UNIT -1

Introduction

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).

❖ Advantages of Flutter

➢ Fast Development

Flutter's fast development cycle allows developers to see changes to the app in real-time as they make modifications to the code. This can greatly increase the speed and efficiency of the development process of the applications.

➢ High Performance

Flutter offers fast and smooth animations and transitions, and is designed to run smoothly on older devices. The framework is optimized for performance, making it an attractive choice for demanding mobile applications. As a result the number of targeted users increases.

➢ Cross-Platform Development

Flutter supports not only mobile app development but also web and desktop app development. This makes it a versatile tool for developing applications that need to run on multiple platforms without any issues.

➢ Open-Source

Flutter is a free and open-source framework, making it accessible to a wide range of developers and companies. The large community of developers and users working with the framework helps to ensure that it continues to evolve and expand its capabilities**.**

❖ Disadvantages of Flutter

➢ Limited Third-Party Libraries

While Flutter has a growing number of packages and plugins available, the framework is still relatively new, and the number of third-party libraries available for it is limited compared to more established frameworks such as React Native**.**

➢ Steep Learning Curve

The Dart programming language used by Flutter can be challenging for some developers, and there may be a steep learning curve for those who are not already familiar with it.

➢ Limited Corporate Adoption

While Flutter has gained significant traction in the development community, it is still relatively new, and its adoption by large corporations is limited compared to more established frameworks.

❖ Flutter First Application

Step 1: Open the Android Studio.

Step 2: Create the Flutter project. To create a project, go to File-> New->New Flutter Project. The following screen helps to understand it more clearly

Step 3: In the next wizard, you need to choose the Flutter Application. For this, select Flutter Application-> click Next, as shown in the below screen

Step 4: Next, configure the application details as shown in the below screen and click on the Next button

Project Name: Write your Application Name.

Flutter SDK Path: <path_to_flutter_sdk>

Project Location: <path_to_project_folder>

Descriptions: <A new Flutter hello world application>

Step 5: In the next wizard, you need to set the company domain name and click the Finish button..

Step 6: Now, let us check the structure of the Flutter project application and its purpose. In the below image, you can see the various folders and components of the Flutter application structure, which are going to discuss here.

❖ Flutter File Structure

- **.idea:** This folder is at the very top of the project structure, which holds the configuration for Android Studio. It doesn't matter because we are not going to work with Android Studio so that the content of this folder can be ignored.

- **.android:** This folder holds a complete Android project and used when you build the Flutter application for Android. When the Flutter code is compiled into the native code, it will get injected into this Android project, so that the result is a native Android application. **For Example:** When you are using the Android emulator, this Android project is used to build the Android app, which further deployed to the Android Virtual Device.

- **.ios:** This folder holds a complete Mac project and used when you build the Flutter application for iOS. It is similar to the android folder that is used when developing an app for Android. When the Flutter code is compiled into the native code, it will get injected into this iOS project, so that the result is a native iOS application. Building a Flutter application for iOS is only possible when you are working on macOS.

- **.lib:** It is an essential folder, which stands for the library. It is a folder where we will do our 99 percent of project work. Inside the lib folder, we will find the Dart files which contain the code of our Flutter application. By default, this folder contains the file **main.dart**, which is the entry file of the Flutter application.

- **.test:** This folder contains a Dart code, which is written for the Flutter application to perform the automated test when building the app. It won't be too important for us here.

- **gitignore:** It is a text file containing a list of files, file extensions, and folders that tells Git which files should be ignored in a project. Git is a version-control file for tracking changes in source code during software development Git.

- **.metadata:** It is an auto-generated file by the flutter tools, which is used to track the properties of the Flutter project. This file performs the internal tasks, so you do not need to edit the content manually at any time.

- **.packages:** It is an auto-generated file by the Flutter SDK, which is used to contain a list of dependencies for your Flutter project.

- **flutter_demoapp.iml:** It is always named according to the Flutter project's name that contains additional settings of the project. This file performs the internal tasks, which is managed by the Flutter SDK, so you do not need to edit the content manually at any time.

- **pubspec.yaml:** It is the project's configuration file that will use a lot during working with the Flutter project. It allows you how your application works. This file contains:

- **pubspec.lock:** It is an auto-generated file based on the **.yaml** file. It holds more detail setup about all dependencies.

- **README.md:** It is an auto-generated file that holds information about the project. We can edit this file if we want to share information with the developers.

Flutter Architecture

- In this section, we are going to discuss the architecture of the Flutter framework. The Flutter architecture mainly comprises of four components.
    - Flutter Engine
    - Foundation Library
    - Widgets
    - Design Specific Widgets

Flutter Engine

It is a portable runtime for high-quality mobile apps and primarily based on the C++ language. It implements Flutter core libraries that include animation and graphics, file and network I/O, plugin architecture, accessibility support, and a dart runtime for developing, compiling, and running Flutter applications. It takes Google's open-source graphics library, Skia, to render low-level graphics.

Foundation Library

It contains all the required packages for the basic building blocks of writing a Flutter application. These libraries are written in Dart language.

Widgets

In Flutter, everything is a widget, which is the core concept of this framework. Widget in the Flutter is basically a user interface component that affects and controls the view and interface of the app. It represents an immutable description of part of the user interface and includes graphics, text, shapes, and animations that are created using widgets. The widgets are similar to the React components.

In Flutter, the application is itself a widget that contains many sub widgets. It means the app is the top-level widget, and its UI is build using one or more children widgets, which again includes sub child widgets. This feature helps you to create a complex user interface very easily.

We can understand it from the hello world example created in the previous section. Here, we are going to explain the example with the following diagram.

- In Flutter, there are mainly two types of widget:

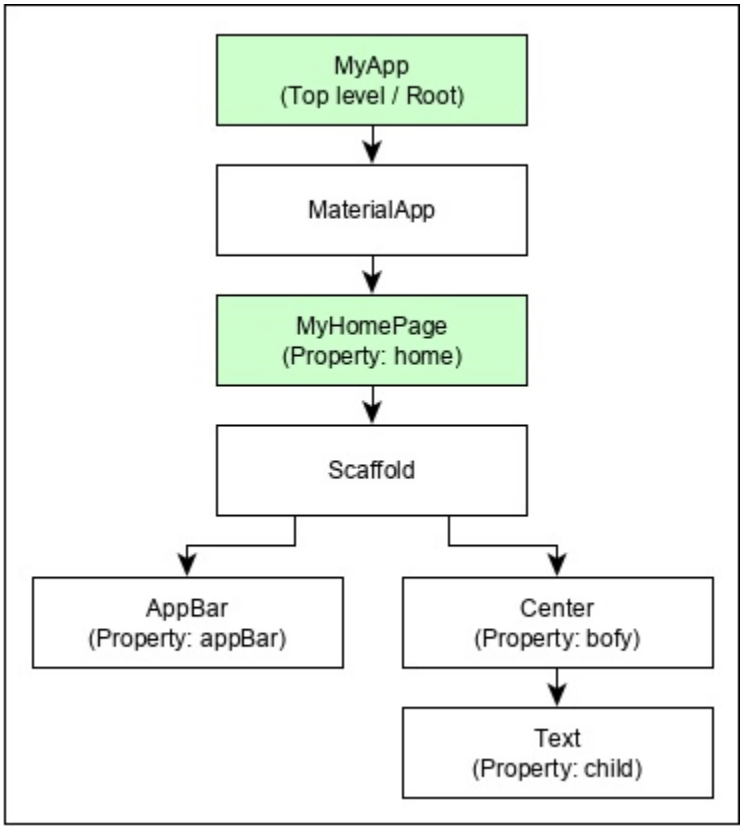
- **StatelessWidget**
- **StatefulWidget**

StatefulWidget

A StatefulWidget has state information. It contains mainly two classes: the state object and the widget. It is dynamic because it can change the inner data during the widget lifetime. This widget does not have a build() method. It has createState() method, which returns a class that extends the Flutters State Class. The examples of the StatefulWidget are Checkbox, Radio, Slider, InkWell, Form, and TextField.

StatelessWidget

The StatelessWidget does not have any state information. It remains static throughout its lifecycle. The examples of the StatelessWidget are Text, Row, Column, Container, etc.



➢ Design Specific Widgets

The Flutter framework has two sets of widgets that conform to specific design languages. These are Material Design for Android application and Cupertino Style for IOS application.

➢ Flutter Vs React native

Concept	Flutter	React Native
Develop By	It is first introduced by Google.	It is first introduced by Facebook.
Release	May 2017	June 2015
Programming Language	It uses Dart language to create a mobile app.	It uses JavaScript to create mobile apps.
Architecture	Flutter uses Business Logic Component (BLoC) architecture.	React Native uses Flux and Redux architecture. Flux created by Facebook, whereas Redux is the preferred choice among the community.
User Interface	It uses custom widgets to build the UI of the app.	It uses native UI controllers to create UI of the app.
Documentation	Flutter documentation is good, organize, and more informative. We can get everything that we want to be written in one place.	React native documentation is user-friendly but disorganized.

Performance	The performance of the Flutter application is fast. Flutter compiles the application by using the arm C/C++ library that makes it closer to machine code and gives the app a better native performance.	The performance of the React Native app is slow in comparison to the Flutter app. Here, sometimes developers face issues while running the hybrid application architecture.
Testing	Flutter provides a very rich set of testing features. This feature allows the developer to perform unit testing, integration testing, and widget testing.	React Native uses third-party tools that are available for testing the app.
Community Support	It has less community support as compared to React Native.	It has very strong community support where the questions and issues can be solved quickly.
Hot Reload	Supported	Supported
Industry Adoption	Google Ads Hamilton Reflectly Xianyu	Facebook Instagram LinkedIn Skype