Experiment 5
Name:Khsuhi Jeswani
Div: D15A
Roll no: 63

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

- Navigation refers to the process of moving between different screens or pages within a Flutter app.
- In Flutter, navigation is typically managed using the Navigator class, which maintains a \ stack of routes.
- Each route represents a screen or page in the app, and the navigator manages the navigation stack, allowing users to move forward and backward between routes.
- Navigation can be triggered by user actions such as tapping buttons, selecting items from lists, or swiping between pages.

Routing:

- Routing is the mechanism used to define and manage the routes within a Flutter app.
- Routes are defined using route names and associated with corresponding widgets or screens.
- Flutter provides several routing mechanisms, including named routes, on-the-fly routes, and nested routes.
- Named routes allow developers to define routes with unique names and navigate to them using the Navigator based on these names.
- On-the-fly routes are created dynamically at runtime and pushed onto the navigation stack as needed.
- Nested routes involve embedding navigators within other navigators to create complex navigation structures, such as tab-based navigation or drawer navigation.

Gestures:

- Gestures refer to user interactions such as tapping, dragging, swiping, pinching, and rotating on the screen.
- Flutter provides a rich set of gesture recognition widgets and APIs to handle user gestures effectively.
- Common gesture recognition widgets include GestureDetector, InkWell, InkResponse, Draggable, Dismissible, etc.
- These widgets allow developers to detect various user gestures and trigger corresponding actions or animations in response.
- Gestures can be used to implement interactive UI elements, such as buttons, sliders, swipers, drag-and-drop interfaces, and more.

Gesture Detection:

- Gesture detection in Flutter involves registering gesture recognizers on widgets to detect

specific user interactions.

- Gesture recognizers analyze touch input and determine whether a specific gesture has occurred, such as a tap, double-tap, long-press, drag, etc.
- Once a gesture is detected, Flutter invokes the corresponding callback function associated with the gesture recognizer.
- Developers can customize gesture detection by configuring properties such as gesture sensitivity, velocity thresholds, and touch area boundaries.

Gesture Handling:

- After a gesture is detected, developers can handle it by performing various actions, such as updating UI state, navigating between screens, triggering animations, or executing business logic.
- Gesture handling involves responding to user interactions in a way that provides feedback and enhances the user experience.
- Flutter's declarative programming model makes it easy to update UI elements in response to user gestures, ensuring a smooth and responsive user interface.

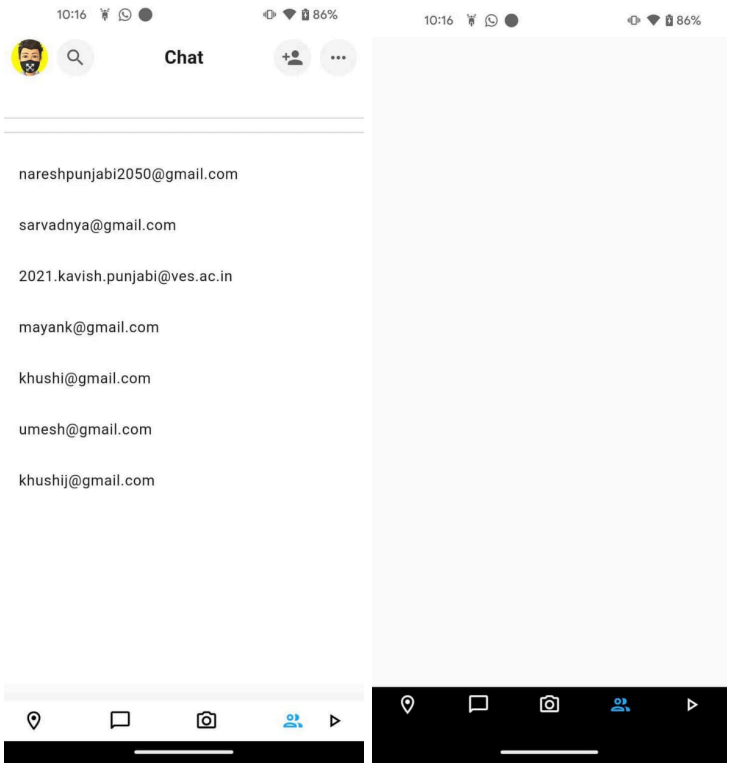
home_screen.dart

```
import 'package:flutter/material.dart';
import 'package:snapchatfinal/page/chat_screen.dart';
import 'reels_page.dart';
import 'stories_page.dart';
import 'caemra_page.dart';
import 'chat_page.dart';
import 'initial_page.dart';
import 'location_page.dart';

class HomePage extends StatefulWidget {
  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  int _selectedIndex = 0;
  static const List<Widget> _widgetOptions = <Widget>[
    LocationPage(),
    ChatPage(),
    CameraPage(),
    StoriesPage(),
    ReelPage(),
  ];

  void _onItemTapped(int index) {
```

```
  setState(() {
    _selectedIndex = index;
    if(index == 1){ // Check if the chat icon is tapped
      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => ChatPage2()), // Navigate to ChatScreen
      );
    }
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.white,
    body: SafeArea(child: _widgetOptions[_selectedIndex]),
    bottomNavigationBar: BottomNavigationBar(
      items: <BottomNavigationBarItem>[
        BottomNavigationBarItem(
          backgroundColor: Colors.black,
          icon: Icon(
            Icons.location_on_outlined,
            size: 25.0,
            color: Colors.white,
          ),
          label: "",
        ),
        BottomNavigationBarItem(
          backgroundColor: Colors.black,
          icon: Icon(
            Icons.chat_bubble_outline_rounded,
            size: 25.0,
            color: Colors.white,
          ),
          label: "",
        ),
        BottomNavigationBarItem(
          backgroundColor: Colors.black,
          icon: Icon(
            Icons.camera_alt_outlined,
            size: 25.0,
            color: Colors.white,
          ),
          label: "",
```

```
          ),
          BottomNavigationBarItem(
            backgroundColor: Colors.black,
            icon: Icon(
              Icons.group_outlined,
              size: 25.0,
              color: Color(0XFF10ACFF),
            ),
            label: "",
          ),
          BottomNavigationBarItem(
            backgroundColor: Colors.black,
            icon: Icon(
              Icons.play_arrow_outlined,
              size: 25.0,
              color: Colors.white,
            ),
            label: "",
          ),
        ],
        type: BottomNavigationBarType.fixed,
        currentIndex: _selectedIndex,
        selectedItemColor: Color(0XFF10ACFF),
        backgroundColor: Colors.black,
        onTap: _onItemTapped,
        unselectedItemColor: Colors.white,
      ),
    );
  }
}
```

App UI:



Widgets used: Images, Text, Bottom nav bar, Icons

Search page Upload page

Widgets used: Images, Text, Bottom nav bar, Icons, Bottom sheet, Text field

Conclusion: Therefore understood navigation, routing, gesture detection and gesture handling in Flutter and implemented the same in my Flutter application to route different pages.

Widgets used: Images, Text, Bottom nav bar, Icons

Search page Upload page

Widgets used: Images, Text, Bottom nav bar, Icons, Bottom sheet, Text field

Conclusion: Therefore understood navigation, routing, gesture detection and gesture handling in Flutter and implemented the same in my Flutter application to route different pages.