Practical No. 11

Title: Introduction to Flutter

Aim: Create an application to demonstrate Flutter

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. 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 drawbacks 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 support 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.

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.

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 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.

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.

Exercise - Create a flutter application to demonstrate various layout widgets and state management

Implementation:

Program:

main.dart

import 'package:flutter/material.dart';

import 'quote.dart';

import 'quote_card.dart';

void main() => runApp(const MaterialApp(

  home: QuoteList()

));

class QuoteList extends StatefulWidget {

  const QuoteList({super.key});

  @override

  _QuoteListState createState() => _QuoteListState();

}

class _QuoteListState extends State<QuoteList> {

 List<Quote> quotes = [

   Quote(author: 'bhushan', text: 'ABCD', index: '1'),

   Quote(author: BHUSHAN', text: "ABCD"', index: '2'),

```dart
    Quote(author: 'bh', text: 'a', index: '3')

  ];

  @override

  Widget build(BuildContext context){

   return Scaffold(

    backgroundColor: Colors.grey[500],

    appBar: AppBar(

     title: const Text('Quotes'),

     centerTitle: true,

     backgroundColor: Colors.red[400],

    ),

    body: Column(

     children: quotes.map((quote) => QuoteCard(quote: quote,)).toList(),

    ),

   );

  }

}
```

quote.dart
```dart
class Quote{
 String text;
 String author;
 String index;
 Quote({
  required this.text, required this.author, required this.index
 });
}
```

quote_card.dart

```
import 'package:flutter/material.dart';

import 'quote.dart';

class QuoteCard extends StatelessWidget {

 final Quote quote;

 const QuoteCard({

   super.key,required this.quote

 });

 @override

 Widget build(BuildContext context) {

  return Card(

    child: Padding(

      padding: const EdgeInsets.all(4.0),

      child: Column(

        crossAxisAlignment: CrossAxisAlignment.stretch,

        children: <Widget>[

         Text(

           quote.text,

           style: TextStyle(

             fontSize: 18.0,

              color: Colors.grey[600],

           ),

         ),

         const SizedBox(height: 6.0),

         Text(
```

```
      quote.author,

      style: TextStyle(

        fontSize: 14.0,

        color: Colors.grey[800],

       ),

      ),

      const SizedBox(height: 8.0),

      TextButton.icon(

        style: TextButton.styleFrom(foregroundColor: Colors.red),

        onPressed: (){
        },

        label: const Text('Fav'),

        icon: const Icon(Icons.favorite_border_outlined),

       )
     ],
    ),
   )
  );
 }
}
```

Output:
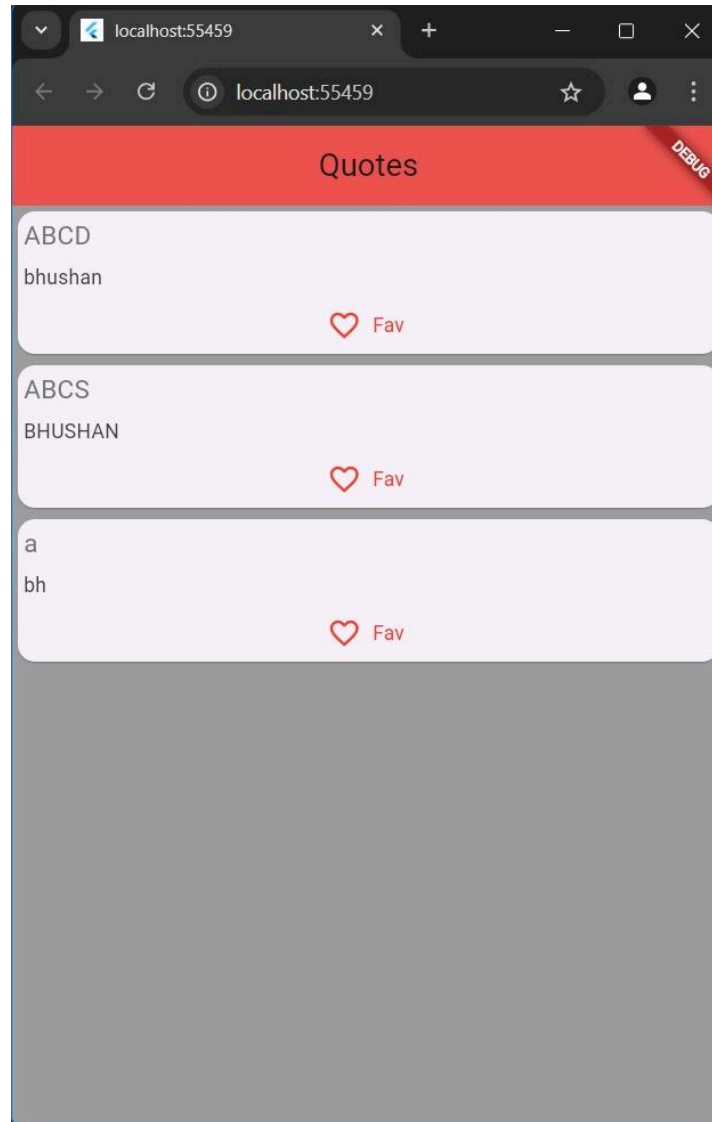
Conclusion -

Understood creating an application to demonstrate Flutter to explore robustness.