EXPERIMENT NO.02

Aim: To design Flutter UI by including common widgets.

Theory :

In Flutter, widgets serve as the fundamental building blocks for creating user interfaces. They encompass both structural components like rows and columns, as well as stylistic elements such as colors and fonts. Every aspect of a Flutter app is constructed using widgets, which define the layout, appearance, and behavior of the UI components. Widgets can be combined and nested to compose intricate and interactive layouts.

When coding in Flutter, all elements are encapsulated within widgets. Widgets define how the app's view should appear based on their current configuration and state. Modifications to the code trigger widget rebuilding, where the framework calculates the differences between the previous and current widget states to determine the minimal changes required for rendering the UI.

Widgets are hierarchically nested to construct the app, with the root of the app itself being a widget, and all subsequent components also represented as widgets. For instance, a widget can handle displaying content, defining design attributes, managing user interactions, and more.

Here are some commonly used widgets in Flutter:
Text: The Text widget is fundamental for displaying textual content in your app. It allows you to customize the text style, including font family, size, weight, color, and more. Additionally, you can align the text, set its direction, and handle overflow behavior.

Container: The Container widget is incredibly versatile, serving as a blank canvas that can contain other widgets. It allows you to apply decoration such as color, border, and gradient to customize its appearance. You can also specify padding, margin, alignment, constraints, and transformations for precise layout control.

ElevatedButton: This widget represents a material design button that responds to touch input by elevating and indicating that it can be pressed. You can customize its appearance using properties like text style, background color, elevation, shape, and padding. Additionally, you can define callback functions to execute when the button is pressed, enabling interactivity within your app.

Scaffold: The Scaffold widget provides a basic layout structure for material design apps, encapsulating common UI elements such as app bars, drawers, bottom navigation bars, and floating action buttons. It simplifies the creation of standard app layouts and navigation patterns, allowing developers to focus on building the app's specific features.

Checkbox: The Checkbox widget allows users to make binary choices by toggling between checked and unchecked states. It is commonly used in forms, settings, and lists where users need to select multiple options. You can customize the checkbox's appearance, label, and state management to suit your app's requirements.

Image: With the Image widget, you can display various types of images in your app, including network images fetched from URLs, asset images stored locally in your project, and memory images generated dynamically. You can specify parameters like width, height, fit, alignment, and loading behavior to control how images are displayed.

ListView: The ListView widget is indispensable for building scrollable lists of widgets. It efficiently renders only the items visible on the screen, making it suitable for displaying large datasets without sacrificing performance. You can choose between different list view types, such as ListView.builder for dynamically generated children or ListView.separated for adding separators between items.

AppBar: The AppBar widget represents the top app bar that typically contains the app's title, leading and trailing actions, and optional navigation controls. It provides built-in support for common app bar features like title styling, background color, elevation, and actions. Additionally, you can customize the app bar's behavior, including scrolling behavior and flexible space.

BoxDecoration: This widget allows you to decorate a box with various visual effects like color, gradient, border, shadow, and shape. It is commonly used as the decoration property of widgets like Container, Card, and DecoratedBox to enhance their appearance and add visual interest to your app's UI.

MaterialApp: The MaterialApp widget serves as the root of your Flutter app, providing essential configurations and services for building material design applications. It sets up navigation, theming, internationalization, accessibility, and other critical aspects of the app's runtime environment. Additionally, it manages the app's lifecycle and routes, ensuring a smooth and consistent user experience.

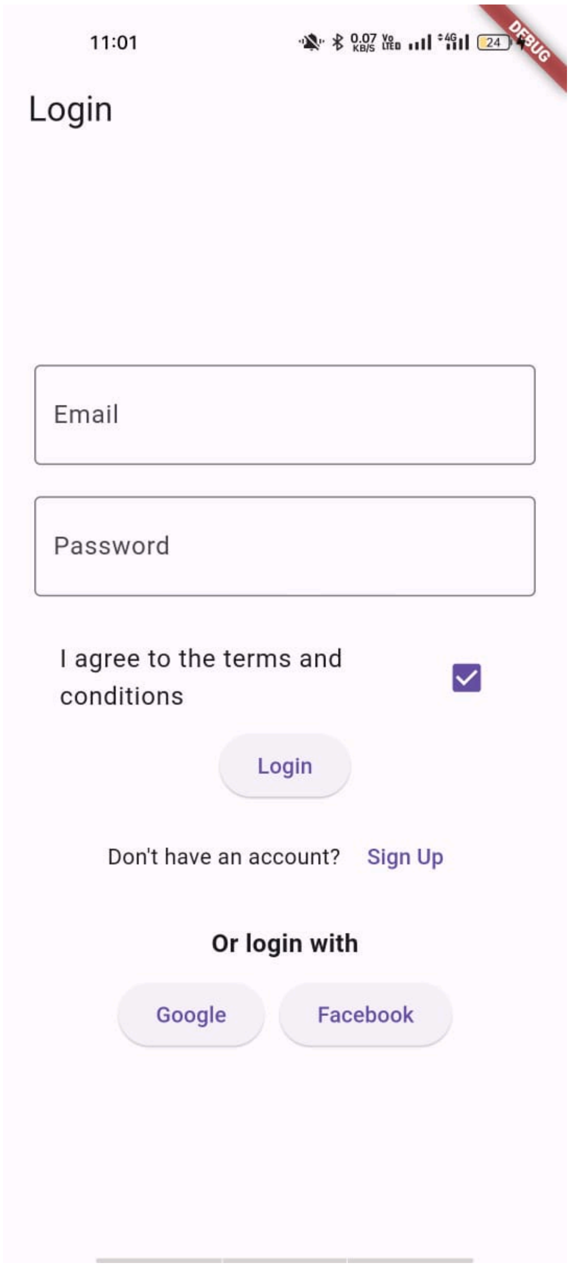
Code :

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

void main() {
  runApp(MaterialApp(
    home: MobileLoginPage(),
  ));
}

class MobileLoginPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Login'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            TextFormField(
              decoration: InputDecoration(
                labelText: 'Email',
                border: OutlineInputBorder(),
              ),
            ),
            SizedBox(height: 20),
            TextFormField(
              decoration: InputDecoration(
                labelText: 'Password',
                border: OutlineInputBorder(),
              ),
              obscureText: true,
            ),
            SizedBox(height: 20),
            CheckboxListTile(
              value: true,
              title: Text('I agree to the terms and conditions'),
              onChanged: (value) {
                print('Checkbox value changed to: $value');
              },
            ),
            ElevatedButton(
              onPressed: () {
                // Add login functionality
              },
              child: Text('Login'),
            ),
            SizedBox(height: 10),
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Text('Don\'t have an account?'),
                SizedBox(width: 5),
                TextButton(
                  onPressed: () {
                    // Add navigation to sign up page
                  },
                  child: Text('Sign Up'),
                ),
              ],
            ),
            SizedBox(height: 20), // Added space
            Text(
              'Or login with',
              style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
            ),
            SizedBox(height: 10), // Added space
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                // You can add more social login buttons here
                ElevatedButton(
                  onPressed: () {
                    // Add Google login functionality
                  },
                  child: Text('Google'),
                ),
                SizedBox(width: 10),
                ElevatedButton(
                  onPressed: () {
                    // Add Facebook login functionality
                  },
                  child: Text('Facebook'),
                ),
              ],
                      ),
                    ],
                  ),
                ),
              );
             }
            }
```

Output :


11:01
DEBUG
Login
Email
Password
I agree to the terms and conditions
Login
Don't have an account? Sign Up
Or login with
Google
Facebook