Name : Mrudula Gotmare **Class : D15A** **Roll no. 23**

Experiment - 5

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

Theory :

1. Navigation and routing
    a. In the MeetingCode widget's build method, a TransparentButton widget with the text 'Join Meeting' is defined.
    b. When this button is pressed, it triggers the onPressed callback, which navigates to the CallPage.
    c. Navigation is handled using Navigator.of(context).push(MaterialPageRoute(builder: (context)=> CallPage(callID: "1",userId: widget.userId, meetingCode: meetingCode)));.
    d. This code pushes a new route onto the Navigator's stack, which creates a new CallPage widget with the specified parameters (callID, userId, and meetingCode).

2. Gestures
    a. The 'Dismiss' text at the bottom of the MeetingCode widget is wrapped with a GestureDetector.
    b. When this text is tapped, it triggers the onTap callback, which calls Navigator.pop(context).
    c. Navigator.pop(context) removes the top route from the Navigator's stack, effectively closing the current screen and returning to the previous screen.

Code :

//meeting-code.dart

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

import 'package:gmeet_clone/screens/call.dart';

class MeetingCode extends StatefulWidget {
   final String userId;
  const MeetingCode({Key? key, required this.userId}) : super(key: key);

  @override
  State<MeetingCode> createState() => _MeetingCodeState();
}

class _MeetingCodeState extends State<MeetingCode> {
  late String meetingCode;
```

```
@override
void initState() {
  super.initState();
  // Generate a random meeting code
  generateMeetingCode();
}

void generateMeetingCode() {
  // Example meeting code "meet.google.com/uyz-vjvj-mbp"
  setState(() {
    meetingCode = 'meet.google.com/${_generateRandomCode()}';
  });
}

String _generateRandomCode() {
  const characters = 'abcdefghijklmnopqrstuvwxyz';
  final random = Random();
  String code = '';
  for (int i = 0; i < 10; i++) {
    code += characters[random.nextInt(characters.length)];
  }
  return code;
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.grey[900], // Set the background color to grey
    body: Center(
      child: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            Text(
              'Share this joining info with people that you want in the meeting',
              style: TextStyle(
                color: Colors.white,
                fontSize: 20,
              ),
              textAlign: TextAlign.center,
            ),
```

```
SizedBox(height: 24),
Container(
  padding: const EdgeInsets.all(8.0),
  decoration: BoxDecoration(
    color: Colors.grey[800],
    borderRadius: BorderRadius.circular(8.0),
  ),
  child: Row(
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: [
      Text(
        meetingCode ?? 'Generating...',
        style: TextStyle(
          color: Colors.indigo[200],
          fontSize: 18,
        ),
        textAlign: TextAlign.left,
      ),
      IconButton(
        onPressed: () {
          // Implement copy functionality
        },
        icon: Icon(
          Icons.content_copy,
          color: Colors.indigo[200],
        ),
      ),
    ],
  ),
),
SizedBox(height: 25),
Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    TransparentButton(
      text: 'Share',
      icon: Icons.share,
      onPressed: () {
        // Implement share functionality
      },
    ),
    TransparentButton(
      text: 'Join Meeting',
      icon: Icons.video_call,
```

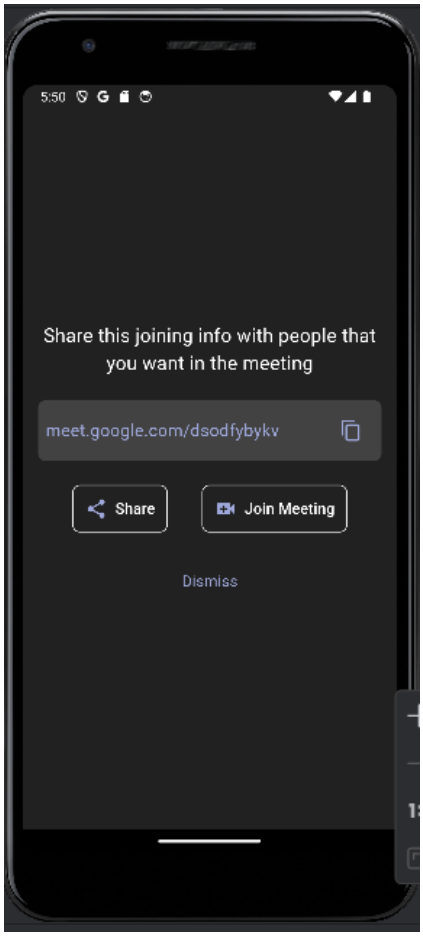
```
                onPressed: () {
                  // Implement join meeting functionality
                  Navigator.of(context).push(MaterialPageRoute(builder: (context)=>
CallPage(callID: "1",userId: widget.userId, meetingCode: meetingCode)));
                },
              ),
            ],
          ),
          SizedBox(height: 40),
          Center(
            child: GestureDetector(
              onTap: () {
                Navigator.pop(context); // Navigate back to the ContactList page
              },
              child: Text(
                'Dismiss',
                style: TextStyle(
                  color: Colors.indigo[200],
                  fontSize: 16,
                  decoration: TextDecoration.underline,
                ),
              ),
            ),
          ),
        ],
      ),
    ),
  );
 }
}

class TransparentButton extends StatelessWidget {
 final String text;
 final IconData? icon;
 final VoidCallback onPressed;

 const TransparentButton({
  required this.text,
  this.icon,
  required this.onPressed,
 });

 @override
```

```
  Widget build(BuildContext context) {
    return Container(
      decoration: BoxDecoration(
        border: Border.all(color: Colors.white),
        borderRadius: BorderRadius.circular(8.0),
      ),
      child: TextButton(
        onPressed: onPressed,
        child: Row(
          mainAxisSize: MainAxisSize.min,
          children: [
            if (icon != null) Icon(icon, color: Colors.indigo[200]),
            SizedBox(width: icon != null ? 8.0 : 0), // Add some spacing if there's an icon
            Text(
              text,
              style: TextStyle(
                color: Colors.white,
                fontSize: 16,
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```



Conclusion : These mechanisms allow users to navigate between screens and interact with the user interface using taps and gestures