Nishtha Joshi
D15B 25
MPL EXP 5

AIM:- To apply navigation, routing and gestures in Flutter App

THEORY :-

Navigation Routing:

Navigation routing is the process of navigating between different screens or routes within a Flutter application. It's a fundamental aspect of creating a seamless user experience. Flutter provides a robust set of tools for handling navigation efficiently.

Essential Components:
1. Navigator Widget: At the core of Flutter's navigation system is the Navigator widget. It manages a stack of Route objects and offers methods for pushing, popping, and replacing routes.

2. Routes: Each screen or page in a Flutter app is represented by a Route object. Flutter supports two primary types of routes: MaterialPageRoute for standard material design transitions and CupertinoPageRoute for iOS-style transitions.

3. Named Routes: Rather than directly managing routes, you can define named routes using a Map of route names to builder functions. This approach streamlines navigation and makes the code more organized.

4. MaterialApp Routes: Within the MaterialApp widget, you can define named routes using the routes property. This simplifies navigation setup and enables seamless transitions between different screens using the Navigator widget.

Data Passing:
When navigating between routes, it's common to need to pass data. Flutter facilitates this by allowing data to be passed between routes using constructor arguments or route settings.

1. Route Arguments: To access arguments passed to a route, you can use ModalRoute.of(context).settings.arguments.

2. Navigation Result: Navigator provides a mechanism for receiving a result from a previously pushed route, which can be useful for managing state or updating data.

Gestures:
Gestures play a crucial role in enabling users to interact with UI elements through touch. Flutter offers a comprehensive suite of widgets and APIs for handling various gestures effectively.

Gesture Detector:
The GestureDetector widget in Flutter detects user gestures such as taps, double taps, long presses, drags, and scales.

1. Callbacks: GestureDetector provides callback properties like onTap, onDoubleTap, onLongPress, onPanUpdate, etc., enabling developers to respond to specific user actions.

2. Gesture Recognizers: Flutter employs gesture recognizers to interpret raw pointer events into higher-level gestures, providing a foundation for handling user interactions.

3. Custom Gestures: Developers can create custom gestures by combining lower-level gesture recognizers using GestureRecognizer, offering flexibility in handling unique touch interactions.

Gesture Feedback:
Providing visual feedback for gestures enhances the user experience and makes interactions more intuitive.

1. Feedback Widgets: Flutter provides widgets like InkWell, InkResponse, and Material that offer built-in feedback for touch interactions, enhancing the visual appeal of the app.

2. Animations can be utilized to provide visual feedback for gestures, such as scaling or changing colors upon tap, further enriching the user experience and making interactions feel responsive.

```dart
import 'package:flutter/material.dart';
import 'gesture.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: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        colorScheme: const ColorScheme.light(),
      ),
      home: const MyHomePage(),
      initialRoute: '/',
      routes: {
        '/gesture': (context) => const GesturePage(),
      },
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({Key? key}) : super(key: key);

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.blueGrey[200], // Changed background color
```

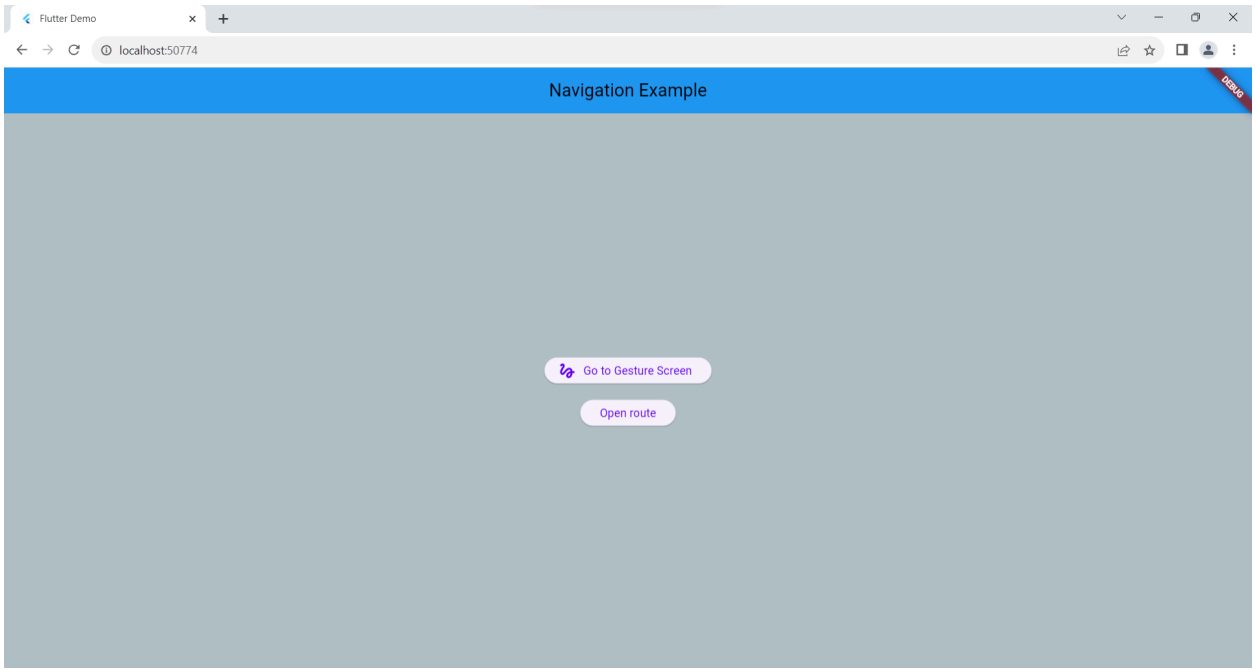
```
      appBar: AppBar(
        title: const Text('Navigation Example'),
        backgroundColor: Colors.blue,
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            ElevatedButton.icon(
              onPressed: () {
                Navigator.pushNamed(context, '/gesture');
              },
              icon: Icon(Icons.gesture), // Added icon
              label: const Text('Go to Gesture Screen'),
            ),
            const SizedBox(height: 20),
            ElevatedButton(
              onPressed: () {
                ScaffoldMessenger.of(context).showSnackBar(
                  const SnackBar(
                    content: Text('Unknown route. Please try again.'),
                  ),
                );
              },
              child: const Text('Open route'),
            ),
          ],
        ),
      ),
    );
  }
}
```

gesture.dart

```
import 'package:flutter/material.dart';
class GesturePage extends StatefulWidget {
const GesturePage({super.key});
@override
State<GesturePage> createState() => _GesturePageState();
}
class _GesturePageState extends State<GesturePage> {
@override
Widget build(BuildContext context) {
return Scaffold(
backgroundColor: Colors.red,
appBar: AppBar(
title: const Text('Gesture Example'),
),
body: Center(
child: Column(
mainAxisAlignment: MainAxisAlignment.center,
children: [
GestureDetector(
onTap: () {
ScaffoldMessenger.of(context).showSnackBar(
const SnackBar(
content: Text('Tap'),
),
);
},
onDoubleTap: () {
ScaffoldMessenger.of(context).showSnackBar(
const SnackBar(
content: Text('Double Tap'),
),
);
},
onLongPress: () {
ScaffoldMessenger.of(context).showSnackBar(
const SnackBar(
content: Text('Long Press'),
),
);
```

```
},
child: Container(
height: 150.0,
width: 150.0,
color: Colors.teal,
child: const Center(
child: Text('Tap Me'),
),
),
),
ElevatedButton(
onPressed:(){
Navigator.pop(context);
},
style: ElevatedButton.styleFrom(
backgroundColor: Colors.purple,
foregroundColor: Colors.black
),
child:const Text('Go to Previous Screen'),
)
],
),
),
);
}
}
```

Output:


Flutter Demo
localhost:50774
Navigation Example
DEBUG
Go to Gesture Screen
Open route