EXPERIMENT NO. 2

Aim: To design Flutter UI by including common widgets.

Theory:

Flutter widgets are built using a modern framework that takes inspiration from React. Widgets describe what their view should look like given their current configuration and state. When a widget's state changes, the widget rebuilds its description, which the framework differs against the previous description in order to determine the minimal changes needed in the underlying render tree to transition from one state to the next.
Flutter comes with a suite of powerful widgets, of which the following are used making the Login Screen of our application.

A Stateful widget is a widget that describes part of the user interface by building a constellation of other widgets that describe the user interface more concretely. They are useful when the part of the user interface you are describing can change dynamically, e.g. due to having an internal clock-driven state, or depending on some system state. Following is an example of the widget I have used in the Login Screen.

The examples of the StatefulWidget are Checkbox, Radio, Slider, InkWell, Form, and TextField. The Scaffold is a widget in Flutter used to implement the basic material design visual layout structure. It is quick enough to create a general-purpose mobile application and contains almost everything we need to create a functional and responsive Flutter app. This widget is able to occupy the whole device screen. Following is an example of the widget I have used in the Login Screen.

The following are the properties and description of the Scaffold widget class.

Property	Description
appBar	It is a horizontal bar that is mainly displayed at the top of the Scaffold widget.
body	It is the other primary and required property of this widget, which will display the main content in the Scaffold.
drawer	It is a slider panel that is displayed at the side of the body.
floatingActionButton	It is a button displayed at the bottom right corner and floating above the body.
backgroundColor	This property is used to set the background color of the whole Scaffold widget.
primary	It is used to tell whether the Scaffold will be displayed at the top of the screen or not.
persistentFooterButton	It is a list of buttons that are displayed at the bottom of the Scaffold widget.
bottomNavigationBar	This property is like a menu that displays a navigation bar at the bottom of the Scaffold.
endDrawer	It is similar to a drawer property, but they are displayed at the right side of the screen by default.
resizeToAvoidBottomInset	Scaffold's floating widgets adjust their size themselves to avoid the onscreen keyboard.
floatingActionButtonLocation	By default, it is positioned at the bottom right corner of the screen.

The Column widget arranges its children in a vertical direction on the screen. In other words, it will expect a vertical array of children widgets. A column widget does not appear scrollable because it displays the widgets within the visible view. Flutter column widget has several other properties like mainAxisSize, textDirection, verticalDirection, etc. These are mainAxisAlignment and crossAxisAlignment properties.

The SizedBox widget is a box with a specified size. If given a child, this widget forces it to have a specific width and/or height. These values will be ignored if this widget's parent does not permit them. In the Login Screen, I have used the SizedBox as a vertical space between the widgets which are placed in the screen.

The Row widget arranges its children in a horizontal direction on the screen. A row widget does not appear scrollable because it displays the widgets within the visible view. Flutter row widget has several other properties like mainAxisSize, textDirection, verticalDirection, etc. These are mainAxisAlignment and crossAxisAlignment properties.

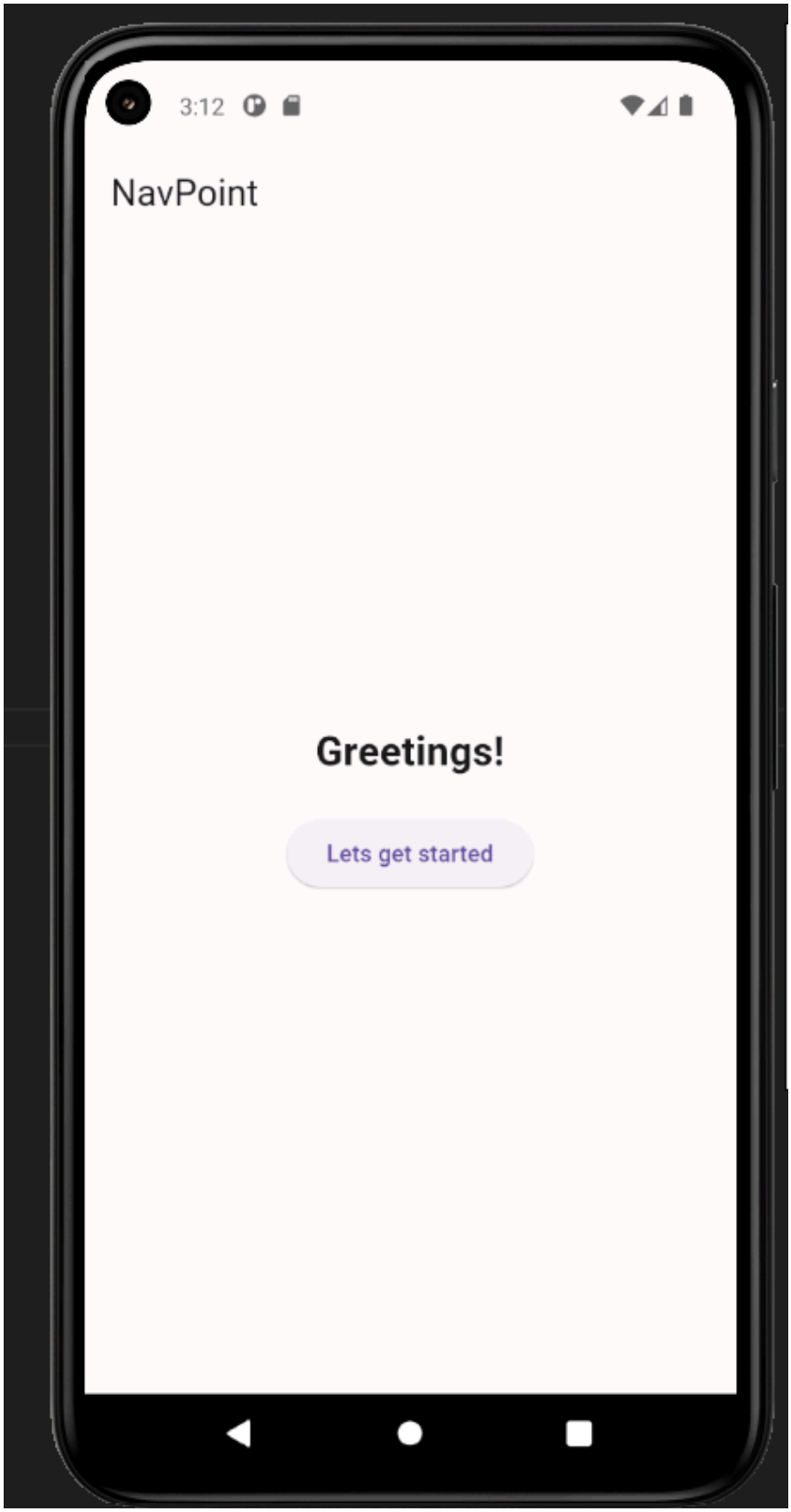
Property	Description
start	It will place the children from the starting of the main axis.
end	It will place the children at the end of the main axis.
center	It will place the children in the middle of the main axis.
spaceBetween	It will place the free space between the children evenly.
spaceAround	It will place the free space between the children evenly and half of that space before and after the first and last children widget.
spaceEvenly	It will place the free space between the children evenly and before and after the first and last children widget.

Code:

```
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(
      debugShowCheckedModeBanner: false,
      title: 'NavPoint',
      theme: ThemeData.light(),
      home: Scaffold(
        appBar: AppBar(
          title: Text('NavPoint'),
        ),
        body: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
```

```
            children: [
              Text(
                'Greetings!',
                style: TextStyle(
                  fontSize: 24,
                  fontWeight: FontWeight.bold,
                ),
              ),
              SizedBox(height: 20),
              ElevatedButton(
                onPressed: () {
                  // Add your functionality here
                },
                child: Text('Let\'s get started'),
              ),
              SizedBox(height: 10),
            ],
          ),
        ),
      ),
    );
  }
}
```

Output:



Conclusion: We have successfully designed Flutter UI by including common widgets.