Experiment 2

Name: Sonali Makhijani
Class: D15A
Roll no:34

Aim: To design Flutter UI by including common widgets.

Theory:

Flutter widgets are the building blocks of the user interface, representing visual and interactive elements. Widgets can be categorized into two main types: Stateless andStateful. Stateless widgets are immutable and do not store any internal state, while Stateful widgets can change and maintain their state over time.

Built-in and Custom Widgets:
Flutter comes with a rich set of built-in widgets that cover a wide range of UI elements and interactions. Additionally, developers can create their custom widgetsto tailor the user interface to specific requirements. This section explores the anatomy of custom widgets and their integration into the Flutter framework.

Layout and Constraints:
Flutter's layout system is based on the concept of constraints, allowing widgets toadapt to various screen sizes and orientations. Understanding how widgets handle layout constraints is essential for creating responsive and adaptive user interfaces.

Animation with Widgets:
Flutter provides a robust animation framework that seamlessly integrates with widgets, enabling the creation of fluid and engaging user experiences. This section explores how animations can be incorporated into Flutter widgets to enhance the overall user interface.

Testing and Debugging Widgets:
As with any software development, testing and debugging are integral parts of the process. This section discusses strategies for testing and debugging Flutter widgets, ensuring the reliability and quality of the UI components.

Container:
The Container widget is a basic building block that can contain other widgets and isoften used to define the dimensions, padding, margin, and decoration of a UI element.

Text:
The Text widget is used to display a paragraph or a line of text. It supports variousstyling options such as font size, color, and alignment.

ListView:
The ListView widget is used to create a scrollable list of widgets. It can display alarge number of children efficiently.

Row and Column:
Row and Column widgets are used to arrange children in a horizontal or verticalline, respectively.

Stack:
The Stack widget allows you to overlay multiple widgets on top of each other. It'soften used for complex layouts.

AppBar:
The AppBar widget represents the top app bar that usually contains the app's title,icons, and actions.

Experiment 2

Code:

main.dart :

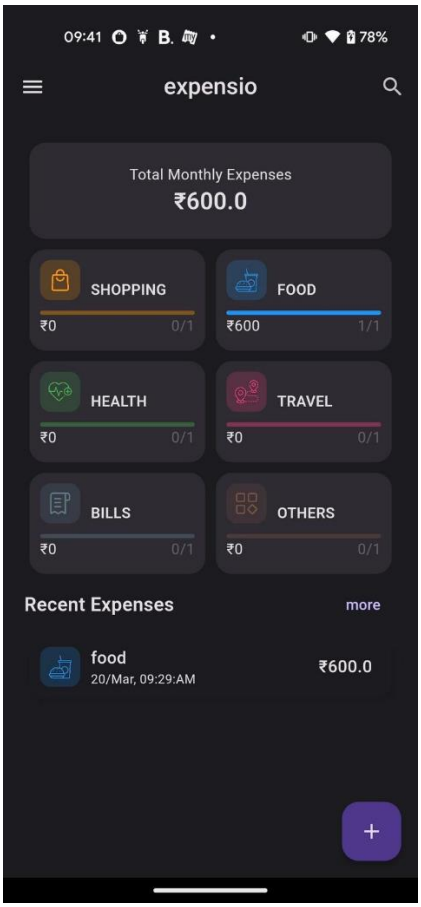
```
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import '/domain/bloc/expenses_bloc/expenses_bloc.dart';
import '/domain/bloc/config_bloc/config_bloc.dart';
import '/domain/bloc/auth_bloc/auth_bloc.dart';
import '/presentation/screens/splash/splash.dart';
import '/data/repositories/initialize_db.dart';
import '/core/config/app_themes.dart';
import 'core/config/firebase_options.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  await InitializeDb.initLocalDb();
  runApp(const MyApp());
}
class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MultiBlocProvider(
      providers: [
        BlocProvider(
          create: (context) => AuthBloc()..add(AuthEventCheckLoggedInUser()),
        ),


        BlocProvider(
          create: (context) => ConfigBloc()..add(ConfigGetEvent()),
        ),
        BlocProvider(create: (context) => ExpensesBloc()),
      ],
      child: BlocBuilder<ConfigBloc, ConfigState>(
        builder: (context, state) => MaterialApp(
          title: 'Expensio',
          home: const SplashScreen(),
          debugShowCheckedModeBanner: false,
          theme: AppTheme.lightTheme,
          darkTheme: AppTheme.darkTheme,
          themeMode: context.watch<ConfigBloc>().themeMode,
        ),
      ),
    );
  }
}
```

Experiment 2

Output:



Conclusion:
Flutter's widget architecture offers great flexibility for building complex UIs. Understanding key widgets and concepts is essential for effective Flutter development.