING UPON THE PREVIOUS LAYERS.

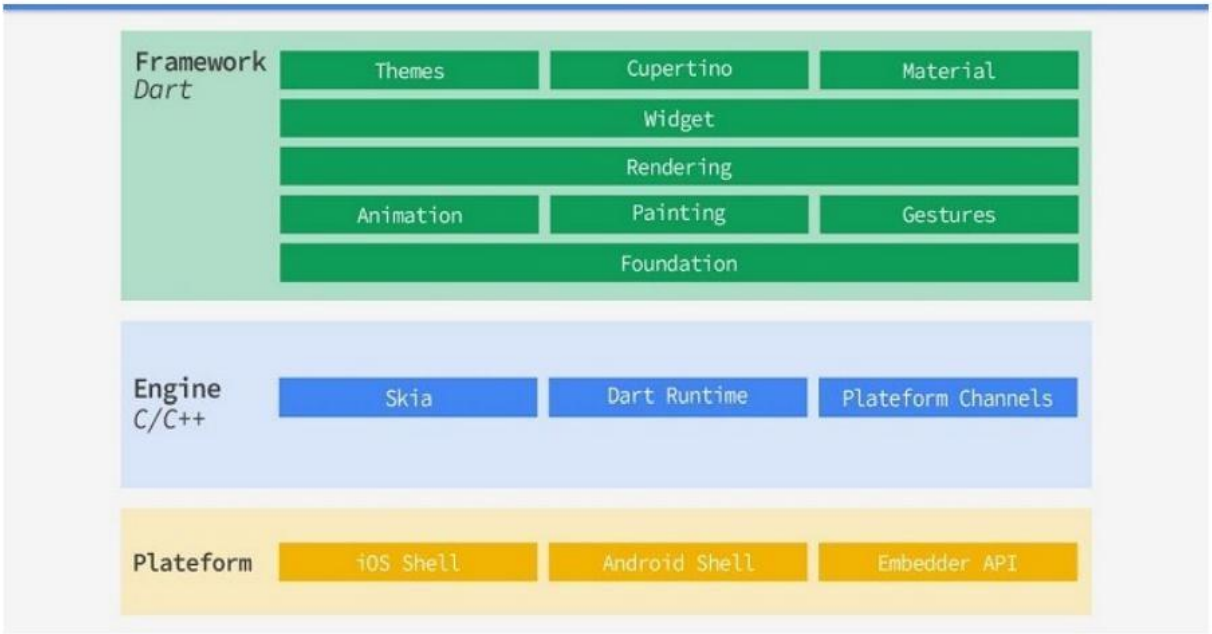

FIG 4.1.1 ARCHITECTURE OF FLUTTER

TO THE UNDERLYING OPERATING SYSTEM, FLUTTER APPLICATIONS ARE PACKAGED IN THE SAME WAY AS ANY OTHER NATIVE APPLICATION. A PLATFORM-SPECIFIC EMBEDDER PROVIDES AN ENTRY POINT; 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:

DART FRAMEWORK FLUTTER APPS ARE WRITTEN IN THE DART LANGUAGE AND MAKE USE OF MANY OF THE LANGUAGE'S MORE ADVANCED FEATURES. ON ANDROID, AND ON WINDOWS, MACES AND LINUX VIA THE SEMI-OFFICIAL FLUTTER DESKTOP EMBEDDING PROJECT, FLUTTER RUNS IN THE DART VIRTUAL MACHINE WHICH FEATURES A JUST-IN-TIME

EXECUTION ENGINE. DUE TO APP STORE RESTRICTIONS ON DYNAMIC CODE EXECUTION, FLUTTER APPS USE AHEAD OF TIME (AOT) COMPILATION ON IOS. A NOTABLE FEATURE OF THE DART PLATFORM IS ITS SUPPORT FOR "HOT RELOAD" WHERE MODIFICATIONS TO SOURCE FILES CAN BE INJECTED INTO A RUNNING APPLICATION. FLUTTER EXTENDS THIS WITH SUPPORT FOR STATEFUL HOT RELOAD, WHERE IN MOST CASES CHANGES TO SOURCE CODE CAN BE REFLECTED IMMEDIATELY IN THE RUNNING APP WITHOUT REQUIRING A RESTART OR ANY LOSS OF STATE.

FLUTTER ENGINE:

FLUTTER'S ENGINE, WRITTEN PRIMARILY IN C++, PROVIDES LOW-LEVEL RENDERING SUPPORT USING GOOGLE'S SKIA GRAPHICS LIBRARY. ADDITIONALLY, IT INTERFACES WITH PLATFORM-SPECIFIC SDKS SUCH AS THOSE PROVIDED BY ANDROID

AND IOS. THE FLUTTER ENGINE IS A PORTABLE RUNTIME FOR HOSTING FLUTTER APPLICATIONS. IT IMPLEMENTS FLUTTER ‘SCORE LIBRARIES, INCLUDING ANIMATION AND GRAPHICS, FILE AND NETWORK I/O, ACCESSIBILITY SUPPORT, PLUGIN ARCHITECTURE, AND A DART RUNTIME AND COMPILE TOOLCHAIN. MOST DEVELOPERS WILL INTERACT WITH FLUTTER VIA THE FLUTTER FRAMEWORK, WHICH PROVIDES A MODERN, REACTIVE FRAMEWORK, AND A RICH SET OF PLATFORM LAYOUT AND APPLICATION WIDGETS.

PLATFORM:

AT THE PLATFORM LEVEL, FLUTTER PROVIDES A SHELL, THAT HOSTS THE DART VM. THE SHELL, IS PLATFORM SPECIFIC, GIVING ACCESS TO THE NATIVE PLATFORM APIS AND HOSTING THE ESTABLISHING THE PLATFORM RELEVANT CANVAS. THERE

IS ALSO AN EMBEDDER API, IF YOU WANT TO USE FLUTTER LIKE A LIBRARY, INSTEAD OF HOSTING RUNNING AN APP. THE SHELLS, ALSO HELP PROVIDE COMMUNICATION TO THE RELEVANT IMES AND THE SYSTEMS APPLICATION LIFECYCLE EVENT.

III. WIDGETS

EVERYTHING IN FLUTTER IS A WIDGET. THIS INCLUDES USER INTERFACE ELEMENTS, SUCH AS LIST VIEW, TEXTBOX, AND IMAGE, AS WELL AS OTHER PORTIONS OF THE FRAMEWORK, INCLUDING LAYOUT, ANIMATION, GESTURE RECOGNITION, AND THEMES, TO NAME JUST A FEW. WIDGETS ARE NECESSARY FOR AN APP'S VIEW AND INTERFACE. THEY MUST HAVE A NATURAL LOOK AND FEEL REGARDLESS OF SCREEN SIZE. THEY ALSO MUST BE FAST, EXTENSIBLE, AND CUSTOMIZABLE. FLUTTER TAKES

THE EVERYTHING'S A WIDGET APPROACH. IT HAS A RICH SET OF WIDGETS AND EXTENSIVE CAPABILITIES FOR CREATING COMPLEX CUSTOM WIDGETS.

IN FLUTTER, WIDGETS AREN'T ONLY USED FOR VIEWS. THEY'RE ALSO USED FOR ENTIRE SCREENS AND EVEN FOR THE APP ITSELF. AS FLUTTER'S DOCUMENTATION PUTS IT, EACH WIDGET IS AN IMMUTABLE DECLARATION OF PART OF THE USER INTERFACE. OTHER FRAMEWORKS SEPARATE VIEWS, VIEW CONTROLLERS, LAYOUTS, AND OTHER PROPERTIES. FLUTTER, ON THE OTHER HAND, HAS A CONSISTENT, UNIFIED OBJECT MODEL: THE WIDGET. A WIDGET CAN DEFINE A STRUCTURAL ELEMENT (LIKE A BUTTON OR MENU); A STYLISTIC ELEMENT (LIKE A FONT OR COLOR SCHEME); AN ASPECT OF THE LAYOUT (LIKE PADDING); AND SO ON. WIDGETS FORM A HIERARCHY BASED ON THEIR COMPOSITION. EACH WIDGET NESTS INSIDE OF AND INHERITS PROPERTIES FROM ITS PARENT. THERE'S NO SEPARATE APPLICATION OBJECT. INSTEAD, THE ROOT WIDGET SERVES THIS ROLE. FLUTTER HAS A FULL SET OF WIDGETS IN GOOGLE'S MATERIAL DESIGN AND IN APPLE'S STYLE WITH THE CUPERTINO PACK. WIDGET RENDERING HAPPENS DIRECTLY IN THE SKI ENGINE WITHOUT US ORIGINAL EQUIPMENT MANUFACTURER WIDGETS. SO WE GET A SMOOTHER UI EXPERIENCE COMPARED WITH OTHER CROSS PLATFORM FRAMEWORK.

STATELESS WIDGETS: A STATELESS WIDGET IS A WIDGET THAT DESCRIBES PART OF THE USER INTERFACE BY BUILDING A CONSTELLATION OF OTHER WIDGETS THAT DESCRIBE THE USER INTERFACE MORE CONCRETELY. THE BUILDING PROCESS

CONTINUES RECURSIVELY UNTIL THE DESCRIPTION OF THE USER INTERFACE IS FULLY CONCRETE.

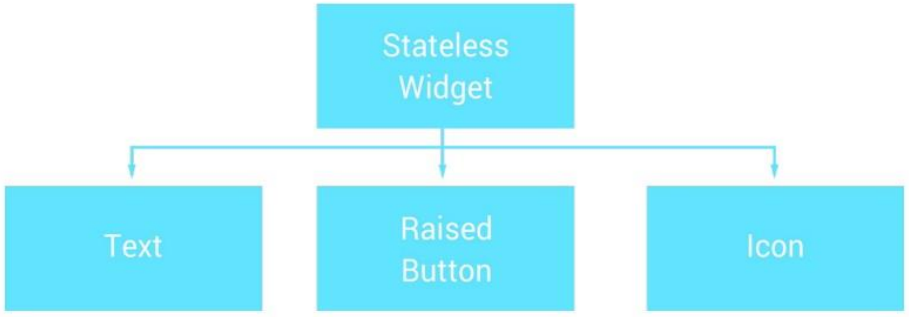

FIG5.1.1.1 STATELESS WIDGETS

STATELESS WIDGET ARE USEFUL WHEN THE PART OF THE USER INTERFACE YOU ARE DESCRIBING DOES NOT DEPEND ON ANYTHING OTHER THAN THE CONFIGURATION INFORMATION IN THE OBJECT ITSELF AND THE BUILD CONTEXT IN WHICH THE
WIDGET IS INFLATED. FOR COMPOSITIONS THAT CAN CHANGE DYNAMICALLY, E.G. DUE TO HAVING AN INTERNAL CLOCK- DRIVEN STATE, OR DEPENDING ON SOME

SYSTEM STATE, CONSIDER USING STATEFUL WIDGETS. THE TEXT WIDGET IS INSTANTIATED USING A CONSTRUCTOR AND THEN THESE PROPERTIES ARE USED TO BUILD THE WIDGET TO BE DISPLAYED ON THE SCREEN. IF A WIDGET'S PARENT WILL REGULARLY CHANGE THE WIDGET'S CONFIGURATION, OR IF IT DEPENDS ON INHERITED WIDGETS THAT FREQUENTLY CHANGE, THEN IT IS IMPORTANT TO OPTIMIZE THE PERFORMANCE OF THE BUILD METHOD TO MAINTAIN A FLUID RENDERING PERFORMANCE. SO, A STATELESS WIDGET IS NOT DYNAMIC. IT DOESN'T DEPEND ON ANY DATA OTHER THAN THAT THAT IS PASSED INTO IT, MEANING THAT THE ONLY WAY IT CAN SET AND HOW IT IS REPRESENTED IS WHEN ARGUMENTS ARE PASSED INTO ITS CONSTRUCTOR.

IV. PLATFORM COMPARISON

1. APPLE OR ANDROID NATIVE

NATIVE APPLICATIONS OFFER THE LEAST FRICTION IN ADOPTING NEW FEATURES. THEY TEND TO HAVE USER EXPERIENCES MORE IN TUNE WITH THE GIVEN PLATFORM SINCE THE APPLICATIONS ARE BUILT USING CONTROLS FROM THE PLATFORMS VENDORS THEMSELVES (APPLE OR GOOGLE) AND OFTEN FOLLOW DESIGN GUIDELINES SET OUT BY THESE VENDORS. ONE BIG ADVANTAGE NATIVE APPLICATIONS HAVE IS THEY CAN ADOPT BRAND NEW TECHNOLOGIES APPLE AND GOOGLE CREATE IN BETA IMMEDIATELY IF DESIRED, WITHOUT HAVING TO WAIT FOR ANY THIRD-PARTY INTEGRATION. THE MAIN DISADVANTAGE TO BUILDING NATIVE APPLICATIONS IS THE LACK OF CODE REUSE ACROSS PLATFORMS, WHICH CAN MAKE DEVELOPMENT EXPENSIVE IF TARGETING IOS AND ANDROID.

2. REACT NATIVE:

REACT NATIVE ALLOWS NATIVE APPLICATIONS TO BE BUILT USING JAVASCRIPT. THE ACTUAL CONTROLS THE APPLICATION USES ARE NATIVE PLATFORM CONTROLS, SO THE END USER GETS THE FEEL OF A NATIVE APP. FOR APPS THAT REQUIRE CUSTOMIZATION BEYOND WHAT REACT NATIVE'S ABSTRACTION PROVIDES, NATIVE DEVELOPMENT COULD STILL BE NEEDED. IN CASES WHERE THE AMOUNT OF CUSTOMIZATION REQUIRED IS SUBSTANTIAL, THE BENEFIT OF WORKING WITHIN REACT NATIVE'S ABSTRACTION LAYER LESSENS TO THE POINT WHERE IN SOME CASES DEVELOPING THE APP NATIVELY WOULD BE MORE BENEFICIAL.

3. XAMARIN:

THERE ARE TWO DIFFERENT APPROACHES THAT NEED TO BE EVALUATED. FOR THEIR MOST CROSS-PLATFORM APPROACH, THERE IS XAMARIN. FORMS. ALTHOUGH THE TECHNOLOGY IS VERY DIFFERENT TO REACT NATIVE, CONCEPTUALLY IT OFFERS A SIMILAR APPROACH IN THAT IT ABSTRACTS NATIVE CONTROLS. LIKEWISE, IT HAS SIMILAR DOWNSIDES WITH REGARD TO CUSTOMIZATION. UNLIKE THESE ALTERNATIVES, FLUTTER ATTEMPTS TO GIVE DEVELOPERS A MORE COMPLETE CROSS- PLATFORM SOLUTION, WITH CODE REUSE, HIGH-PERFORMANCE, FLUID USER INTERFACES, AND EXCELLENT TOOLING.

V. CONCLUSION

Flutter offers a great solution for building cross-platform applications. With its excellent tooling and hot reloading, it bringsa very pleasant development experience. The wealth of open-source packages and excellent documentation make it easy to getstarted with. Looking forward, Flutter developers will be able to target Fuchsia in addition to iOS and Android. Considering the extensibility of the engine's architecture, Flutter land on a variety of other platforms as well. With a growing community, it's a great time to jump in.

VI. REFERENCES

1. Chris Bracken. "Release v0.0.6: Rev alpha branch version to 0.0.6,flutter 0.0.26 (#10010) · flutter/flutter" .Godthab. Retrieved 2018-08-08.
2. https://github.com/flutter/flutter/releases
3. https://developers.googleblog.com/2018/09/flutter-release-preview-2-pixel-perfect.html
4. https://github.com/flutter/flutter/wiki/Changelog 5."Google's "Fuchsia" smartphone OS dumps Linux, has a wild new UI" .Ars Technica.