A Seminar Report

on
Native Application Development Using Flutter

A Seminar Report Submitted

in Partial Fulfilment of the Requirements

for the

Diploma in

in

Information Technology

by

Pratyay Prasad Dhond

(Roll No: 1907011)

Department of Information Technology

Government Polytechnic, Nagpur

Near Mangalwari Bazar,Sadar, Nagpur 440 001 (M.S.)

[A Y 2021-22]

DECLARATION

I, **Pratyay Prasad Dhond, (Roll No: 1907011)**, hereby declare that, this report entitled **"Native Application Development Using Flutter"** submitted to Department of Information Technology, Government Polytechnic Nagpur towards partial requirement of **Diploma** in **Information Technology.**

Seminar work carried out by me under the supervision of Dr. A. R. Mahajan. Whenever an external information or statement or result is used then, that have been duly acknowledged and cited.

Place:-Government Polytechnic, Nagpur **Pratyay Prasad Dhond**

Date:-[December 15, 2021]

CERTIFICATE

This is to certify that the Seminar Report entitled **Native Application Development Using Flutter** Submitted by **Pratyay Prasad Dhond (Roll No: 1907011)**, is a bonafide work carried out under the supervision of **Dr. A. R. Mahajan** and it is submitted towards the partial fulfilment of the requirement Government Polytechnic,Nagpur for the award of the Diploma in Information Technology.

Dr. A. R. Mahajan	**Dr.A R Mahajan**
Guide-Lecturer,	Head of the Department,
Department of Information Technology	Department of Information Technology
.	
.	**Dr. M. B. Daigavane**
Seal/Stamp of the College	Principal

Place:-Government Polytechnic, Nagpur

Date:-[December 15, 2021]

ABSTRACT

Flutter is a cross-platform User Interface development framework that is used to develop cross platform applications using a single code base. Flutter can be used to build beautiful, natively compiled application for various platforms such as Android, iOS, Windows, Mac, Linux, Web, etc. using a single codebase.

The key features of the Flutter Framework are Fast Development using the Hot Reload, Expressive and Flexible User Interface(UI) and the Native Performance.

Fast Development : Flutter's hot reload helps you quickly and easily experiment, build UIs, add features, and fix bugs faster. Experience sub-second reload times without losing state on emulators, simulators, and hardware.

Expressive, beautiful UIs: Flutter comes with easy to build, highly customizable UI widgets which can be used for easy development. Flutter also supports custom widget building using other widgets. Native Performance : Flutter's widgets incorporate all critical platform differences such as scrolling, navigation, icons and fonts to provide full native performance on both iOS and Android.

Contents

List of Figures **6**

List of Tables **7**

1 Introduction **1**

1.1 History 2

2 Architecture **4**

2.1 Dart Framework 4

2.2 Flutter's Engine 6

2.3 Platform 6

3 Widgets **7**

3.1 StateLess Widgets 8

3.2 StateFul Widget 9

4 Comparision to other development platforms **11**

4.1 Apple and Android Development 11

4.2 React Native 12

5 Future Of Flutter **13**

6 Conclusion **14**

7 References **15**

List of Figures

2.1 Architecture of Flutter Framework 5

3.1 Stateless Widget 9

3.2 Stateful Widget 10

List of Tables

Chapter 1

Introduction

If you ask ten different mobile developers how they develop their mobile applications for Android or iOS devices, you'll probably get 10 different answers. This use of different languages for different Operating systems creates the need of employment of development teams for each operating system like Android, iOS, Windows, Mac, Linux, as well as webapps,etc. This severely affects the cost of the project to build. As with every project to be built there is a need for different meetings, development phases, testing phases, designing phases, different codebases and ultimately these softwares are hard to manage as a change in the software will need the change to be reflected in each and every code base individually by the different teams.

To avoid this reason there was a need for cross platform development from the old-fashioned porting of apps. For solving this problem, Google

developed the flutter framework to make cross platform development easier, with the view of -'One codebase, multiple platforms!'.

1.1 History

The principal adaptation of Flutter was known by the codename "Sky" and ran on the Android operating framework. It was revealed at the 2015 Dart engineer summit with the expressed expectation of having the option to deliver reliably at 120 frames per second. During the feature of Google Developer Days in Shanghai in September 2018, Google declared Flutter Release Preview 2, which is the last large delivery before Flutter 1.0. On December fourth of that year, Flutter 1.0 was delivered at the Flutter Live occasion, meaning the principal "stable" variant of the Framework. On December 11, 2019, Flutter 1.12 was delivered at the Flutter Interactive event.

On May 6, 2020, the Dart programming advancement unit (SDK) in variant 2.8 and the Flutter in form 1.17.0 were delivered, where backing was added to the Metal API, further developing execution on iOS gadgets (around half), new Material gadgets, and new organization following.

On March 3, 2021, Google delivered Flutter 2 during an internet based Flutter Engage occasion. This significant update brought official help for online applications with new Canvas Kit renderer and web explicit gadgets, early-access desktop application support for Windows, macOS, and

Linux and further developed Add-to-App APIs. This delivery included sound invalid wellbeing, which caused many breaking changes and issues with numerous outer bundles, yet the Flutter group included directions to alleviate these progressions also.

On September eighth, 2021, the Dart SDK in variant 2.14 and Flutter adaptation 2.5 were delivered by Google. The update carried upgrades to the Android Full-Screen mode and the most recent variant of Google's Material Design called Material You. Dart got two new updates, the most up to date build up conditions have been normalized and preset as the default conditions too Dart for Apple Silicon is presently steady.

Chapter 2

Architecture

The Flutter framework is organized into a series of layers, with each layer building upon the previous layer

2.1 Dart Framework

Flutter applications are written in the Dart language and utilize a considerable lot of the language's more progressed highlights. On Android, and on Windows, macOS and Linux by means of the semi-official Flutter Work area Embedding project, Flutter runs in the Dart virtual machine which includes an in the nick of time execution engine. Because of App Store

Figure 2.1: Architecture of Flutter Framework

limitations on unique code execution, Flutter applications use ahead-of-time (AOT) compilation on iOS.

An outstanding element of the Dart stage is its help for "hot reload" where adjustments to source documents can be infused into a running application. Flutter expands this with help for stateful hot reload, where as a rule changes to source code can be reflected promptly in the running application without requiring a restart or any deficiency of state.

2.2 Flutter's Engine

Flutter's Engine, composed essentially on C++, gives low-level rendering support utilizing Google's Skia Graphics library. Furthermore, it interfaces with platform explicit SDKs, for example, those given by Android and iOS. The Flutter Engine is a compact runtime for facilitating Flutter applications. It executes Flutter's core libraries, including activity and designs, file and network I/O, accessibility support, plugin architecture, and a Dart runtime and compile toolchain. Most engineers will associate with Flutter by means of the Flutter Framework, which gives a cutting edge, responsive system, and a rich arrangement of stage, design and establishment gadgets.

2.3 Platform

At the platform level, Flutter gives a Shell, that has the Dart Virtual Machine. The Shell, is platform explicit, giving admittance to the local APIs and facilitating the building up the stage applicable material. There is additionally an embedder API, to utilize Flutter like a library, rather than facilitating running an application. The Shells, additionally assist with giving correspondence to the applicable IMEs and the frameworks application lifecycle occasions.

Chapter 3

Widgets

Everything in Flutter is a Widget. This incorporates UI components, like ListView, TextBox, and Image, just as different parts of the system, including format, activity, motion acknowledgment, and subjects, to give some examples.

Widgets are fundamental for an application's view and interface. They should have a characteristic look and feel paying little heed to screen size. They likewise should be quick, extensible, and adjustable. Flutter takes the Everything is a Widget approach. It has a rich arrangement of Widgets and broad abilities for making complex custom Widgets. In Flutter, Widgets aren't just utilized for views. They're additionally utilized for whole screens and in any event, for the actual application.

As Flutter's documentation puts it, every Widget is an unchanging presentation of part of the UI. Different structures separate perspectives, view regulators, designs, and different properties. Flutter, then again, has a

predictable, consistent, brought together item model: the Widget. A Widget can characterize an underlying component (like a button or menu); a complex component (like a textual style or on the other hand color scheme); a part of the format (like padding, etc. Widgets structure a chain of command dependent on their piece. Every Widget settles within and acquires properties from its parent. There's no different application object. All things considered, the root Widget serves this job.
Flutter has a full arrangement of Widgets in Google's Material Design and in Apple's style with the Cupertino pack. Widget delivering happens straightforwardly in the Skia engine without utilizing Original Equipment Manufacturer Widgets. So we get a smoother UI experience contrasted and other cross platform structures.

Widgets in Flutter are divided into two sub categories:
1) StateLess Widgets
2) StateFul Widgets

3.1 StateLess Widgets

A stateless Widget is a Widget that portrays part of the UI by building a network of different Widgets that portray the UI all the more solidly. The building process proceeds recursively until the description of the UI is completely concrete.

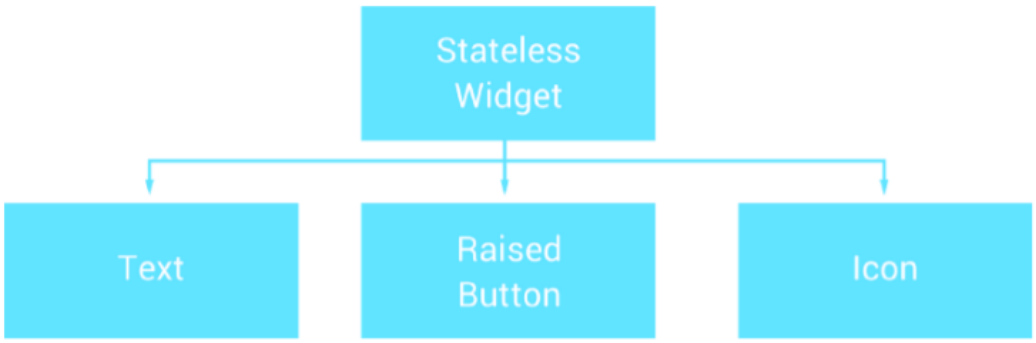

Figure 3.1.1 Stateless Widgets

Figure 3.1: Stateless Widget

Stateless Widget are valuable when the piece of the UI you are portraying doesn't rely upon something besides the arrangement data in the actual article and the BuildContext in which the Widget is expanded. For compositions that can change dynamically, for example due to having an inside clock-driven state, or relying upon some system state, think about utilizing StatefulWidget.

3.2 StateFul Widget

A stateful 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 (e.g., consists entirely of RenderObjectWidgets, which describe concrete RenderObjects).

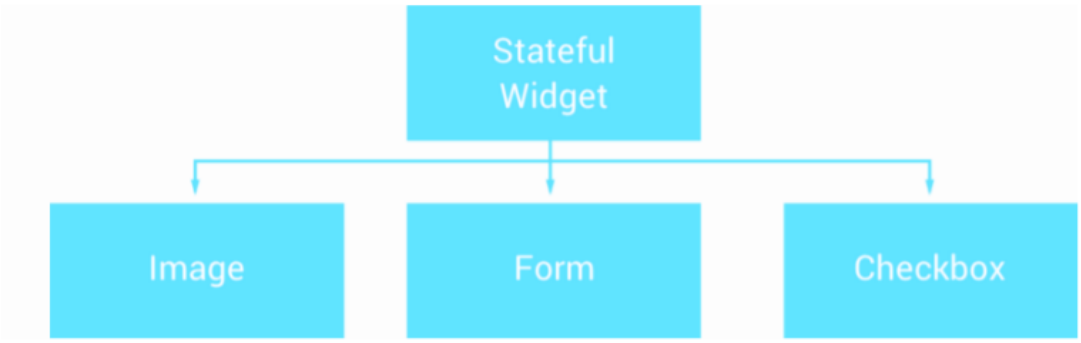

Figure 3.2: Stateful Widget

Stateful widgets are useful when the part of the user interface you are describing can change dynamically, e.g. due to having an internal clock-driven state, or depending on some system state. For compositions that depend only on the configuration information in the object itself and the BuildContext in which the widget is inflated, consider using StatelessWidget.

Chapter 4

Comparision to other development platforms

4.1 Apple and Android Development

Native applications offer the least trouble in adopting new features. These Native Applications tend to have user experiences more in tune with the given platform since the applications are built specifically for the platforms, using the controls from the platform vendors themselves (Apple or Google) and often they follow design guidelines set out by these vendors.

One big advantage of the native applications is that they can adopt brand new technologies, make changes, and access the APIs for the platform directly with the help of lots of documentation available for the Native

application Development framework. Apple and Google create features available 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, as the code is specific to the platform, that is it cannot be used across platforms, which can make development expensive if targeting iOS and Android.

4.2 React Native

React Native allows the software developers to build native applications 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

Chapter 5

Future Of Flutter

Flutter is currently still in beta for windows, Linux and MacOS applications, and is out for pre-release for the web-development. Having Google supporting it is plays a huge role in the success of flutter in the future along with the benefits that flutter offers.

Flutter has a lot of potential and will for sure sustain and make its place in the development environment, as a lot of startups and companies are already switching to Flutter. It can be said that it is soon expected to be capable of overtaking the app development market.

Chapter 6

Conclusion

So overall flutter is a very efficient and cost effective framework which has a high scope for development of cross platform as well as native applications in the present as well as upcoming future.
Even as of now, Flutter offers a great solution for building cross-platform applications. With its excellent tooling and hot reloading, it brings a very pleasant development experience. With a growing community, it's a great time to jump in.

Chapter 7

References

1. https://flutter.dev/
   "This is the official website of Flutter."
2. https://pub.dev/
   "This is a website where developers can access various open source dependencies and implement them in their project."
3. https://dart.dev/
   "This is the official website for Dart programming language."
4. Practical Flutter - Frank Zammetti
   "This is a great book for anyone who wants to learn to code using flutter framework and build cross platform apps."
5. https://www.youtube.com/playlist?list=PLOU2XLYxmsIJ7dsVN4iRuA7BT8XHzGtCr
   "This is a link to a flutter playlist on youtube made by people working on flutter at google themselves."

6. https://www.reddit.com/r/FlutterDev/

   ”Flutter forum on reddit contains various doubts, release notes , bugs, bug fixes, etc.”

7. https://discord.com/invite/N7Yshp4

   ”Flutter community on discord is a great place to chat with other flutter developers, post your doubts, solve other people’s doubts and to grow better as a developer.”