Name:Priyank Kulkarni Roll no.: 32 Class: D15B

MAD Experiment 5

Aim : To apply navigation, routing and gestures in Flutter App.

Theory :

Flutter Navigation and Routing :

Navigation and routing are some of the core concepts of all mobile application, which allows the user to move between different pages. We know that every mobile application contains several screens for displaying different types of information. For example, an app can have a screen that contains various products. When the user taps on that product, immediately it will display detailed information about that product. In Flutter, the screens and pages are known as routes, and these routes are just a widget. In Android, a route is similar to an Activity, whereas, in iOS, it is equivalent to a ViewController. In any mobile app, navigating to different pages defines the workflow of the application, and the way to handle the navigation is known as routing. Flutter provides a basic routing class MaterialPageRoute and two methods Navigator.push() and Navigator.pop() that shows how to navigate between two routes. The following steps are required to start navigation in your application.

Step 1: First, you need to create two routes.

Step 2: Then, navigate to one route from another route by using the Navigator.push() method.

Step 3: Finally, navigate to the first route by using the Navigator.pop() method.

Gestures :

Gestures are used to interact with an application. It is generally used in touch-based devices to physically interact with the application. It can be as simple as a single tap on the screen to a more complex physical interaction like swiping in a specific direction to scrolling down an application. It is heavily used in gaming and more or less every application requires it to function as devices turn more touch-based than ever. In this article, we will discuss them in detail.

Some widely used gestures are mentioned here :

- Tap: Touching the surface of the device with the fingertip for a small duration of time period and finally releasing the fingertip.
- Double Tap: Tapping twice in a short time.
- Drag: Touching the surface of the device with the fingertip and then moving the fingertip steadily and finally releasing the fingertip.
- Flick: Similar to dragging, but doing it in a speedier way.

- Pinch: Pinching the surface of the device using two fingers. Zoom: Opposite of pinching.
- Panning: Touching the device surface with the fingertip and moving it in the desired direction without releasing the fingertip.

The GestureDetector widget in flutter is used to detect physical interaction with the application on the UI. If a widget is supposed to experience a gesture, it is kept inside the GestureDetector widget. The same widget catches the gesture and returns the appropriate action or response.

Below is the list of gestures and their corresponding events :
Tap
- onTapDown
- onTapUp
- onTap
- onTapCancel

Double tap
- onDoubleTap

Long press
- onLongPress

Vertical drag
- onVerticalDragStart
- onVerticalDragUpdate
- onVerticalDragEnd

Horizontal drag
- onHorizontalDragStart
- onHorizontalDragUpdate
- onHorizontalDragEnd

Pan
- onPanStart
- onPanUpdate
- onPanEnd

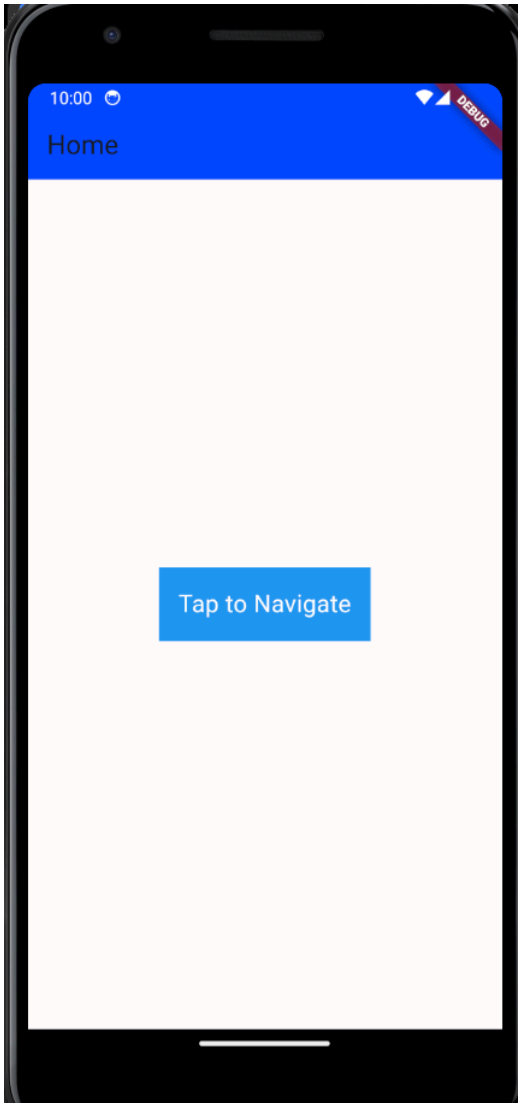
Main.dart:

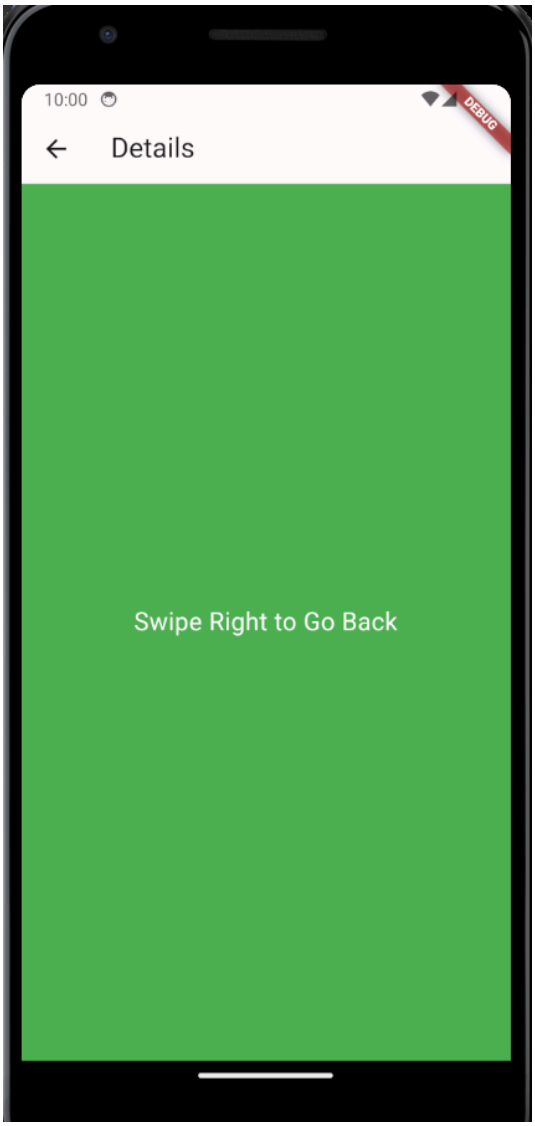
```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}
```

```dart
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Navigation and Gestures Example',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      initialRoute: '/',
      routes: {
        '/': (context) => HomeScreen(),
        '/details': (context) => DetailsScreen(),
      },
    );
  }
}

class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor:Color.fromARGB(255, 1, 73, 255),
        title: Text('Home'),
      ),
      body: Center(
        child: GestureDetector(
          onTap: () {
            Navigator.pushNamed(context, '/details');
          },
          child: Container(
            padding: EdgeInsets.all(16.0),
            color: Colors.blue,
            child: Text(
              'Tap to Navigate',
              style: TextStyle(color: Colors.white, fontSize: 20.0),
            ),
          ),
```

```
        ),
      ),
    );
  }
}

class DetailsScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Details'),
      ),
      body: GestureDetector(
        onHorizontalDragEnd: (details) {
          if (details.velocity.pixelsPerSecond.dx > 0) {
            Navigator.pop(context);
          }
        },
        child: Container(
          padding: EdgeInsets.all(16.0),
          color: Colors.green,
          child: Center(
            child: Text(
              'Swipe Right to Go Back',
              style: TextStyle(color: Colors.white, fontSize: 20.0),
            ),
          ),
        ),
      ),
    );
  }
}
```

Output:





Conclusion:

We have understood the concept of gestures, their use and implemented it in our flutter app as a search bar. Also, we created two pages and route it in our app and navigate using Navigate.push method. The flow is smooth for our app.