FLUTTER

Author: Deepti S
Co-Author: DR. Samitha Khaiyum

AUTHOR M.C.A. Post Graduate Students D.S.C.E
CO-AUTHOR M.C.A Associate Professor & HOD D.S.C.E

---***---

Abstract:

In general, developing a mobile application may be a complex and challenging task. There are many frameworks available to develop a mobile application. Android provides a native framework supported Java language and iOS provides a native framework supported Objective-C / Swift language. This paper walks through the fundamentals of Flutter framework, installation of Flutter SDK, fixing Android Studio to develop Flutter based application, architecture of Flutter framework and developing all form of mobile applications using Flutter framework.

1.Introduction:

Flutter is an open source framework to form top quality, high performance mobile applications across mobile operating systems - Android and iOS. It provides an easy, powerful, efficient and straightforward to know SDK to put in writing mobile application in Google's own language, Dart.

Flutter – an easy and high-performance framework supported Dart language, provides high performance by rendering the UI directly within the operating system's canvas instead of through native framework.

Flutter also offers many able to use widgets (UI) to form a contemporary application. These widgets are optimized for mobile environment and designing the applying using widgets is as simple as designing HTML.

Flutter application is itself a widget. Flutter widgets also supports animations and gestures. the applying logic is predicated 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 mandatory changes rather than re-rendering the entire widget.[1]

NOTE:

The kind of apps built using Flutter:

I. Flutter is optimized for 2D mobile applications that want to run on both Android and iOS.

II. Applications that need to offer branded designs are particularly suitable for Flutter. However, apps that should look like stock platform apps can also be created with Flutter.

III. You can create complete applications with Flutter, including camera, geolocation, network, storage, third-party SDKs and more.

How Flutter came into Existence:

Previously, where the Flutter frame did not exist, there were other frames that were popular. Other frameworks such as Xamarin, PhoneGap, React Native Apache Cordova, Titanium and many others have been used before those which are being introduced currently. Then the flutter appeared and became the best of all frames

It is believed that the problem with other frameworks before launching the flutter framework is the user experience. Well, I'm not saying they were slow or something, but they lacked something like the user experience in a native app. On the other hand, flutter already had everything the other framework lacked at the time or didn't support. Flutter also has better development speed and native UX than its user expects. Most Flutter application development companies now use Flutter application development to build a cross-platform application for their customers.

2. Body of Paper:

Prerequisites:

Before starting Flutter, the readers should already bear in mind about what a Framework is which the readers have a sound knowledge on Object Oriented Programming and basic knowledge on Android framework and Dart programming. If you're a beginner to any of those concepts, we advise you bear tutorials associated with these first, before you begin with Flutter.

Flutter Installation:

a) Get the Flutter SDK

b) Download the installation package to get the latest stable version of the Flutter SDK.

c) Extract the zip file and place the floating content in the desired installation location for the Flutter SDK (for example, C: \ src \ flutter; do not install Flutter in a directory like C: \ Program Files \ which requires high privileges).

d) Run flutter doctor

   a. C:\src\flutter>flutter doctor

e) Android Setup: Install Android Stdio

f) Start Android Studio and go through the Android Studio setup wizard. This installs the latest Android SDK, the Android SDK command line tools, and the Android SDK build tools, which are required by Flutter during development for Android.

g) Setup Android Device

h) Setup Android Emulator

3. Flutter Databases:

Flutter supports two databases:

➢ SQLITE

➢ Firebase database

SQLITE:
The SQLite database [5] is the de facto and standard SQL-based integrated database engine. It is a small, proven database engine. The SQLite package provides many features to work effectively with the SQLite database.

Steps to create a database:

Create a new file, Database.dart.

Import necessary import statement in Database.dart

```
Import 'dart:async';

import 'dart:io';

import 'package:path/path.dart';

import 'package:path_provider/path_provider.dart';

import 'package:sqflite/sqflite.dart'; import

'Product.dart';
```

Next declare a singleton based, static SQLiteDbProvider object

```
class SQLiteDbProvider {

  SQLiteDbProvider. _();

  static final SQLiteDbProvider db =

SQLiteDbProvider. _();

  static Database _database;}
```

Create a method to get database products

```
Future<List<Product>>

getAllProducts() async {

final db = await database;

List<Map> results = await

db.query("Product", columns:Product.columns,
orderBy:

"Id ASC");

List<Product> products = new List ();

results.forEach((result) {

Product product = Product.fromMap(result);

products.add(product); }); return products;
```

Cloud Fire store:
Firebase is a platform for developing BaaS applications. It provides many features to accelerate the development

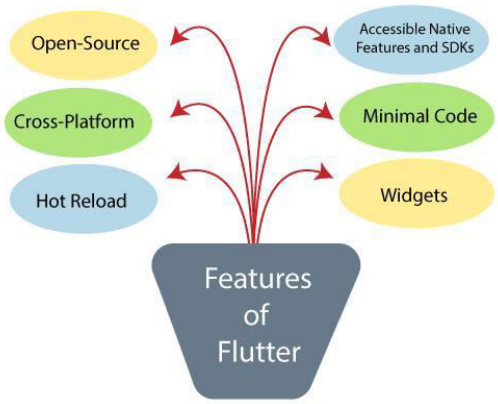
of mobile applications such as authentication service, cloud storage, etc. One of the main features of Firebase is Cloud Fire store, a real-time cloud-based NoSQL database.

Steps to create a database:

- Create a new Flutter app in Android studio, product_firebase_app.
- Replace the default start code (main.dart) with our product_rest_app code.
- Copy the Product.dart file from product_rest_app to the lib folder.

```
class Product {

 final String name;

final String description;

final int price;

final String image;

Product (this.name, this.description, this.price, this.image);

factory Product.fromMap(Map<String, dynamic> json)

{return Product (

json['name'],

 json['description'],

  json['price'],

 json['image'],

 ); }}
```

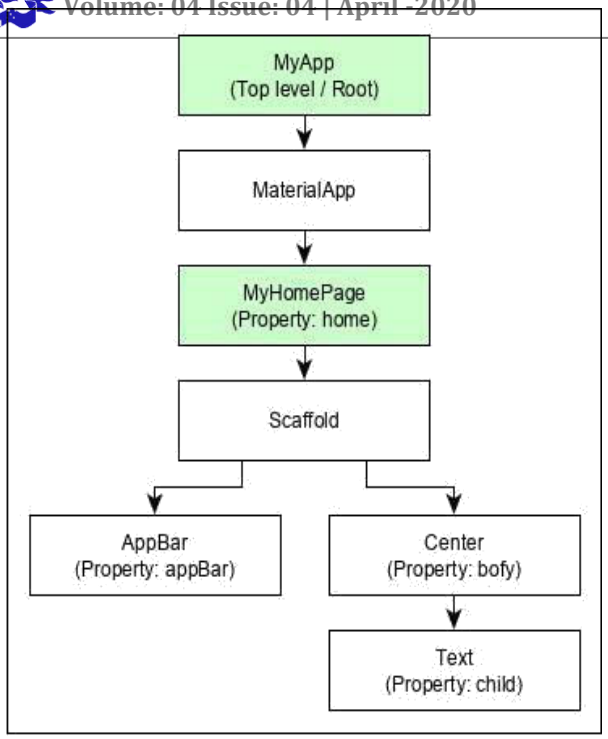
4. Features of Flutter:



- Flutter framework offers the subsequent features to developers
- Open-Source: Flutter may be a free and open-source framework for the event of mobile applications
- Cross-platform: This feature allows Flutter to put in writing code once, maintain it and run on different platforms. It saves developers time, effort and money.
- Hot Reload: Whenever the developer makes changes within the code, then these changes will be seen instantaneously with Hot Reload. It means the changes immediately visible within the app itself. it's a really handy feature, which allows the developer to mend the bugs instantly.
- Accessible native features and SDKs: This feature enables an easy and enjoyable application development process due to native Flutter code, third-party integration and platform APIs. So, we will easily access the SDKs on both platforms.
- Minimal code: The Flutter application is developed by the Dart programing language, which uses JIT and AOT compilation to boost overall startup time, operation and accelerates performance. JIT improves the event system and refreshes the programme without making any extra effort to form a brand new one.
- Widgets: The Flutter framework offers widgets, which may develop customizable specific designs. most significantly, Flutter has two sets of widgets: Material Design and Cupertino widgets that help to supply a glitch-free experience on all platforms.[1]

NOTE:

Flutter provides a tool, flutter doctor to test that each one the need of flutter development is met. flutter doctor



NOTE:

What makes Flutter Unique?

I. Flutter is different from most other mobile app creation options because Flutter does not use WebView or the OEM widgets that come with the device. Instead, Flutter uses its own high-performance rendering engine to draw widgets.

II. In addition, Flutter is different because it only has a thin layer of C / C ++ code. Flutter implements most of its system (compositing, gestures, animation, framework, widgets, etc.) in Dart (a modern, concise and object-oriented language) that developers can easily approach to read, modify, replace or delete. This gives developers enormous control over the system, while drastically lowering the accessibility bar for the majority of the system.[2]

5. Architecture of Flutter:

a) The core concept of the Flutter framework is In Flutter, everything may be a widget. Widgets are basically programme components accustomed create the programme of the applying.

b) In Flutter, the applying is itself a widget. the applying is that 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 form a programme of any complexity.

c) MyApp is the user created widget and its build using the Flutter native widget, MaterialApp.

d) MaterialApp has a home property to specify the programme of the house page, which is again a user created widget, My Homepage.

e) My Homepage is build using another flutter native widget, Scaffold

f) Scaffold has two properties – body and appBar

g) body is accustomed specify its main programme and appBar is accustomed specify its header programme

h) Header UI is build using flutter native widget, AppBar and Body UI is build using Center widget.

i) The Center widget features a property, Child, which refers the particular content and its build using Text widget [2]

Dart Platform:

Flutter applications are written in the Dart language and use many more advanced features of the language. On Windows, macOS and Linux via the semi-official Flutter Desktop Embedding project, Flutter runs in the Dart virtual machine which has an execution engine just in time. When writing and debugging an application, Flutter uses Just in Time compilation, enabling "hot reload", with which changes 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 the source code can be immediately reflected in the current application without requiring a restart or loss of state. This functionality, as implemented in Flutter, has received much praise

The release versions of the Flutter applications are compiled with advance compilation (AOT) on Android and iOS, making the high performance of Flutter possible on mobile devices.[5]

Flutter Engine:

The Flutter engine, written primarily in C ++, provides low-level rendering support using Google's Skia graphics library. In addition, it interfaces with platform-specific SDKs such as those provided by Android and iOS. [9] The Flutter engine is a portable runtime engine for hosting Flutter applications. It implements basic Flutter libraries, including animation and graphics, file and network I / O, accessibility support, plugin architecture and Dart runtime

and compilation tools in chain. Most developers will interact with Flutter via the Flutter Framework, which provides a modern and responsive framework, and a rich set of platform, layout, and foundation widgets.[5]

6. Web Support for Flutter:

In addition to mobile applications, Flutter supports the generation of rendered web content using standard web technologies: HTML, CSS and JavaScript. With web support, you can compile existing Flutter code written in Dart into a client experience that can be integrated into the browser and deployed to any web server. You can use all of Flutter's features and you don't need a browser plug-in.

Adding web support to Flutter involved implementing the main drawing layer of Flutter on top of standard browser APIs, in addition to compiling Dart in JavaScript, instead of the ARM machine code used for mobile applications. Using a combination of DOM, Canvas and CSS, Flutter can deliver a high-quality, high-performance portable user experience on modern browsers. We fully implemented the main drawing layer in Dart and used Dart's optimized JavaScript compiler to compile the kernel and the Flutter framework with your application in a single reduced source file that can be deployed to any web server.[3]

Create a new project using web support:

1. Setup

   $ flutter channel beta

   $ flutter upgrade

   $ flutter config –enable-web

2. Create and run

   $ flutter create

   demo $ cd demo

   $ flutter run -d chrome

3. Build

   $ flutter build web

4. Add web support to an existing app

   $ flutter create.

   $ flutter run -d chrome

7. Widgets in Flutter:

Widgets are basically programme components accustomed create the programme of the applying. Widgets are everything in Flutter Framework. [4]

Stateful and Stateless Widgets:

I. A widget is stateful or stateless. If a widget can change - when a user interacts with it, for example - it is stateful.

II. A stateless widget never changes. Icon, IconButton and Text are examples of stateless widgets. StatelessWidget subclass of stateless widgets.

III. A stateful widget is dynamic: for example, it can change its appearance in response to events triggered by user interactions or when it receives data. Checkbox, Radio, Slider, InkWell, Form and TextField are examples of stateful widgets. Widgets of the StatefulWidget subclass.

IV. The state of a widget is stored in a State object, separating the state of the widget from its appearance. The state consists of values that can change, such as the current value of a cursor or if a check box is checked. When the widget's state changes, the state object calls setState (), telling the framework to redraw the widget.

Types of Widgets:

- Platform specific widgets
- Layout widgets
- State maintenance widgets
- Platform independent / basic widgets

Layout Widgets:

In Flutter, a widget will be created by composing one or more widgets. To compose multiple widgets into one widget, Flutter provides sizable number of widgets with layout feature.

Widgets and Layout Widgets

The heart of Flutter's layout mechanism is made up of widgets. In Flutter, almost everything is a widget - even the layout templates are widgets. The images, icons a

text you see in a Flutter app are all widgets. But the things you don't see are also widgets, such as rows, columns, and grids that organize, constrain, and align visible widgets.To create a Layout design using Widgets we need to follow some of the following steps:

Selecting a Layout Widgets:
Choose the Layout Widgets based on how we need to align or constrain the visible widget.

Creating a visible Layout Widgets:

1. To create a Text Widgets:
   Ex: Text ('Good Morning'),

2. To create an Image Widgets: Ex: Image.asset(
   'images/xyz.jpg',
   fit: BoxFit.cover,
   ),

3. To create an Icon Widgets: Ex: Icon (
   Icons.star,
   color: Colors.red[500],
   ),

Next Add Visible Widgets to the Layout Widgets:

- All the layout widgets have one of the following:
- A child's property if they take only one child

  For Ex: for example, Center or Container
- A property for children if they take a list of widgets

  For Ex: Row, Column, List View, or Stack.

Adding the Text widget to the Center widget:

```
Ex: Center (
  child: Text('abc'),
),
```

Add the layout widget to the page:
A Flutter application is itself a widget, and most widgets have a build () method. Instantiating and returning a widget in the build () method of the application displays the widget.

NOTE: While we are creating a flutter app, we come across with two types of apps

- Material apps
- Non-Material apps\

Material apps
For a Material application, you can use a Scaffold widget; it provides a default banner, background color and has an API to add drawers, snack bars and back sheets. Then you can add the Center widget directly to the body property of the home page.

Ex:

```
class MyApp extends StatelessWidget {
 @override
 Widget build (BuildContext context) {
  return MaterialApp(
   title: 'Flutter layout demo',
   home: Scaffold (
    appBar: AppBar(
     title: Text ('Flutter layout demo'),
    ),
   body: Center (
    child: Text('abc'),
   ), ),);}}
```

Note:
A scaffolding widget provides a framework that implements the basic hardware design visual layout structure of the Flutter application. It provides APIs to display drawers, snack bars and bottom sheets.

Non-Material apps

For a non-material application, you can add the Center widget to the application's build () method: [7]

Ex:

```
class MyApp extends StatelessWidget {
 @override
 Widget build (BuildContext context) {
  return Container (
   decoration: BoxDecoration(color: Colors.white),
```

```
child: Center (
 child: Text (
  'Good Morning',
  textDirection: TextDirection.ltr,
  style: TextStyle (
   fontSize: 32,
   color: Colors.red,),),),), } }
```

Layout Widgets are of two types:

- Single-child layout widgets
- Multi-child layout widgets

Single-child layout widgets

ALIGN layout widget:

This widget will be as large as possible if its dimensions are constrained and width Factor and height Factor are null.[8]

```
Center (
 child: Container (
  height: 150.0,
  width: 150.0,
  color: Colors. Red [50],
  child: Align (
   alignment: Alignment (0.7, 0.6),
   child: Flutter Logo (
size: 60,
   ),),),)
```

Multi-child layout widgets:

Column Class
A widget that displays its children in a vertical table. To have a child expand to fill the available vertical space, wrap him in an expanded widget. [8]

```
Column (
 children: <Widget> [
  Text ('Flutter response'),
  Text('Welcome'),
  Expanded (
   child: FittedBox(
    fit: BoxFit.contain,
    child: const FlutterLogo(),
  ),),],)
```

Flutter Gesture:

Gestures are mainly a way for a user to interact with a mobile application (or any touch device). Gestures are generally defined as any physical action / movement of a user in order to activate a specific control of the mobile device. Gestures are as simple as touching the screen of the mobile device for more complex actions used in game applications.[6]

```
body: Centre (
        Child: GestureDetector(
        onTap: () {
        _showDialog(context);
        },
        Child:Text('Welcome to flutter gesture', )
        ) ),
```

8. Animations in Flutter:

I. Flutter's animation system is based on typed animation objects. Widgets can either incorporate these animations into their build functions directly by reading their current value and listening to their state changes, or they can use the animations as a basis for more elaborate animations which they pass on to other widgets.

II. Well-designed animations make the user interface more intuitive, contribute to the appearance of a refined application, and improve the user experience. The support for Flutter animations facilitates the implementation of different types of animation. Many widgets, especially Material widgets, come with the standard motion effects defined in their design specifications, but it is also possible to customize these effects.

III. After introducing some of the essential concepts, classes and methods into the animation library, it guides you through 5 animation examples. The examples build on each other, showing you different aspects of the animation library.

IV. The Flutter SDK also provides transition animations, such as FadeTransition, SizeTransition and SlideTransition. These simple animations are triggered by defining a starting point and an ending point. They are easier to implement than the explicit animations, which are described here.

V. In Flutter, an Animation object knows nothing about what is on the screen. An animation is an abstract class that understands its current value and its state (finished or rejected). One of the most commonly used animation types is Animation <double>.

VI. An Animation object sequentially generates numbers interpolated between two values over a certain duration. The output of an Animation object can be linear, a curve, a step function, or any other mapping you can design. Depending on how the Animation object is controlled, it can work in the opposite direction or even change direction in the middle.[9]

State maintenance layout:

In Flutter, all widgets are either derived from Stateless Widget or Stateful Widget.
Widget derived from Stateless Widget does not have any state information, but it's going to contain widget derived from Stateful Widget. The dynamic nature of the applying is thru interactive behavior of the widgets and therefore the state changes during interaction

Platform Independent widgets:

Flutter provides large number of basic widgets to create simple as well as complex user interface in a platform independent manner.

Some of Platform Independent Widgets include:

Text

Ex: Text ('Hello World!', style: TextStyle(fontWeight:FontWeight.bold))

Image

Image. Asset('assets/smiley.png')

Icon

Icon

(Icons.email)

9. 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 – [10]

- 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 enough 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 make it even more powerful.
- With Flutter, developers have full control over the widgets and its layout.
- Flutter offers great developer tools, with amazing hot reload.

10. Disadvantage of flutter:

Despite its many advantages, flutter has the following drawbacks in it: [10]

- 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.

11. Conclusion:

The major point when we learn Flutter is it is easy to develop and learn. And state management is better in flutter.

For beginners it feels like flutter architecture is a tough task but once implemented the next steps becomes easy.

Another disadvantage in flutter would be its hardware implementation because its bit suppressed.

Flutter offers its own widgets, one of them is a Table widget, which made creating the table on the cocktail detail page very simple. I had problems creating this table with Android and iOS, so it was a good change of pace. The code below is the function to create all the rows of the table.

12. References:

[1] M. Armbrust et al., "Introduction to Flutter," Commun. ACM, vol. 53, no. 4, pp. 50–58, 2010. https://www.javatpoint.com/flutter

[2] S. Ghemawat, H. Gobioff, and S.- T. Leung, "Architecture of flutter," ACM SIGOPS Oper. Syst. Fire up., vol. 37, no. 5, pp. 29–43, 2003. https://www.tutorialspoint.com/flutter/index.htm

[3] J. Senior member and S. Ghemawat, "Features of flutter," Commun. ACM, vol. 51, no. 1, pp. 107–113, 2008 https://flutter.dev/docs/development/ui/interacti

[4] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica,Widgets and Layouts" in Proc. 2nd USENIX Conf. Hot Topics mobile android app, vol. 10. Boston, MA, USA, 2010, p. https://flutter.dev/docs/development/ui/layout

[5] K. Ashton, "Database and Installation," RFiD J., vol. 22, no. 7, pp. 97–114, 2009.https://www.tutorialspoint.com/flutter/flutter_database_concepts.htm

[6] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "mobile android flutter," IEEE Pervasive Comput., vol. 8, no. 4, pp. 14–23, Oct./Dec. 2009

[7] Feld, "Material and Nonmaterial apps" in Proc. IEEE Int. Workshop Factory Commun. Syst., Vienna, Austria, 2004, pp. 33–38.

[8] D. Evans, "FlutterGesture," CISCO White Paper, vol. 1, pp. 1–11, 2011.

[9] Schiraldi, G. R. (2001). *Flutter Animations* [Adobe Digital Editions version]. doi:10.1036/0071393722

[10] Laxdal, L. S. (2009 Retrieved from ProQuest Dissertations & Theses Global database. (MR82087)https://habr.com/en/post/455020/