Name: Niyati V. Gaonkar
Class: D15B Roll no.: 17

MAD-PWD Experiment - 5

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

THEORY:

Flutter Navigation

Navigation and routing are some of the core concepts of all mobile application, which allows the user to move between different pages. We know that every mobile application contains several screens for displaying different types of information. For example, an app can have a screen that contains various products. When the user taps on that product, immediately it will display detailed information about that product.

In Flutter, the screens and pages are known as routes, and these routes are just a widget. In Android, a route is similar to an Activity, whereas, in iOS, it is equivalent to a ViewController.

In any mobile app, navigating to different pages defines the workflow of the application, and the way to handle the navigation is known as routing. Flutter provides a basic routing class MaterialPageRoute and two methods Navigator.push() and Navigator.pop() that shows how to navigate between two routes. The following steps are required to start navigation in your application.

Step 1: First, you need to create two routes.

Here, we are going to create two routes for navigation. In both routes, we have created only a single button. When we tap the button on the first page, it will navigate to the second page. Again, when we tap the button on the second page, it will return to the first page. The below code snippet creates two routes in the Flutter application.

```
class FirstRoute extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('First Route'),
      ),
```

```
      body: Center(
        child: RaisedButton(
          child: Text('Open route'),
          onPressed: () {
            // Navigate to second route when tapped.
          },
        ),
      ),
    );
  }
}

class SecondRoute extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Second Route"),
      ),
      body: Center(
        child: RaisedButton(
          onPressed: () {
            // Navigate back to first route when tapped.
          },
          child: Text('Go back!'),
        ),
      ),
    );
  }
}
```

Step 2: Then, navigate to one route from another route by using the Navigator.push() method.
The Navigator.push() method is used to navigate/switch to a new route/page/screen. Here, the push() method adds a page/route on the stack and then manage it by using the Navigator. Again we use MaterialPageRoute class that

allows transition between the routes using a platform-specific animation. The below code explain the use of the Navigator.push() method.

```
// Within the `FirstRoute` widget
onPressed: () {
  Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => SecondRoute()),
  );
}
```

Step 3: Finally, navigate to the first route by using the Navigator.pop() method.
Now, we need to use Navigator.pop() method to close the second route and return to the first route. The pop() method allows us to remove the current route from the stack, which is managed by the Navigator.
To implement a return to the original route, we need to update the onPressed() callback method in the SecondRoute widget as below code snippet:

```
// Within the SecondRoute widget
onPressed: () {
  Navigator.pop(context);
}
```

There are three types of navigation that are common to all apps – stack, tab, and drawer navigation. Flutter supports all three types, and implementing them is similar to how you do it in other apps. But I found it super smooth to build navigation into my Flutter app.We'll build a Flutter app that uses drawer type of navigation in a single app so you can learn how they work.
Types of Navigation
There are three main types of navigation that you might use in your apps. Again, they are:

- Stack Navigation
- Tab Navigation
- Drawer Navigation

Let's understand how each one works.

- Stack Navigation

Picture a deck of cards, where you can add or remove cards from the top of the stack. Stack Navigation in Flutter works in a similar fashion. It helps you navigate between pages or screens by stacking new pages on top of existing ones. When you move to a new screen, the current screen is pushed onto the navigation stack, and when you return, the top screen is popped off the stack. This navigation type is commonly used for hierarchical and linear flows within an app.

- Tab Navigation

Tabs are a staple of mobile app navigation, allowing users to quickly switch between different sections or views without losing their current context. Flutter makes it easy to implement tabbed navigation with its built-in widgets, such as TabBar and TabBarView. By using these widgets, you can create a beautiful and functional tab navigation experience, perfect for organizing content into logical sections. You also have the freedom to customize the appearance of your tabs, making it simple to create a unique look and feel for your app.

- Drawer Navigation

The Drawer Navigation pattern, also known as the "hamburger menu" or "side menu," is a popular navigation style in mobile apps. It consists of a hidden panel that slides out from the side of the screen, revealing a menu with various navigation options.This space-saving technique keeps your app's main content visible while providing easy access to additional features or sections.

Let's start building the app and see how to implement each of these navigation features.

CODE:

my_drawer.dart :

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

class  MyDrawer extends StatelessWidget {
 const MyDrawer({super.key});
 void logout(){
   FirebaseAuth.instance.signOut();
 }
 @override
 Widget build(BuildContext context) {
   return Drawer(
```

```
    backgroundColor: Theme.of(context).colorScheme.background,
    child: Column(
      children: [
        DrawerHeader(
            child: Icon(Icons.favorite,
              color: Theme.of(context).colorScheme.inversePrimary,
            ),
        ),

        const SizedBox(height: 25,),

        Padding(
          padding: const EdgeInsets.only(left : 25.0),
          child: ListTile(
            leading: Icon(Icons.home ,
              color: Theme.of(context).colorScheme.inversePrimary,
            ),
            title: Text("H O M E"),
            onTap: (){
              Navigator.pop(context);
            },
          ),
        ),

        Padding(
          padding: const EdgeInsets.only(left : 25.0),
          child: ListTile(
            leading: Icon(Icons.person ,
              color: Theme.of(context).colorScheme.inversePrimary,
            ),
            title: Text("P R O F I L E"),
            onTap: (){
              Navigator.pop(context);
              Navigator.pushNamed(context, '/profile_page');
            },
          ),
        ),

        Padding(
          padding: const EdgeInsets.only(left : 25.0),
          child: ListTile(
            leading: Icon(Icons.group ,
              color: Theme.of(context).colorScheme.inversePrimary,
            ),
            title: Text("U S E R S"),
            onTap: (){
              Navigator.pop(context);
              Navigator.pushNamed(context, '/users_page');
            },
```

```
          ),
        ),

        Padding(
          padding: const EdgeInsets.only(left : 25.0),
          child: ListTile(
            leading: Icon(Icons.home ,
              color: Theme.of(context).colorScheme.inversePrimary,
            ),
            title: Text("L O G O U T"),
            onTap: (){
              Navigator.pop(context);
              logout();
            },
          ),
        ),

      ],
    ),
  );
 }
}
```

:

my_profile.dart

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


class ProfilePage extends StatelessWidget {
 ProfilePage({super.key});

 final User? currentUser = FirebaseAuth.instance.currentUser;

 Future<DocumentSnapshot<Map<String, dynamic>>> getUserDetails() async {
   return await FirebaseFirestore. instance
       .collection("Users")
       .doc(currentUser!.email)
       .get();

 }

 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: Text("Profile"),
     ),
     body: FutureBuilder<DocumentSnapshot<Map<String, dynamic>>>(
```

```
      future : getUserDetails(),
      builder: (context, snapshot){
        if( snapshot.connectionState== ConnectionState.waiting) {
          return const Center(
            child: CircularProgressIndicator(),
          );
        }

        else if(snapshot.hasError){
          return Text("Error : ${snapshot.error}");
        }

        else if (snapshot.hasData){
          Map<String, dynamic>? user = snapshot.data!.data();

          return Center(
            child: Column(
              children: [
                Text(user!['email']),
                Text(user['username']),
              ],
            ),
          );
        } else{
          return Text("No data");
        }
      },
    ),
  );
 }
}
```

users.dart

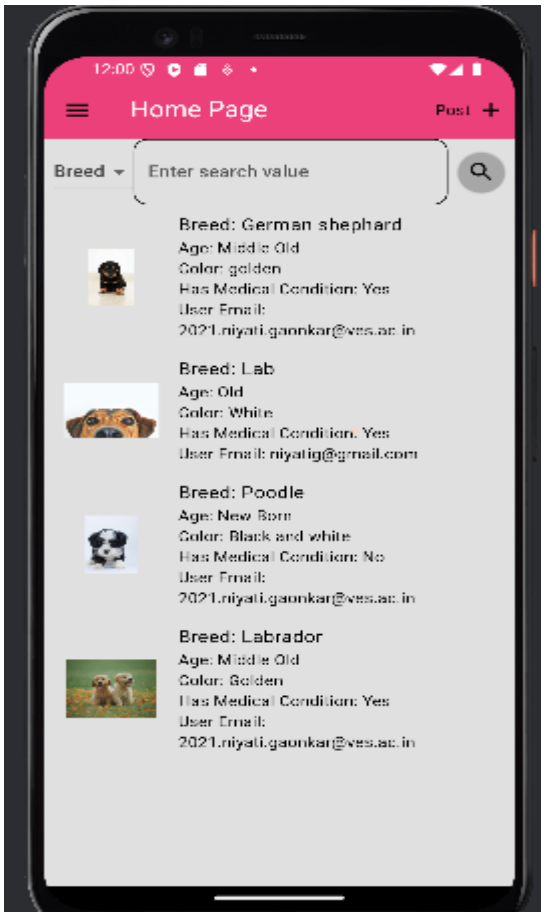
```
import 'package:flutter/material.dart';

class UsersPage extends StatelessWidget {
 const UsersPage({Key? key});

 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: Text("Users"),
     ),
   );
 }

 Widget _buildUserItem(String email) {
   return ListTile(
```

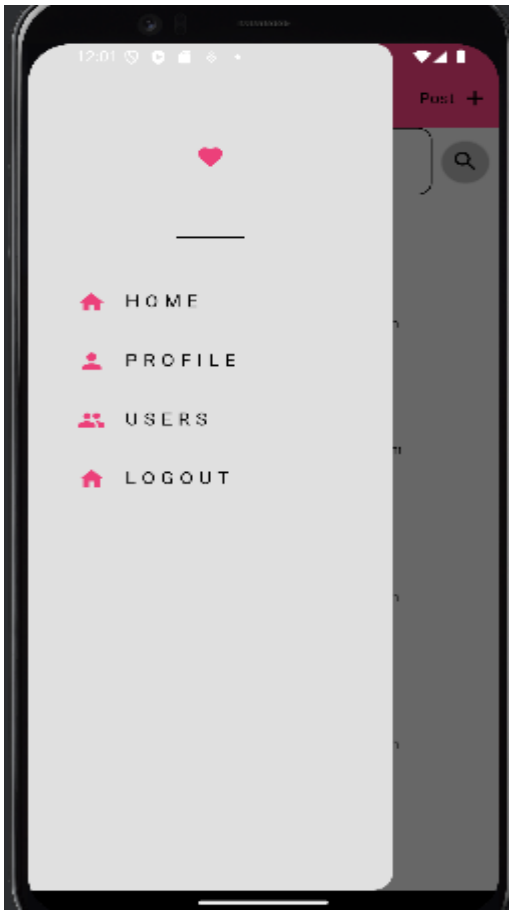
```
      title: Text("Useremail: $email"),
    );
  }
}
```

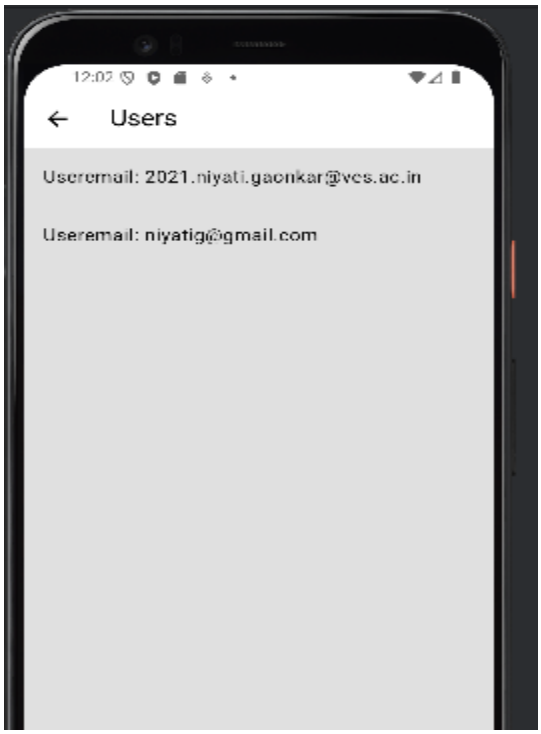
home.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:dog_adoption/components/my_post_button.dart';
import 'package:dog_adoption/components/my_textfield.dart';
import 'package:dog_adoption/database/firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:dog_adoption/components/my_drawer.dart';

import 'add_post_page.dart';

class HomePage extends StatefulWidget {
 HomePage({Key? key}) : super(key: key);

 @override
 _HomePageState createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
 void logout() {
   FirebaseAuth.instance.signOut();
 }

 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: Text(
         "Home Page",
         style: TextStyle(
           color: Colors.white, // Change text color here
         ),
       ),
       backgroundColor: Theme.of(context).colorScheme.inversePrimary,
     drawer: MyDrawer(),
 ],
     ),
   );
 }
}
```

Output:


Home Page
Post
Breed
Enter search value
Breed: German shephard
Age: Middle Old
Color: golden
Has Medical Condition: Yes
User Email:
2021.niyati.gaonkar@ves.ac.in
Breed: Lab
Age: Old
Color: White
Has Medical Condition: Yes
User Email: niyatig@gmail.com
Breed: Poodle
Age: New Born
Color: Black and white
Has Medical Condition: No
User Email:
2021.niyati.gaonkar@ves.ac.in
Breed: Labrador
Age: Middle Old
Color: Golden
Has Medical Condition: Yes
User Email:
2021.niyati.gaonkar@ves.ac.in


Post
HOME
PROFILE
USERS
LOGOUT


Users
Useremail: 2021.niyati.gaonkar@ves.ac.in
Useremail: niyatig@gmail.com

EXPLANATION :
Step 1: Create a my_drawer.dart file and may a function in it and call it in the home page.
Step 2: In the drawer, make home,profile,users, and logout button.
Step 3: In the gesture controls of these buttons, use ontap function and useNavigator.push() to navigate to the respective pages.

CONCLUSION: Thus we have created a drawer to navigate to the different pages of our app.We have used the gesture ontap to click the button and Navigator.push() to navigate to a different page. Thus we have understood navigation and routing in flutter.