Name: Anushka Karhadkar
Class : D15B
Roll No : 28

Experiment No. 5

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

Theory:

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.

Navigation:
Navigation in Flutter refers to the ability to move between different screens or pages within an app. Flutter provides a Navigator widget to manage the navigation stack and perform common navigation operations.

Navigation Operations:

- Pushing a Screen:

  Use Navigator.push to navigate to a new screen.

  Example:
  Navigator.push(context, MaterialPageRoute(builder: (context) =>
  DetailsScreen()));

- Popping a Screen:

  Use Navigator.pop to go back to the previous screen.

  Example:
  Navigator.pop(context);

Routing:
Routing, in the context of Flutter, involves defining the paths or routes that lead to different screens in your app. It allows you to organize and structure the flow of your application. Flutter supports both named routes and unnamed (or default) routes.

- Named Routes:

  Named routes are routes identified by a unique string identifier.
  They provide a more organized and maintainable way to navigate between screens.
  You can define named routes in the MaterialApp widget using the routes property.

  Example,

```
MaterialApp(
  routes: {
    '/': (context) => HomeScreen(),
    '/details': (context) => DetailsScreen(),
  },
)
```

- Unnamed (Default) Routes:

  -Unnamed routes refer to the default route that is used when the app starts.
  -You can set the initialRoute property in MaterialApp to specify the initial route.

  Example,
  MaterialApp(

```
    initialRoute: '/',
    routes: {
      '/': (context) => HomeScreen(),
      '/details': (context) => DetailsScreen(),
    },
  )
```

Gestures in Flutter:
Gestures in Flutter refer to user interactions with the screen, such as taps, swipes, pinches, etc.
Gestures are an interesting feature in Flutter that allows us to interact with the mobile app (or any touch-based device). Generally, gestures define any physical action or movement of a user in the intention of specific control of the mobile device.
Flutter provides various widgets for gesture detection:

GestureDetector:

The GestureDetector widget is a versatile widget that can detect various gestures. It wraps its child and provides callbacks for different types of gestures.

Example,

```
GestureDetector(
  onTap: () {
    // Handle tap gesture
  },
  onDoubleTap: () {
    // Handle double tap gesture
  },
  // Other gesture callbacks...
  child: YourWidget(),
)
```

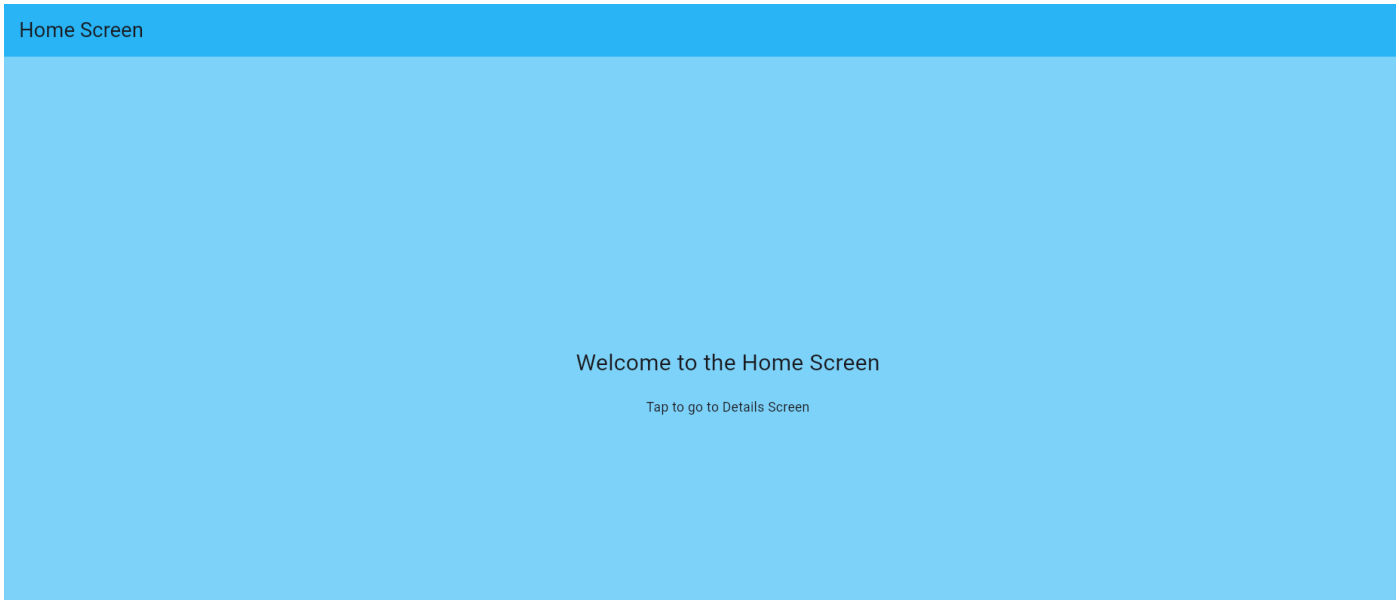
Code:

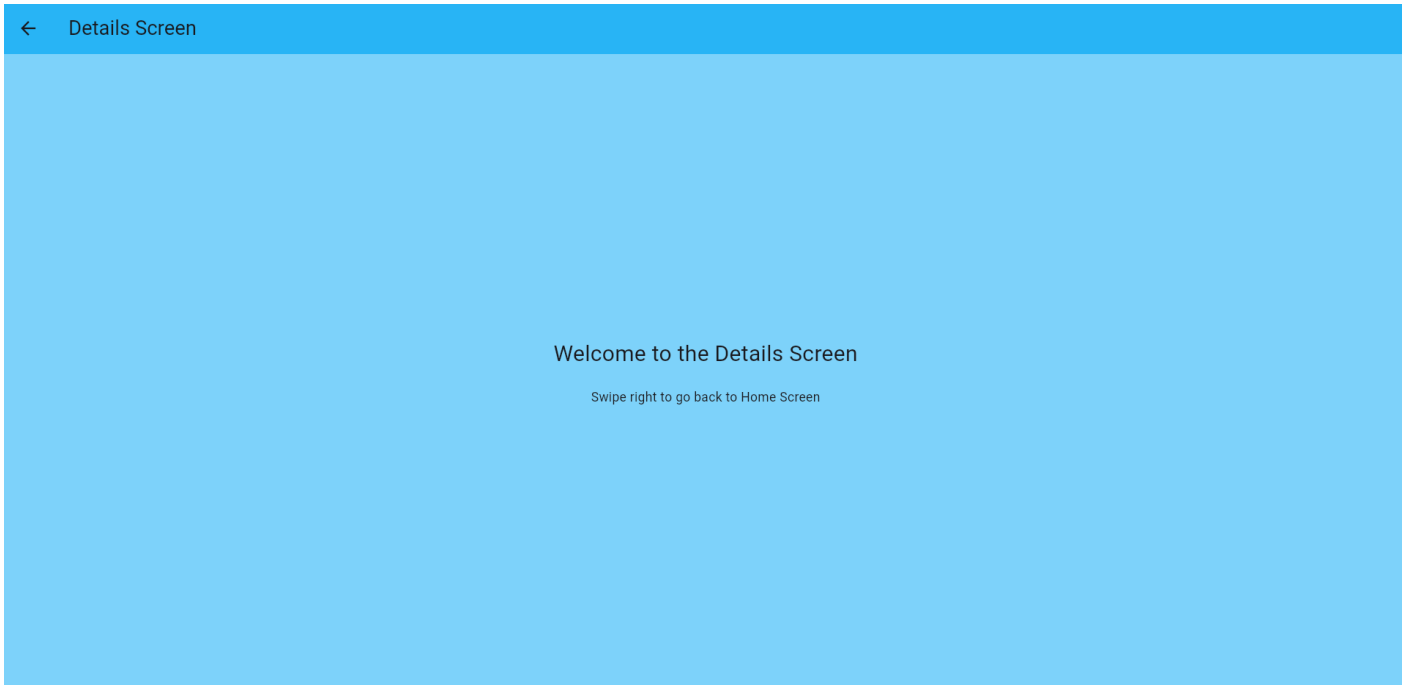
```dart
import 'package:flutter/material.dart';

void main() {
```

```dart
  runApp(
    MaterialApp(
      debugShowCheckedModeBanner: false,
      initialRoute: '/',
      routes: {
        '/': (context) => HomeScreen(),
        '/details': (context) => DetailsScreen(),
      },
    ),
  );
}

class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Home Screen'),
      ),
      backgroundColor: Colors.lightBlue[200], // Set background color
      body: Center(
        child: GestureDetector(
          onTap: () {
            Navigator.pushNamed(context, '/details');
          },
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text(
                'Welcome to the Home Screen',
                style: TextStyle(fontSize: 24.0),
              ),
              SizedBox(height: 20.0),
              Text('Tap to go to Details Screen'),
            ],
          ),
        ),
      ),
    );
  }
```

```dart
}

class DetailsScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Details Screen'),
      ),
      backgroundColor: Colors.lightBlue[200], // Set background color
      body: Center(
        child: GestureDetector(
          onHorizontalDragEnd: (details) {
            // Navigate back to the HomeScreen on right swipe
            if (details.primaryVelocity! > 0) {
              Navigator.pop(context);
            }
          },
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text(
                'Welcome to the Details Screen',
                style: TextStyle(fontSize: 24.0),
              ),
              SizedBox(height: 20.0),
              Text('Swipe right to go back to Home Screen'),
            ],
          ),
        ),
      ),
    );
  }
}
```

Output:


Home Screen
Welcome to the Home Screen
Tap to go to Details Screen


Details Screen
Welcome to the Details Screen
Swipe right to go back to Home Screen

Conclusion:

We understood the concept of Navigation,routing and gestures in flutter and implemented them by creating two pages home screen,details screen and demonstrated navigation,routing and gestures through them.