**Experiment 03 : To include icons, images, fonts in Flutter app**

Aim: To include icons, images, fonts in Flutter app

We can split Flutter widgets into two categories:

* **Visible (Output and Input)**
* **Invisible (Layout and Control)**

**Visible Widgets**

Visible widgets are related to user input and output data. Some of the important types of these widgets are:

Code :

import 'package:fintracker/bloc/cubit/app\_cubit.dart';  
import 'package:fintracker/screens/main.screen.dart';  
import 'package:flutter/material.dart';  
import 'package:flutter/services.dart';  
import 'package:flutter\_bloc/flutter\_bloc.dart';  
import 'package:flutter\_localizations/flutter\_localizations.dart';  
  
class App extends StatelessWidget {  
 const App({super.key});  
 @override  
 Widget build(BuildContext context) {  
 SystemChrome.*setSystemUIOverlayStyle*(SystemUiOverlayStyle(  
 statusBarColor: Colors.*transparent*,  
 statusBarIconBrightness: MediaQuery.*of*(context).platformBrightness  
 ));  
 return BlocBuilder<AppCubit, AppState>(  
 builder: (context, state){  
 return MaterialApp(  
 title: 'Fintracker',  
 theme: ThemeData(  
 useMaterial3: true,  
 brightness: MediaQuery.*of*(context).platformBrightness,  
 navigationBarTheme: NavigationBarThemeData(  
 labelTextStyle: WidgetStateProperty.*resolveWith*((Set<WidgetState> states){  
 TextStyle style = const TextStyle(fontWeight: FontWeight.*w500*, fontSize: 11);  
 if(states.contains(WidgetState.selected)){  
 style = style.merge(const TextStyle(fontWeight: FontWeight.*w600*));  
 }  
 return style;  
 }),  
 )  
 ),  
 home: const MainScreen(),  
 localizationsDelegates: const [  
 GlobalWidgetsLocalizations.*delegate*,  
 GlobalMaterialLocalizations.*delegate*,  
 ],  
 );  
 }  
 );  
 }  
}

**1. Text**

A Text widget holds some text to display on the screen. We can align the text widget by using the textAlign property, and the style property allows customization of the text, including font, weight, style, letter spacing, and color.

**Example:**

new Text(

'Hello, ALL!',

textAlign: TextAlign.center,

style: new TextStyle(fontWeight: FontWeight.bold),

)

**2. Button**

This widget allows users to perform some actions on a click. Flutter does not allow direct use of a Button widget; instead, it provides different types like FlatButton and RaisedButton.

**Example:**

// FlatButton Example

new FlatButton(

child: Text("Click here"),

onPressed: () {

// Do something here

},

),

// RaisedButton Example

new RaisedButton(

child: Text("Click here"),

elevation: 5.0,

onPressed: () {

// Do something here

},

)

**3. Image**

This widget holds an image, which can be fetched from multiple sources such as an asset folder or directly from a URL. It provides many constructors for loading images:

* Image: A generic image loader using ImageProvider.
* asset: Loads an image from the project's asset folder.
* file: Loads an image from the system folder.
* memory: Loads an image from memory.
* network: Loads an image from the internet.

**Adding an Image in Flutter**

To add an image to the project, first create an assets folder where the image is stored, and then add the following line to the pubspec.yaml file:

assets:

- assets/comp.jpg

**Example:**

import 'package:flutter/material.dart';

void main() {

runApp(const MyApp());

}

class MyApp extends StatelessWidget {

const MyApp({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Welcome to Flutter',

home: Scaffold(

appBar: AppBar(

title: const Text('Welcome to Flutter'),

),

body: Center(

child: Image.asset('assets/comp.jpg'),

),

),

);

}

}

**Output:**

The app will display an image from the assets folder.

**Single Child Widget Example:**

class MyButton extends StatelessWidget {

MyButton({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return Container(

decoration: const BoxDecoration(

border: Border(

top: BorderSide(width: 1.0, color: Color(0xFFFFFFFFFF)),

left: BorderSide(width: 1.0, color: Color(0xFFFFFFFFFF)),

right: BorderSide(width: 1.0, color: Color(0xFFFF000000)),

bottom: BorderSide(width: 1.0, color: Color(0xFFFF000000)),

),

),

child: Container(

padding: const EdgeInsets.symmetric(horizontal: 20.0, vertical: 2.0),

decoration: const BoxDecoration(

border: Border(

top: BorderSide(width: 1.0, color: Color(0xFFFFDFDFDF)),

left: BorderSide(width: 1.0, color: Color(0xFFFFDFDFDF)),

right: BorderSide(width: 1.0, color: Color(0xFFFF7F7F7F)),

bottom: BorderSide(width: 1.0, color: Color(0xFFFF7F7F7F)),

),

color: Colors.grey,

),

child: const Text(

'OK',

textAlign: TextAlign.center,

style: TextStyle(color: Colors.black),

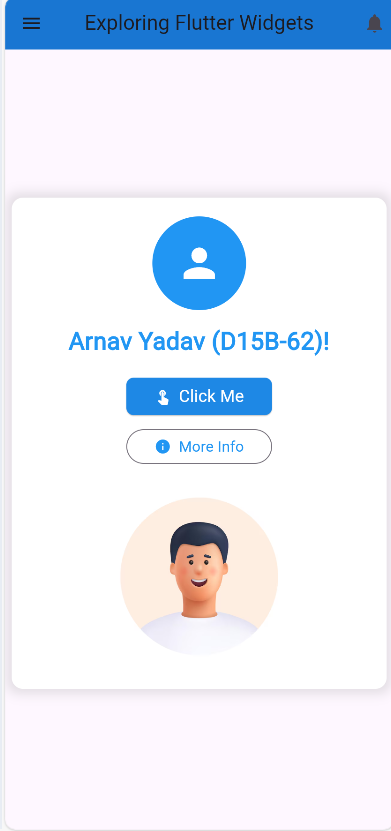
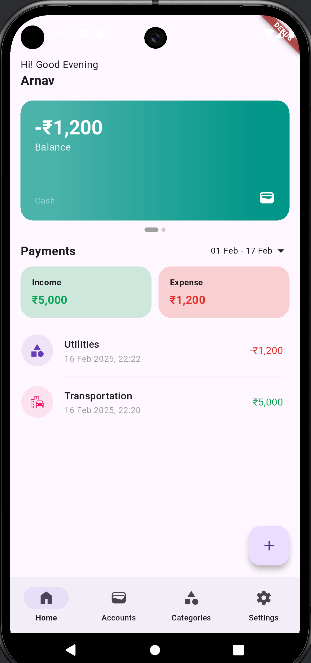
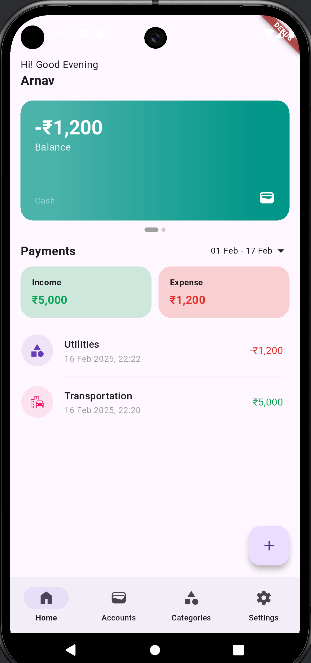
),

),

);

}

}



***Conclusion:***

In this experiment, we explored Flutter widgets, specifically focusing on Text, Button, and Image widgets. These widgets play a crucial role in building interactive UI elements in Flutter applications. We investigated their properties, customization options, and how they can be combined to create more complex layouts. Furthermore, we examined how to handle user interactions with buttons and display images from various sources.