Name: Mohd Sufiyan Siddiqui
Class : D15A
Roll NO : 55

Lab - 05

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

Theory:
Here's a theoretical outline on how to apply navigation, routing, and gestures in a Flutter app:

1. Understanding Navigation and Routing :
   - Navigation : Navigation refers to moving between different screens or pages within your app.
   - Routing : Routing involves defining the paths or routes that users can take to navigate through your app.

2. Implementing Navigation :
   - Use the `Navigator` widget to manage navigation within your app. This widget maintains a stack of routes and handles transitions between them.
   - Define named routes for each screen/page in your app using the `MaterialApp` or `CupertinoApp` widget's `routes` property.
   - Use the `Navigator.push()` method to navigate to a new screen, and `Navigator.pop()` to return to the previous screen.

3. Handling Route Transitions :
   - Customize route transitions using the `PageRouteBuilder` class or predefined transition animations like `MaterialPageRoute` or `CupertinoPageRoute`.
   - You can specify transition animations, duration, and other parameters to create visually appealing navigation effects.

4. Implementing Gestures :
   - Use gesture detection widgets like `GestureDetector` to detect user gestures such as taps, swipes, drags, etc.
   - Wrap the relevant widgets with `GestureDetector` and provide callback functions to handle specific gestures.
   - GestureDetector provides properties like `onTap`, `onDoubleTap`, `onLongPress`, `onHorizontalDrag`, `onVerticalDrag`, etc., to capture various types of gestures.

5. Adding Gesture Navigation :
   - Implement gesture-based navigation features, such as swiping to navigate between screens or using gestures to trigger specific actions.
   - Combine gestures with navigation methods like `Navigator.push()` to achieve seamless navigation based on user interactions.

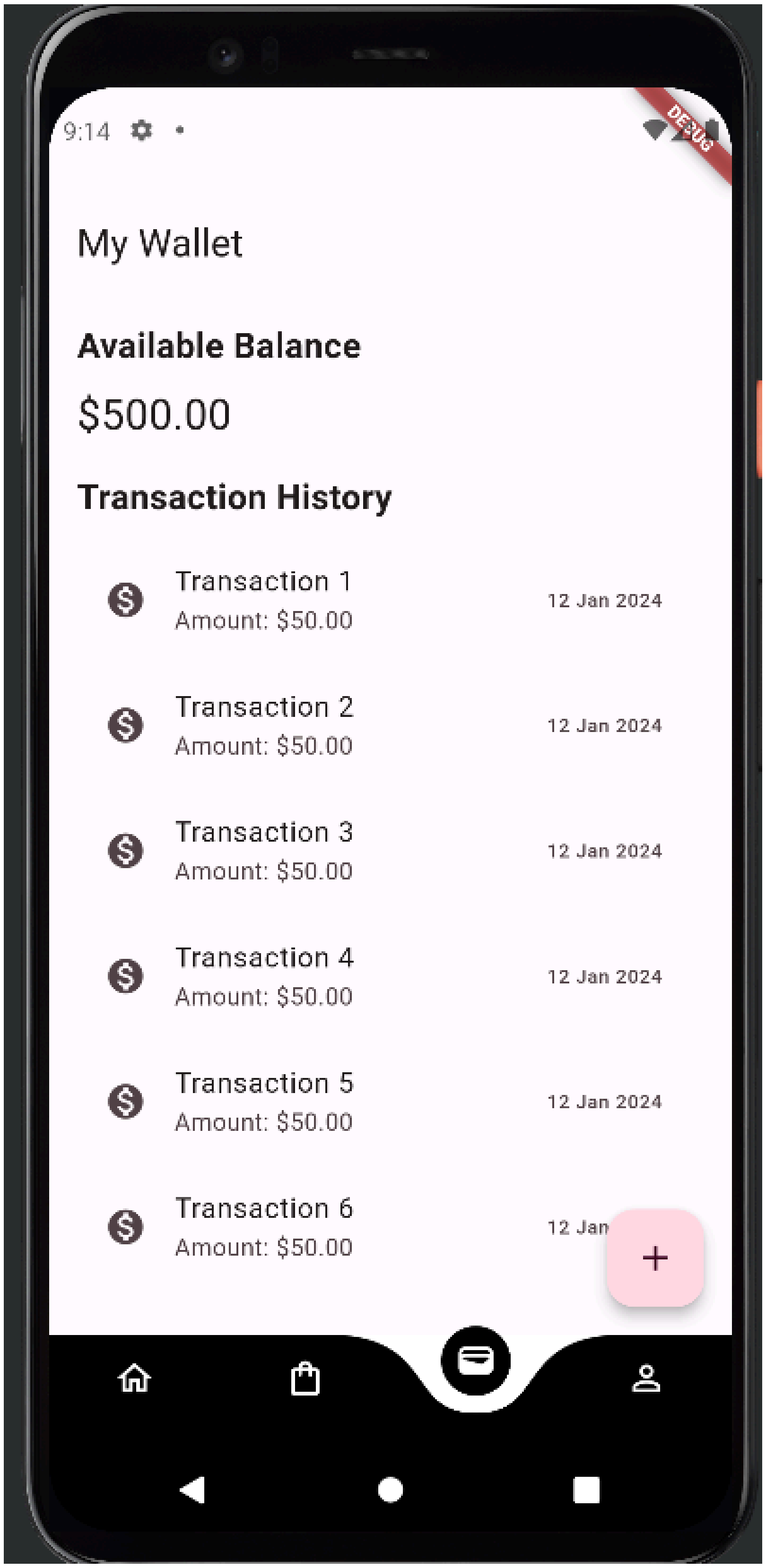
6. Handling Complex Gestures :
   - For more complex gestures, consider using gesture recognizers like `DragGestureRecognizer`, `ScaleGestureRecognizer`, `PanGestureRecognizer`, etc., to detect and handle intricate user interactions.
   - Customize gesture recognizers to suit your app's requirements by defining custom gesture recognizer subclasses.

7. Testing and Refinement :
   - Thoroughly test your navigation and gesture implementation to ensure smooth and intuitive user experience.
   - Solicit feedback from users and iterate on your implementation to address any usability issues or bugs.
   - Continuously refine and optimize your navigation and gesture handling based on user feedback and evolving app requirements.

By following these steps, you can effectively incorporate navigation, routing, and gestures into your Flutter app to create a seamless and interactive user experience.

Code:

```
import 'package:curved_navigation_bar/curved_navigation_bar.dart';
import 'package:flutter/material.dart';
import 'package:flutter_application_1/pages/Home.dart';
import 'package:flutter_application_1/pages/order.dart';
import 'package:flutter_application_1/pages/profile.dart';
import 'package:flutter_application_1/pages/wallet.dart';

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

  @override
  State<BottomNav> createState() => _BottomNavState();
}

class _BottomNavState extends State<BottomNav> {
  int currentTabIndex = 0;

  late List<Widget> pages;
  late Widget currentPage;
  late Home homepage;
  late Profile profile;
  late Order order;
  late Wallet wallet;

  @override
  void initState() {
    homepage = Home();
    order = Order();
    profile = Profile();
    wallet = Wallet();
    pages = [homepage, order, wallet, profile];
    //pages = [homepage, order, profile, profile];
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      bottomNavigationBar: CurvedNavigationBar(
          height: 65,
          backgroundColor: Colors.white,
          color: Colors.black,
          animationDuration: Duration(milliseconds: 500),
          onTap: (int index) {
            setState(() {
              currentTabIndex = index;
            });
          },
          items: [
            Icon(
              Icons.home_outlined,
              color: Colors.white,
            ),
            Icon(
              Icons.shopping_bag_outlined,
              color: Colors.white,
            ),
            Icon(
              Icons.wallet_outlined,
              color: Colors.white,
            ),
            Icon(
              Icons.person_outline,
              color: Colors.white,
            )
          ]),
      body: pages[currentTabIndex],
    );
  }
}
```



9:14

DEBUG

My Wallet

Available Balance

$500.00

Transaction History

Transaction 1
Amount: $50.00
12 Jan 2024

Transaction 2
Amount: $50.00
12 Jan 2024

Transaction 3
Amount: $50.00
12 Jan 2024

Transaction 4
Amount: $50.00
12 Jan 2024

Transaction 5
Amount: $50.00
12 Jan 2024

Transaction 6
Amount: $50.00
12 Jan