Assignment No.: 05
Title: Applying Navigation, Routing, and Gestures in Flutter App
Name: Harshit Raheja
Class: D15B
Roll Number: 45

Aim:
To implement screen navigation, named routing, and gesture detection in a Flutter app.

Theory:
Flutter provides navigation mechanisms to move between screens (routes) using Navigator.push() and Navigator.pop(). It also supports named routing using MaterialApp's routes and initialRoute. Gestures like taps, drags, and long-presses can be captured using GestureDetector, enabling user interaction beyond buttons.

Steps for Navigation & Routing:

1. Create two screens.
2. Use Navigator.push() to go to the second screen.
3. Use Navigator.pop() to return to the first screen.
4. Alternatively, use named routes and Navigator.pushNamed() for scalable navigation.

Steps for Gesture Handling:

1. Wrap widgets with GestureDetector.
2. Define gesture callbacks like onTap, onDoubleTap, onLongPress.
3. For advanced use, use RawGestureDetector.

Extracted Code Snippets:

Navigation Example (Direct Push):

```
ElevatedButton(
 child: Text('Open route'),
 onPressed: () {
  Navigator.push(
   context,
   MaterialPageRoute(builder: (context) => SecondRoute()),
  );
 },
)
```

Navigation Example (Named Route):

```
Navigator.pushNamed(context, '/second');
```

Gesture Detector Example (Tap Gesture):

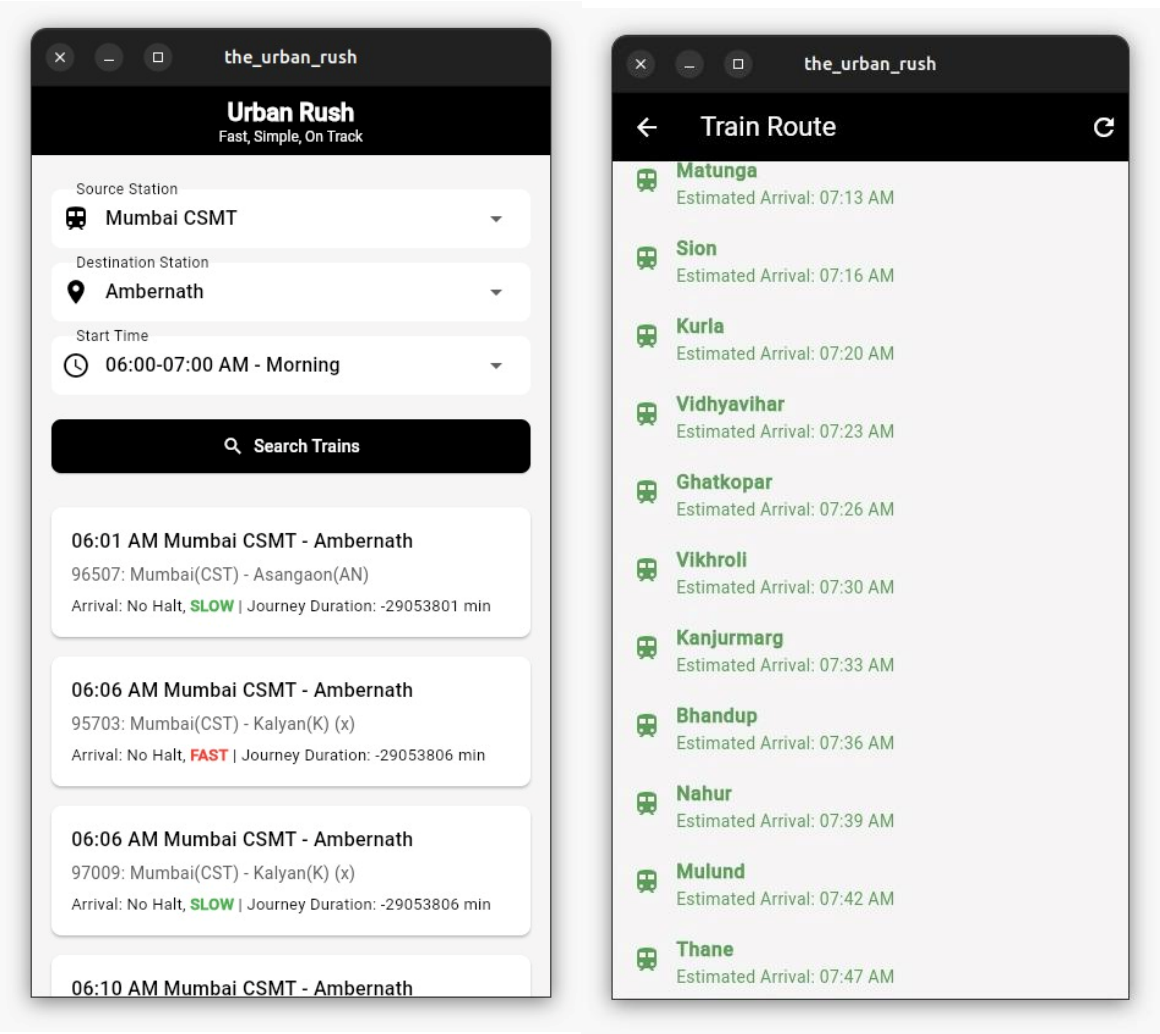
```
GestureDetector(
 onTap: () {
  print('Box Clicked');
 },
 child: Container(
  height: 60.0,
  width: 120.0,
  decoration: BoxDecoration(
   color: Colors.blueGrey,
   borderRadius: BorderRadius.circular(15.0),
  ),
  child: Center(child: Text('Click Me')),
 ),
)
```

Multiple Gesture Handling (Nested Taps):

```
RawGestureDetector(
 gestures: {
  AllowMultipleGestureRecognizer: GestureRecognizerFactoryWithHandlers<
     AllowMultipleGestureRecognizer>(
   () => AllowMultipleGestureRecognizer(),
   (instance) {
    instance.onTap = () => print('It is the parent container gesture');
   },
  )
 },
 child: Container(
  color: Colors.green,
  child: Center(
   child: RawGestureDetector(
    gestures: {
     AllowMultipleGestureRecognizer:
        GestureRecognizerFactoryWithHandlers<
           AllowMultipleGestureRecognizer>(
       () => AllowMultipleGestureRecognizer(),
       (instance) {
        instance.onTap = () => print('It is the nested container');
       },
      )
    },
    child: Container(
```

```
        color: Colors.deepOrange,
        width: 250.0,
        height: 350.0,
      ),
    ),
  ),
 ),
)
```

Output:



Conclusion:

Navigation and gesture functionality were successfully implemented in the Flutter app. Routes were created using both direct and named navigation techniques. Gestures were handled using GestureDetector and RawGestureDetector for layered interactions.