NAME:- RAGHAV MUNDHARA
CLASS :- D15B ROLL NO:- 41

EXPERIMENT NO 5

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

THEORY :-

Navigation Routing:
Navigation routing refers to the mechanism of navigating between different screens or routes within a Flutter application. Flutter provides several built-in widgets and packages to handle navigation routing efficiently.

Basic Navigation:

1. **Navigator Widget:** The Navigator widget manages a stack of Route objects and provides methods to push, pop, and replace routes.
2. **Routes:** Each screen in a Flutter app is typically represented by a Route object. Flutter offers two main types of routes: MaterialPageRoute for standard material design transitions and CupertinoPageRoute for iOS-style transitions.
3. **Named Routes:** Instead of directly pushing routes onto the navigator stack, you can define named routes using a Map of route names to builder functions. This approach provides a more organized way to manage navigation within the app.
4. **MaterialApp Routes:** In Flutter, you can define named routes within the MaterialApp widget using the routes property. This enables easy navigation between different screens using the Navigator widget.
5. **Passing Data**: When navigating between routes, you often need to pass data. Flutter allows you to pass data between routes using constructor arguments or route settings.
6. **Route Arguments:** You can use ModalRoute.of(context).settings.arguments to access arguments passed to a route.
7. **Navigation Result:** Navigator provides a mechanism for receiving a result from a previously pushed route.

Gestures:
Gestures in Flutter enable users to interact with the UI elements through touch. Flutter provides a rich set of widgets and APIs to handle various gestures effectively.

Gesture Detector:
GestureDetector Widget:

The GestureDetector widget in Flutter detects various user gestures such as tap, double tap, long press, drag, and scale.

1) **Callbacks:** GestureDetector provides callback properties like onTap, onDoubleTap, onLongPress, onPanUpdate, etc., which allow you to respond to specific user actions.
2) **Handling Gestures:**
    a) **Gesture Recognizers:** Flutter uses gesture recognizers under the hood to interpret raw pointer events into higher-level gestures.
    b) **Custom Gestures:** You can create custom gestures by combining lower-level gesture recognizers using GestureRecognizer.
    c) **Discrete Gestures:** Flutter provides discrete gestures like TapGestureRecognizer for handling tap events.
3) **Gesture Feedback:**
    a) **Feedback:** Providing visual feedback for gestures enhances the user experience. Flutter offers widgets like InkWell, InkResponse, and Material that provide built-in feedback for touch interactions.
    b) **Animations:** You can use animations to provide visual feedback for gestures, such as scaling or changing colors on tap.

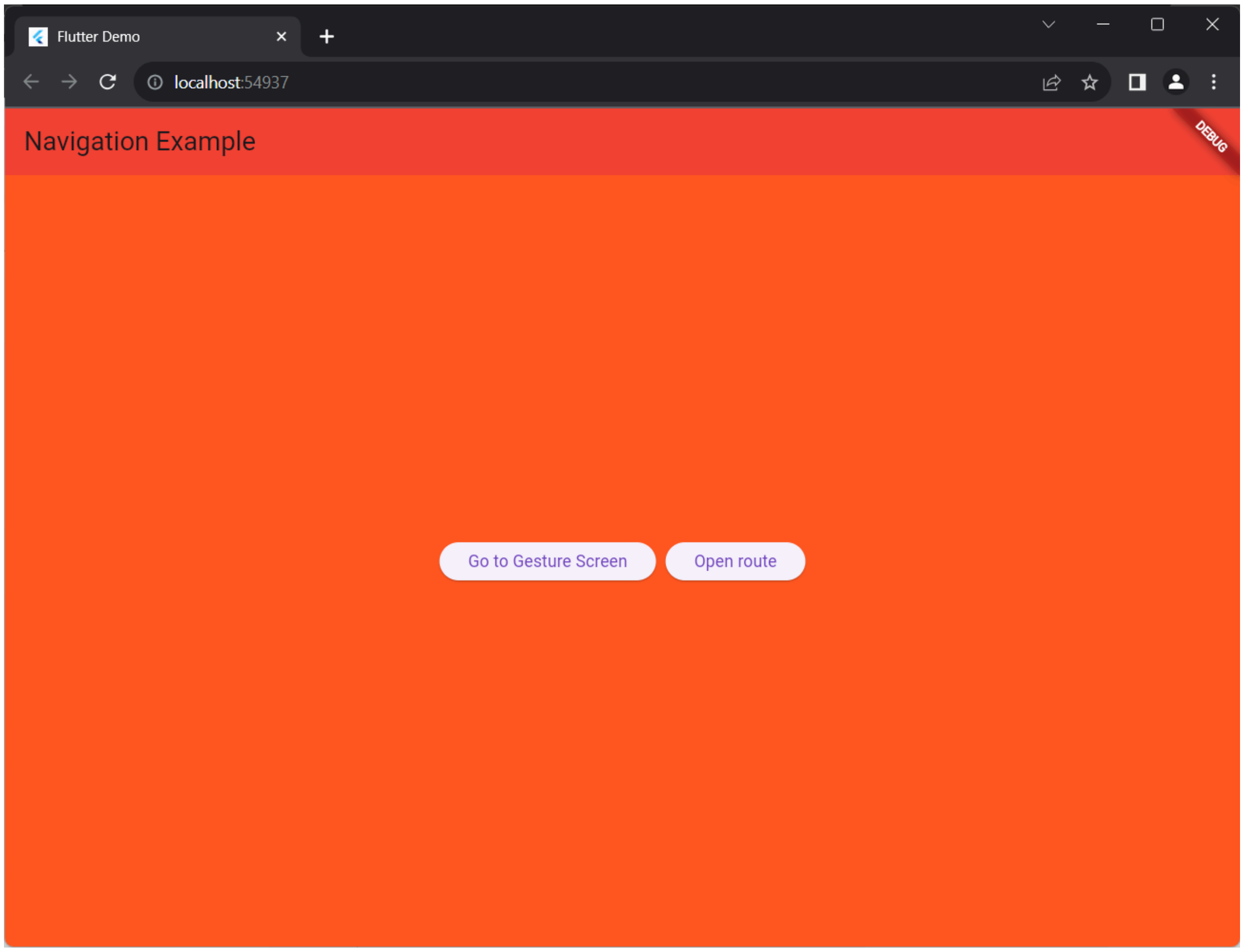
CODE & OUTPUT :-
main.dart

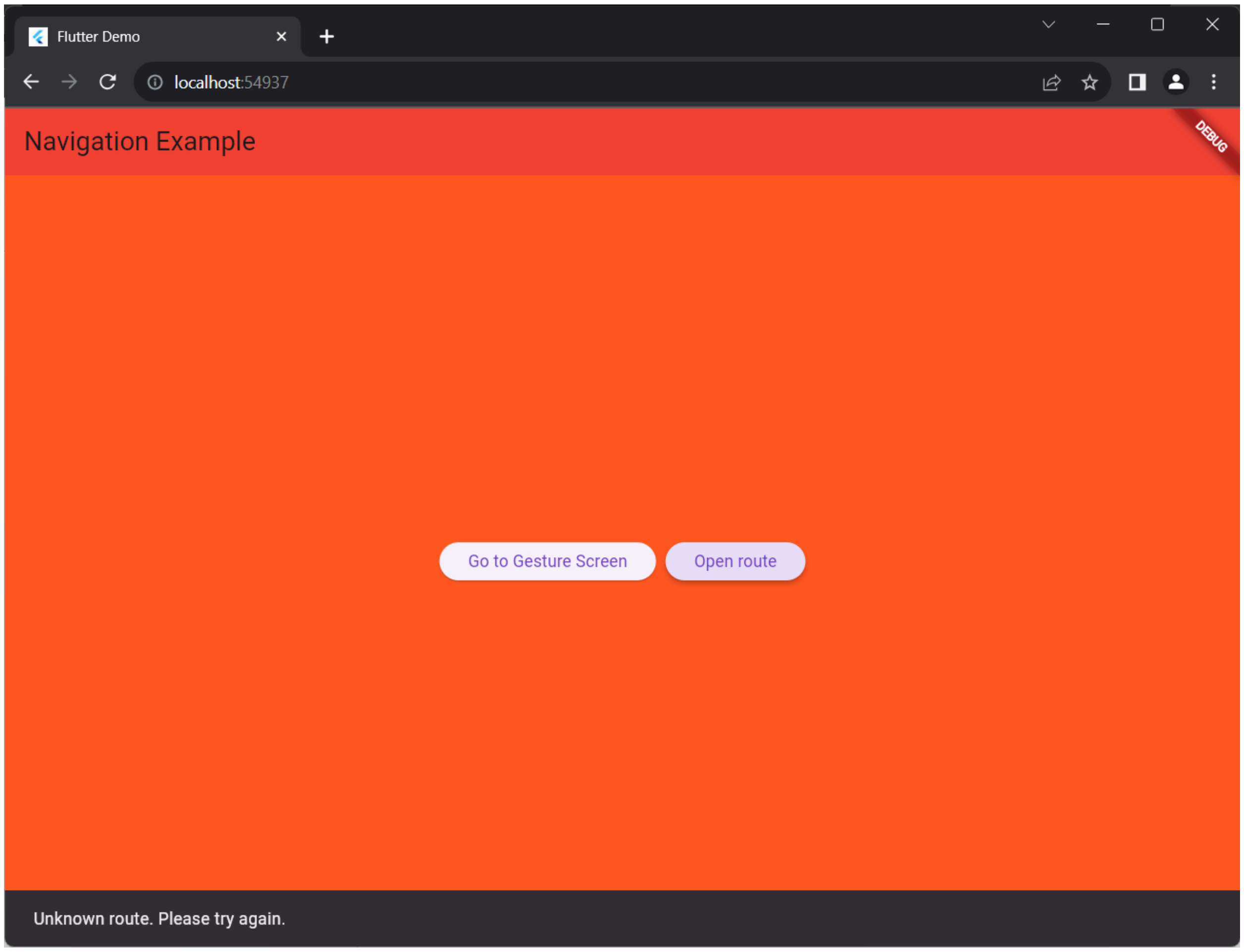
```
import 'package:expt5/gesture.dart';
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const MyHomePage(),
      initialRoute: '/',
      routes: {
        '/gesture': (context) => const GesturePage(),
      },
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key});

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

```
class _MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.deepOrange,
      appBar: AppBar(
        title: const Text('Navigation Example'),
        backgroundColor: Colors.red,
      ),
      body: Center(
        child: ButtonBar(
          alignment: MainAxisAlignment.center,
          children: <Widget>[
            ElevatedButton(
              onPressed: () {
                Navigator.pushNamed(context, '/gesture');
              },
              child: const Text('Go to Gesture Screen'),
            ),
            ElevatedButton(
              onPressed: () {
                ScaffoldMessenger.of(context).showSnackBar(
                  const SnackBar(
                    content: Text('Unknown route. Please try again.'),
                  ),
                );
              },
              child: const Text('Open route'),
            ),
          ],
        ),
      ),
    );
  }
}
```

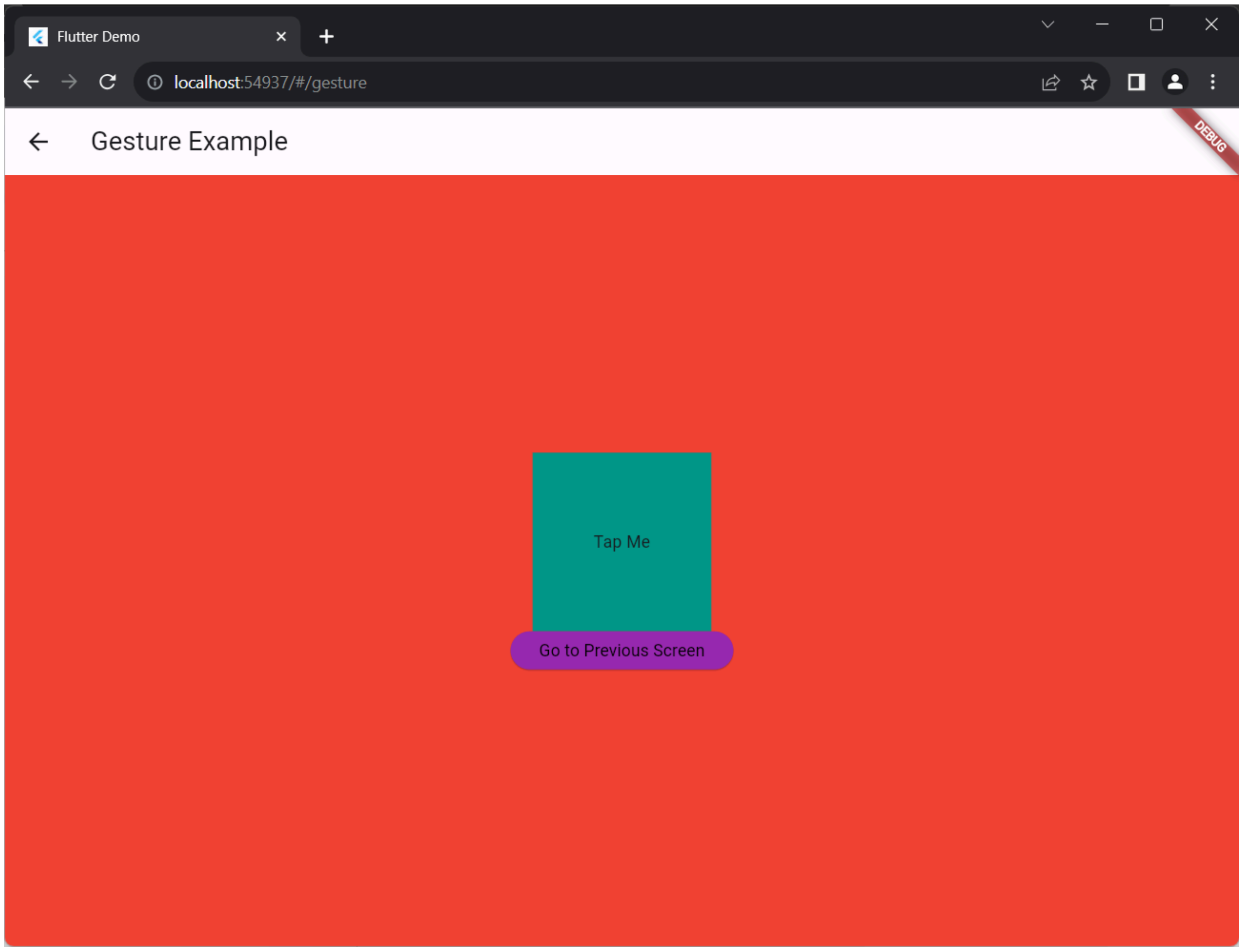
gesture.dart

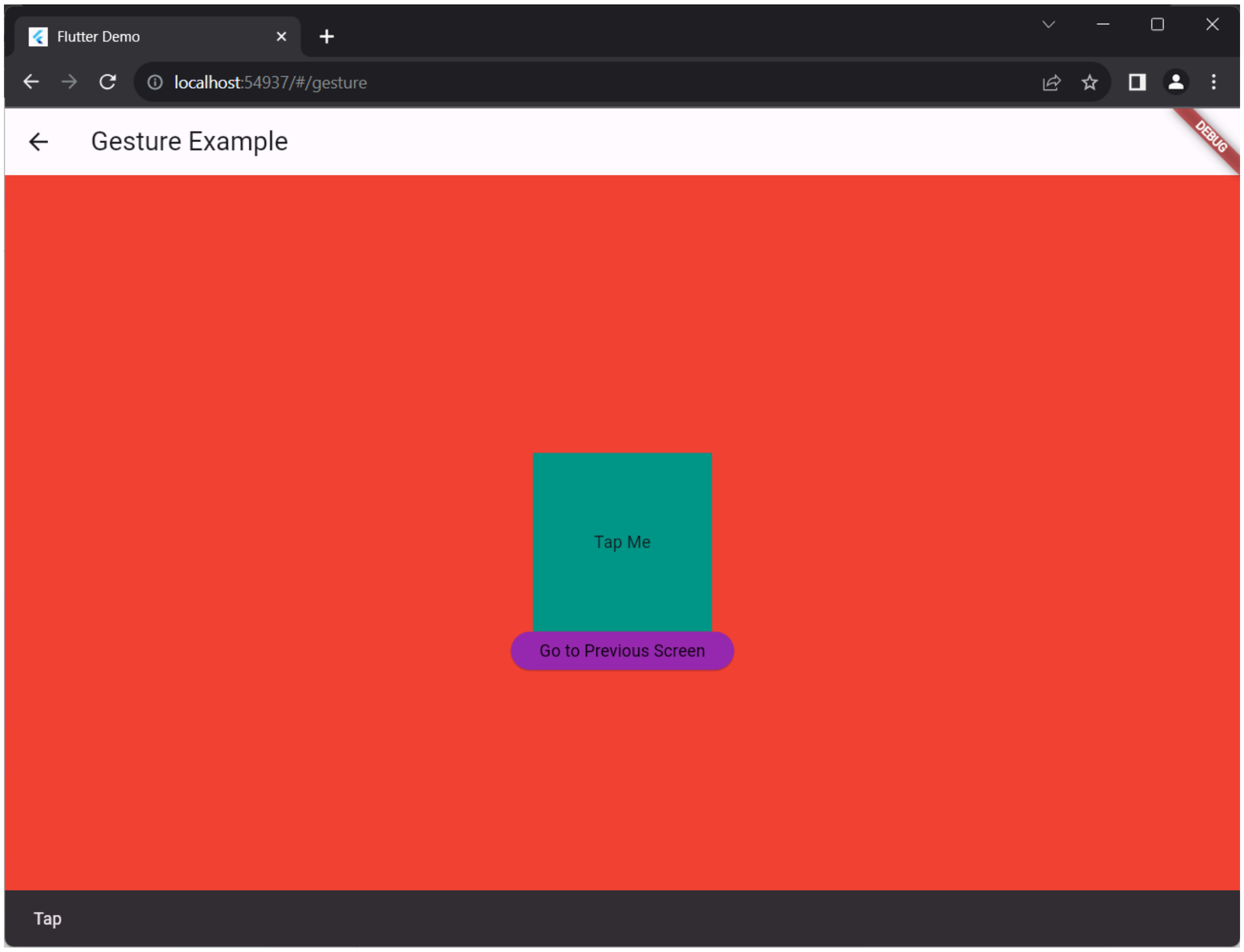
```
import 'package:flutter/material.dart';

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'),
            )
          ],
        ),
      ),
    );
  }
}
```

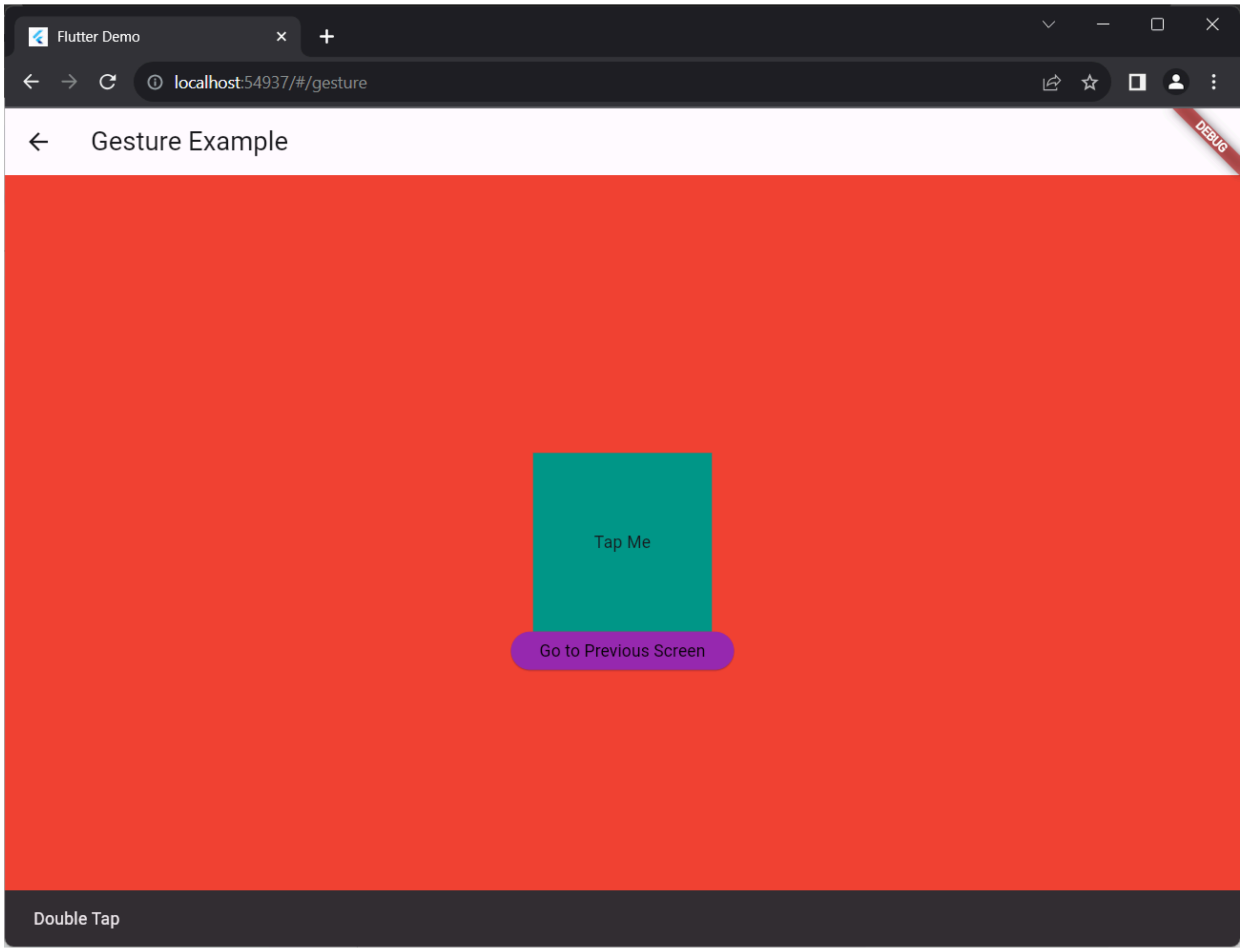

Flutter Demo
localhost:54937
Navigation Example
DEBUG
Go to Gesture Screen
Open route

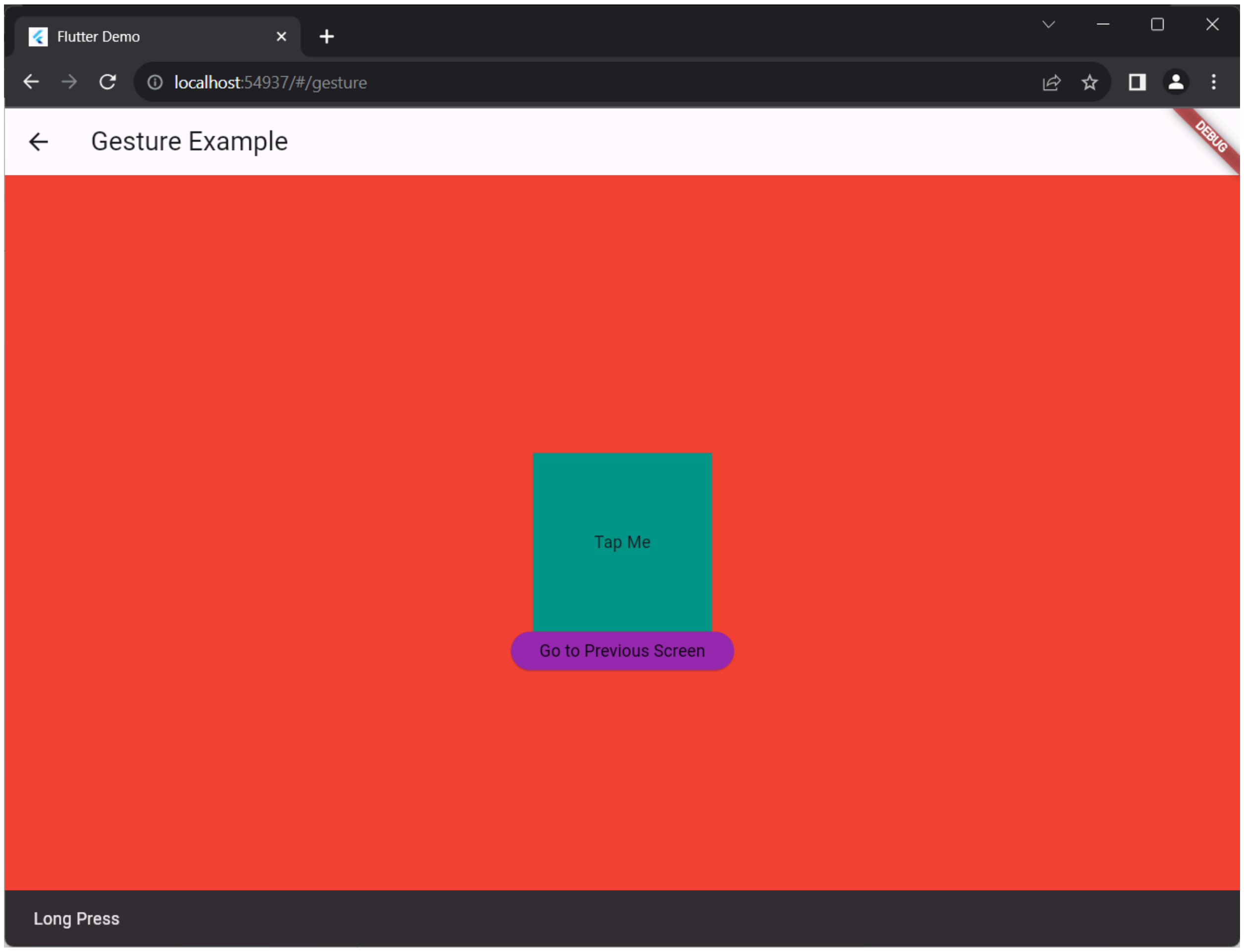


Unknown route. Please try again.


Flutter Demo
localhost:54937/#/gesture
Gesture Example
DEBUG
Tap Me
Go to Previous Screen


Flutter Demo
localhost:54937/#/gesture
Gesture Example
DEBUG
Tap Me
Go to Previous Screen
Tap


Flutter Demo
localhost:54937/#/gesture
Gesture Example
DEBUG
Tap Me
Go to Previous Screen
Double Tap



CONCLUSION :- Navigation routing and gestures are essential aspects of building intuitive and interactive Flutter applications. By leveraging the provided widgets and APIs effectively, developers can create seamless user experiences that are both visually appealing and easy to navigate.