Name:Divit Wadhwani Roll no:71 Exp:2

EXPERIMENT 2

AIM: To design flutter UI by including common widgets

Theory:

In Flutter, widgets are the basic building blocks of the user interface. Everything in Flutter is a widget, and widgets are used to create the visual elements and layout of the application. Flutter provides a wide range of pre-built widgets that you can use to construct your UI, and you can also create your own custom widgets.

Widgets in Flutter can be categorized into two main types:

- Stateless Widgets: These widgets are immutable, meaning their properties (attributes) cannot change once they are set. Stateless widgets are used for parts of the UI that don't change dynamically.

- Stateful Widgets: These widgets can change dynamically based on user interactions, data changes, or other factors. Stateful widgets are used for parts of the UI that need to be updated over time.

Flutter widgets are composable, meaning you can nest them and combine them to build complex UIs. The framework provides a rich set of widgets for common UI elements like buttons, text, images, layouts, and more. This composability and the reactive nature of Flutter widgets contribute to the flexibility and expressiveness of Flutter applications.

Flutter, widgets are not just visual components; they are also the basic elements for handling layout, interaction, animation, and more. Here are some key aspects and features of Flutter widgets:

Widget Tree:

- Flutter applications are built using a widget tree. The widget tree is a hierarchical structure of widgets, where each widget can contain other widgets as its children.
- The root of the widget tree is typically the MaterialApp or CupertinoApp widget, depending on the design language you are using (Material Design for Android, Cupertino for iOS).

Composable and Nestable:

- Flutter widgets are composable, meaning you can combine and nest them to create complex UIs. This composability allows for the easy construction of sophisticated user interfaces.

Layout Widgets:

- Flutter provides a variety of layout widgets to structure and position other widgets on the screen. Examples include Container, Row, Column, Stack, Expanded, and more.

Gesture Handling:

- Flutter has widgets for handling gestures like taps, swipes, and drags. For example, GestureDetector is a versatile widget that can recognize various gestures and trigger corresponding callbacks.

Animation:

- Flutter makes it easy to create animations using widgets. The Animation and Tween classes, along with widgets like AnimatedContainer and Hero, allow you to add motion and transitions to your UI.

Material Design and Cupertino Widgets:

- Flutter provides widgets that follow the Material Design guidelines (for Android) and Cupertino design (for iOS). For example, AppBar, BottomNavigationBar, TextField, and others adhere to the respective platform's design principles.

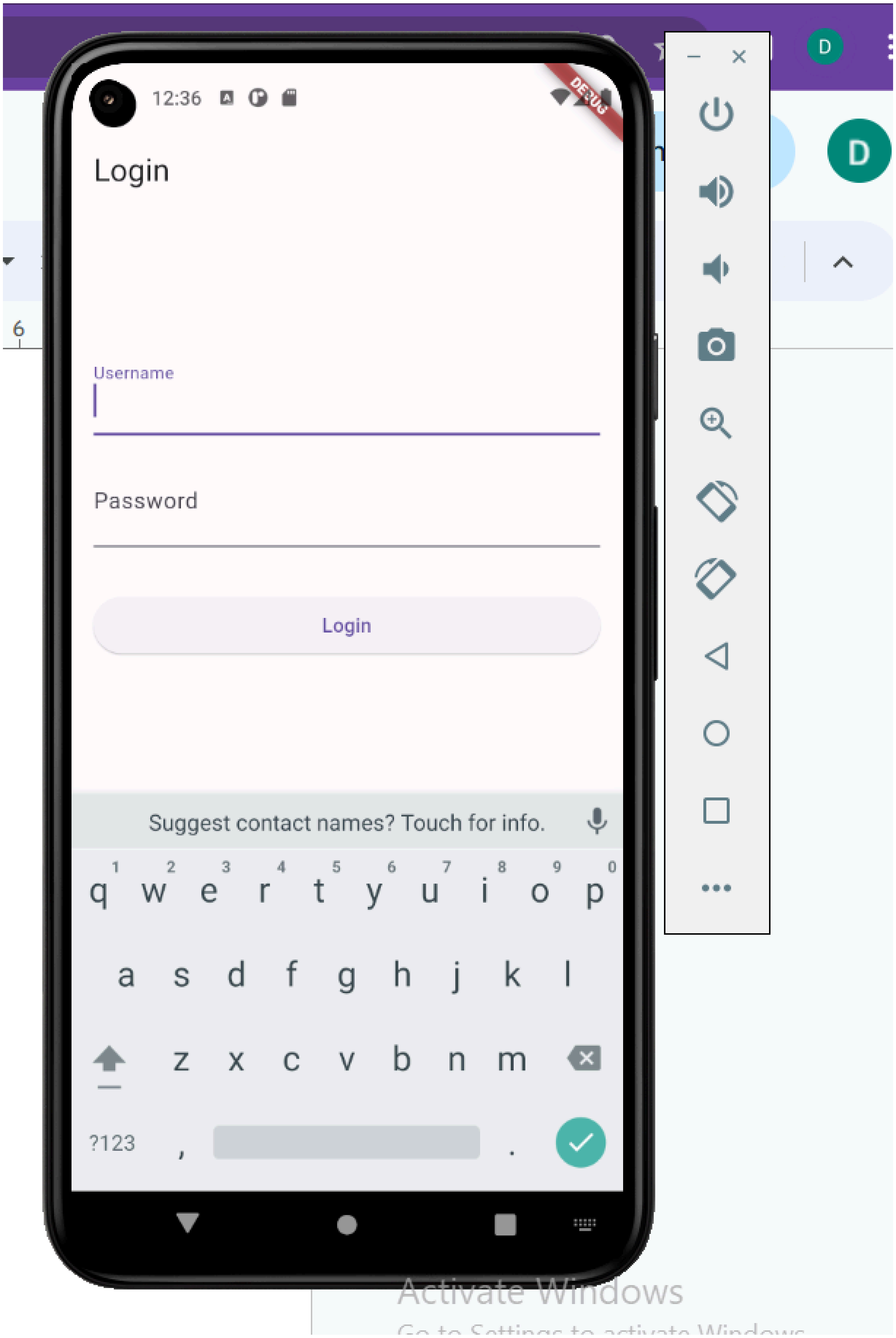
code:
```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter UI Example',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: LoginScreen(),
    );
  }
}

class LoginScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Login'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            TextField(
              decoration: InputDecoration(
                labelText: 'Username',
              ),
            ),
            SizedBox(height: 16.0),
            TextField(
              obscureText: true,
              decoration: InputDecoration(
                labelText: 'Password',
              ),
            ),
            SizedBox(height: 32.0),
            ElevatedButton(
              onPressed: () {
                // Implement login logic here
              },
              child: Text('Login'),
            ),
          ],
        ),
      ),
    );
  }
}
```

Name:Divit Wadhwani Roll no:71 Exp:2


12:36
DEBUG
Login
Username
Password
Login
Suggest contact names? Touch for info.
Activate Windows
Go to Settings to activate Windows

conclusion:

In conclusion, the provided Flutter code demonstrates the creation of a simple login screen using common widgets. The UI includes a title bar, username and password input fields, and a login button. This example can serve as a starting point for more complex applications, and you can further customize and expand the UI based on your specific requirements.