

Experiment No:02

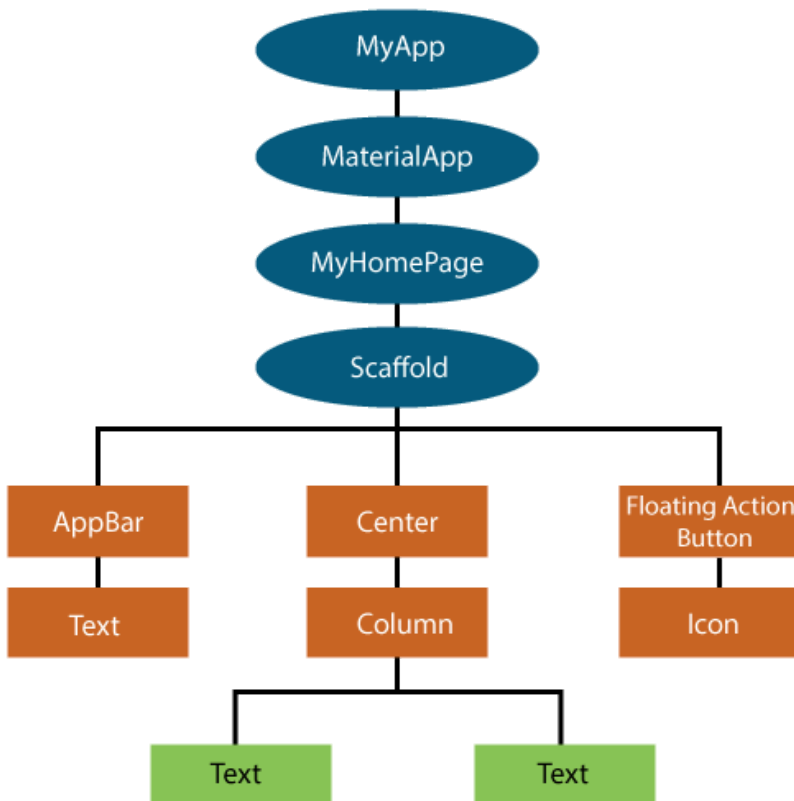
Aim: To design Flutter UI by including common widgets.

Theory:

Whenever you are going to code for building anything in Flutter, it will be inside a widget. The central purpose is to build the app out of widgets. It describes how your app view should look like with their current configuration and state. When you made any alteration in the code, the widget rebuilds its description by calculating the difference of previous and current widget to determine the minimal changes for rendering in UI of the app.

Widgets are nested with each other to build the app. It means the root of your app is itself a widget, and all the way down is a widget also. For example, a widget can display something, can define design, can handle interaction, etc.

The below image is a simple visual representation of the widget tree.



Types of Widget

We can split the Flutter widget into two categories:

1. Visible (Output and Input)
2. Invisible (Layout and Control)

Common Widgets in Flutter

Flutter provides a wide range of widgets that developers can use to build rich and interactive user interfaces. These widgets serve various purposes, from displaying text and images to handling user input and managing layouts.

In the below code we use the following widgets:

- 1.MaterialApp:** This widget represents a Flutter application that uses material design.
- 2.Scaffold:** Scaffold is a layout structure widget from the Material library that provides a default layout structure for the app. It includes an app bar, a body, and other structural elements.
- 3.AppBar:** AppBar is a Material Design app bar that displays the title and other actions above the app's main content.
- 4.SingleChildScrollView:** This widget enables scrolling when the content is too large to fit within the visible area. It allows the child widget to be scrolled in one direction.
- 5.Column:** Column is a layout widget that arranges its children vertically, one after another.
- 6.Image:** The Image widget displays an image. In this code, it's used to display the login page's logo.
- 7.SizedBox:** SizedBox is a widget that creates a fixed-size box. It's used here to create space between widgets vertically.
- 8.TextField:** TextField is a widget that allows users to enter text. It's used here for the email and password input fields.
- 9.ElevatedButton:** ElevatedButton is a button widget with a raised appearance, typically used for primary actions.
- 10.TextButton:** TextButton is a button widget with only text, suitable for secondary actions.
- 11.Text:** Text is a widget that displays a string of text. It's used to provide labels and button text in this code.

These widgets are used to create a simple login page with email and password input fields, along with login and forgot password buttons.

Code and output:

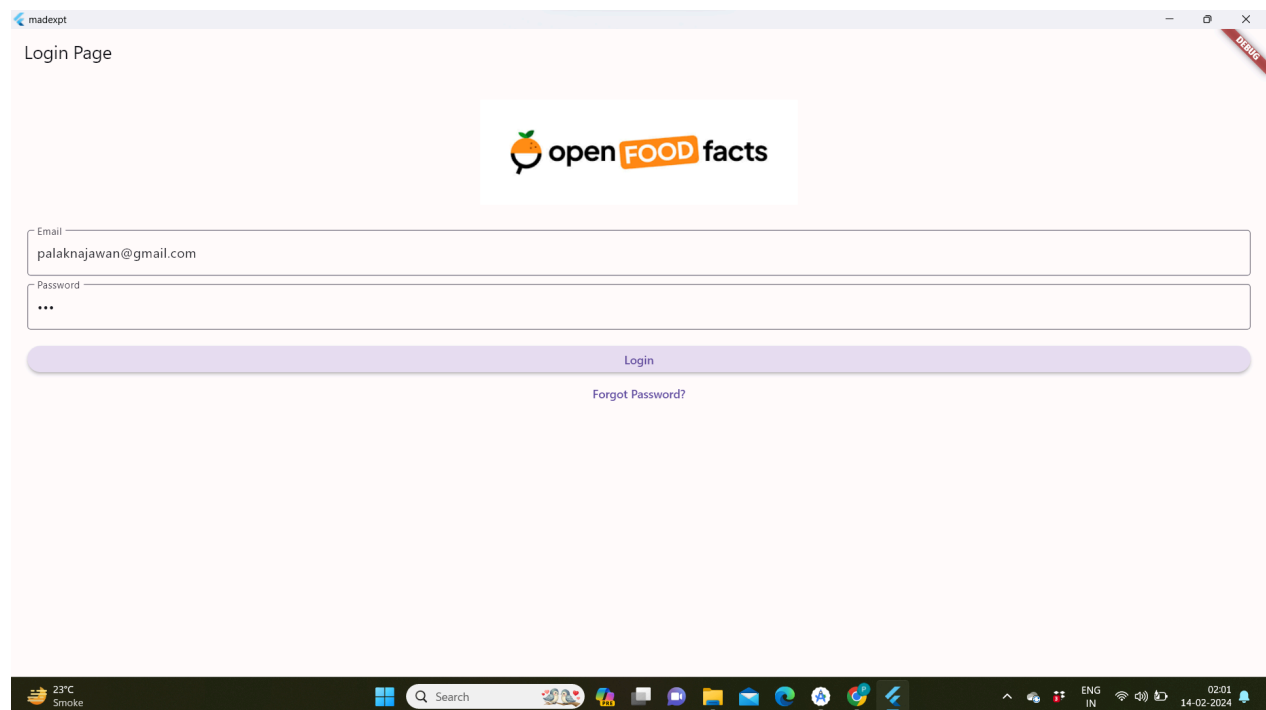
```
import 'package:flutter/material.dart';

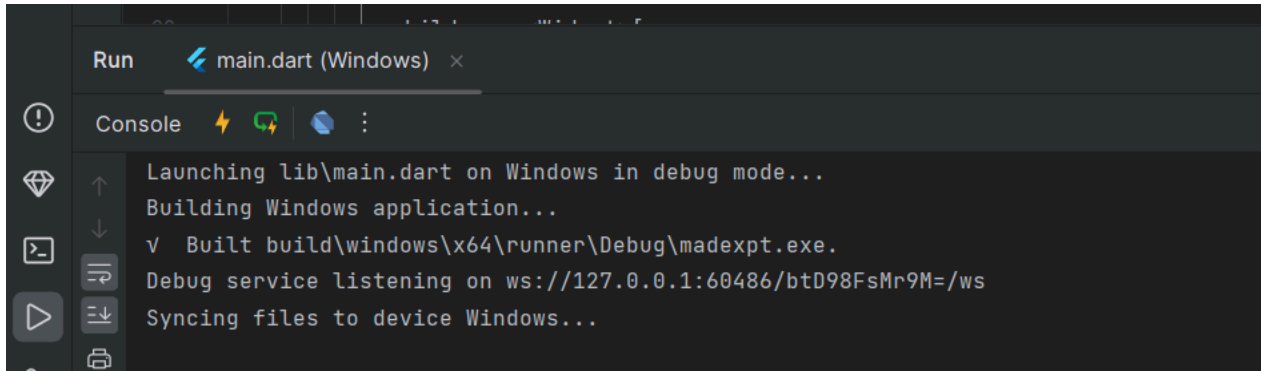
void main() {
  runApp(LoginPage());
}

class LoginPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Login Page'),
        ),
        body: SingleChildScrollView(
          padding: EdgeInsets.all(20.0),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            crossAxisAlignment: CrossAxisAlignment.stretch,
            children: <Widget>[
              Image(
                image: AssetImage('images/download.png'),
                height: 150.0,
                width: 150.0,
              ),
              SizedBox(height: 20.0),
              TextField(
                decoration: InputDecoration(
                  labelText: 'Email',
                  border: OutlineInputBorder(),
                ),
              ),
              SizedBox(height: 10.0),
              TextField(
                obscureText: true,
                decoration: InputDecoration(
                  labelText: 'Password',
                  border: OutlineInputBorder(),
                ),
              ),
              SizedBox(height: 20.0),
              ElevatedButton(
                onPressed: () {
                  // Add login logic here
                },
                child: Text('Login'),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

```
    ),  
    SizedBox(height: 10.0),  
    TextButton(  
      onPressed: () {  
        // Add forgot password logic here  
      },  
      child: Text('Forgot Password?'),  
    ),  
  ],  
),  
),  
),  
),  
),  
);  
}  
}
```

Output:





```
Run main.dart (Windows) x
Console ⚡ ↻ 🔍 ⋮
Launching lib\main.dart on Windows in debug mode...
Building Windows application...
✓ Built build\windows\x64\runner\Debug\madexpt.exe.
Debug service listening on ws://127.0.0.1:60486/btD98FsMr9M=/ws
Syncing files to device Windows...
```

Conclusion:

In conclusion, designing Flutter UIs with common widgets provides a robust foundation for creating beautiful, functional, and responsive applications. By leveraging the versatility and flexibility of these widgets, developers can efficiently build UIs that deliver an exceptional user experience on both iOS and Android platforms.