Experiment 2

Name: Ameya Angne **D15B/01**

Aim: To design Flutter UI by including common widgets.

Introduction:

Flutter, developed by Google, is a popular open-source UI toolkit for building natively compiled applications for mobile, web, and desktop from a single codebase. Flutter excels in creating visually appealing and responsive user interfaces. The key to this lies in the extensive use of common widgets that Flutter provides out of the box.

Widgets in Flutter are the building blocks of the user interface, representing everything from buttons and text to complex layouts and animations. These widgets are highly customizable and can be combined to create diverse and engaging user interfaces. In this context, a "widget" refers to both visual elements and structural components of the app.

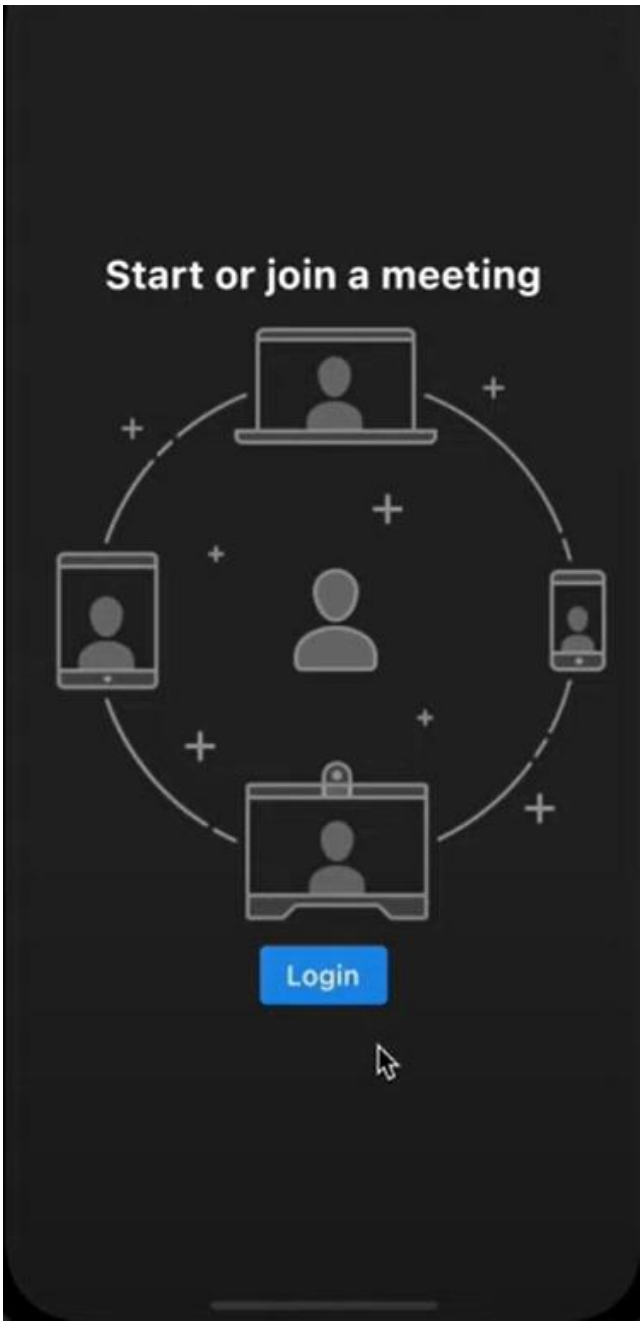
Code:

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

import 'package:zoom_clone_tutorial/resources/auth_methods.dart';

import 'package:zoom_clone_tutorial/widgets/custom_button.dart';

class LoginScreen extends StatefulWidget {
  const LoginScreen({Key? key}) : super(key: key);

  @override
  State<LoginScreen> createState() => _LoginScreenState();
}
```

```
class _LoginScreenState extends State<LoginScreen> {
  final AuthMethods _authMethods = AuthMethods();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          const Text(
            'Start or join a meeting',
            style: const TextStyle(
              fontSize: 24,
              fontWeight: FontWeight.bold,
            ),
          ),
          Padding(
            padding: const EdgeInsets.symmetric(vertical: 38.0),
            child: Image.asset('assets/images/onboarding.jpg'),
          ),
          CustomButton(
            text: 'Google Sign In',
            onPressed: () async {
              bool res = await _authMethods.signInWithGoogle(context);
              if (res) {
                Navigator.pushNamed(context, '/home');
              }
            },
          ),
        ],
      ),
    );
```

```
  }
}
```

Output:



Explanation:

1.Imports:

import 'package:flutter/material.dart';: Imports the Flutter material library, which provides a set of widgets implementing Material Design.

import 'package:zoom_clone_tutorial/resources/auth_methods.dart';: Imports functionality related to authentication methods from an external source, likely another file or package.

import 'package:zoom_clone_tutorial/widgets/custom_button.dart';: Imports a custom button widget from an external source, likely another file.

2.LoginScreen Class:

LoginScreen is a StatefulWidget, meaning it's a widget that maintains state.

It doesn't have any parameters in its constructor.

It overrides the createState() method to return an instance of _LoginScreenState.

3._LoginScreenState Class:

_LoginScreenState is the state class associated with LoginScreen.

It has a private instance of AuthMethods class named _authMethods.

build() Method:

This method builds the UI of the LoginScreen.

It returns a Scaffold widget as the root widget, providing structure for the screen.

Inside the Scaffold, there's a Column widget containing various children widgets vertically aligned.

The first child is a Text widget displaying "Start or join a meeting" with specific styling.

The second child is an Image widget displaying an image, likely related to onboarding or branding.

The third child is a CustomButton widget, labeled "Google Sign In". When pressed, it triggers the _authMethods.signInWithGoogle() method, likely to initiate the Google sign-in process. Upon successful sign-in (res is true), it navigates to the '/home' route using Navigator.pushNamed().

4.Widget Structure:

The screen is structured vertically with a heading, an image, and a button for Google Sign In.

The sign-in process with Google is asynchronous, as indicated by the use of async and await.

Conclusion:

This experiment demonstrates the use of common Flutter widgets to create a basic login screen. Flutter provides a rich set of widgets that can be easily combined to build complex and beautiful user interfaces. Customization and theming can be applied to enhance the visual appeal of the app. This is just a simple example, and you can extend and modify it based on your specific requirements. Remember to handle the actual authentication logic securely in a real-world application.