Experiment no. 5
Name: Pranav Chindage
Div: D15B
Roll no.: 10

Aim: To apply navigation, routing and gestures in Flutter
App Theory:
Navigation and Routing 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. Navigation: Navigation in Flutter refers to the ability to move between different screens or pages within an app. Flutter provides a Navigator widget to manage the navigation stack and perform common navigation operations. Navigation Operations:

Pushing a Screen: Use Navigator.push to navigate to a new screen.
Example:
Navigator.push(context, MaterialPageRoute(builder: (context) => DetailsScreen()));

Popping a Screen: Use Navigator.pop to go back to the previous screen.
Example:
Navigator.pop(context);

Routing: Routing:
In the context of Flutter, involves defining the paths or routes that lead to different screens in your app. It allows you to organize and structure the flow of your application. Flutter supports both named routes and unnamed (or default) routes.

Named Routes:
Named routes are routes identified by a unique string identifier. They provide a more organized and maintainable way to navigate between screens. You can define named routes in the MaterialApp widget using the routes property.

Example:
MaterialApp(routes: { '/': (context) => HomeScreen(), '/details': (context) => DetailsScreen(), },)

```
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:sports_tracker/firebase_funcs.dart';
import 'package:sports_tracker/firebase_options.dart';
import 'package:sports_tracker/home_main.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';

import 'package:sports_tracker/home_tab.dart';
import 'package:sports_tracker/matches_tab.dart';
import 'package:sports_tracker/points_table.dart';
import 'package:sports_tracker/profile_tab.dart';

Future <void> main() async {


 WidgetsFlutterBinding.ensureInitialized();

 await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform,);
 fetch_data();
 runApp(MaterialApp(
   home: Home(),
   theme: ThemeData(
   fontFamily: 'Teko',
   ),
   initialRoute: '/',
   routes: {
     '/home_tab': (context) => Home_tab(),
     '/matches_tab': (context) => Matches_tab(),
     '/profile_tab': (context) => Profile_tab(),
```

```
    '/home_tab_points_table': (context) => Points_table(),
  },
 ));

}
```

```
import 'package:firebase_database/firebase_database.dart';
import 'package:flutter/material.dart';
import 'package:sports_tracker/firebase_funcs.dart';
import 'package:sports_tracker/points_table.dart';
import 'package:sports_tracker/squad.dart';
import 'package:sports_tracker/stats.dart';


class Home_tab extends StatefulWidget {
 const Home_tab({super.key});
 // final dynamic home_tab_squad;
 //
 // const Home_tab({Key? key, required this.home_tab_squad}) : super(key: key);


 @override
 State<Home_tab> createState() => _Home_tabState();
}

class _Home_tabState extends State<Home_tab> {

 var entire_squad;

 Future<Object?> getsquad() async {
```

```
  FirebaseDatabase database = FirebaseDatabase.instance;
  final ref = FirebaseDatabase.instance.ref();

  try {
    final snapshot = await ref.child('Squad').get();
    var sd = snapshot.value;
    return sd; // Return the fetched data
  } catch (error) {
    // Handle error if needed
    return null;
  }
}


int selected_index=0;
void change (int index) {
setState(() {
    selected_index = index;
  }
  );
}


@override
Widget build(BuildContext context) {

  return SingleChildScrollView(
    child:Column(
      children: [
        Container(
        child: Row(
          mainAxisAlignment: MainAxisAlignment.spaceAround,


          children: [

            ElevatedButton( // for player stats

              onPressed: () {
                change(0);
```

```
  },
  child: Column(
    mainAxisSize: MainAxisSize.min,
    // Ensure that the column only occupies the space required by its children
    children: [
      Icon(Icons.people, color: Colors.red[700]), // Icon widget
      SizedBox(height: 8), // Spacer between icon and text
      Text('Squad',
        style: TextStyle(
          color: Colors.black,
        ),),
      // Text widget
    ],
  ), // Specify the label
),

ElevatedButton( //for squad stats
  onPressed: () {
    change(1);
  },
  child: Column(
    mainAxisSize: MainAxisSize.min,
    // Ensure that the column only occupies the space required by its children
    children: [
      Icon(Icons.person, color: Colors.red[700]), // Icon widget
      SizedBox(height: 8), // Spacer between icon and text
      Text('Stats',
          style: TextStyle(
            color: Colors.black,
          )), // Text widget
    ],
  ), // Specify the label
),

ElevatedButton( //for points table
  onPressed: () {
    //change(2);
    Navigator.pushNamed(context, '/home_tab_points_table');
  },
  child: Column(
```

```
            mainAxisSize: MainAxisSize.min,
            // Ensure that the column only occupies the space required by its children
            children: [
              Icon(Icons.table_rows_outlined, color: Colors.red[700]),
              // Icon widget
              SizedBox(height: 8),
              // Spacer between icon and text
              Text('Points table',
                  style: TextStyle(
                  color: Colors.black,
                  )),
              // Text widget
            ],
          ), // Specify the label
        ),
      ],
    ),
  ),
  SizedBox(height: 30),
  Container(
    // children: [
    // FutureBuilder<Object?>(
    //       future: getsquad(), // Call getsquad() to fetch data
    //       builder: (BuildContext context, AsyncSnapshot<Object?> snapshot) {
    //         // Check if the Future has completed
    //         if (snapshot.connectionState == ConnectionState.done) {
    //           // Check if data has been successfully fetched
    //           if (snapshot.hasData) {
    //             // Data has been fetched successfully, use it
    //             entire_squad = snapshot.data;
    //             List<Widget> textWidgets = [];
    //
    //             for (var i in entire_squad) {
    //               textWidgets.add(
    //                 ExpansionTile(
    //                   title: Row(
    //                     children: [
    //                       Icon(Icons.sports_soccer),
    //                       SizedBox(width: 10), // Add some spacing between the icon
and the text
```

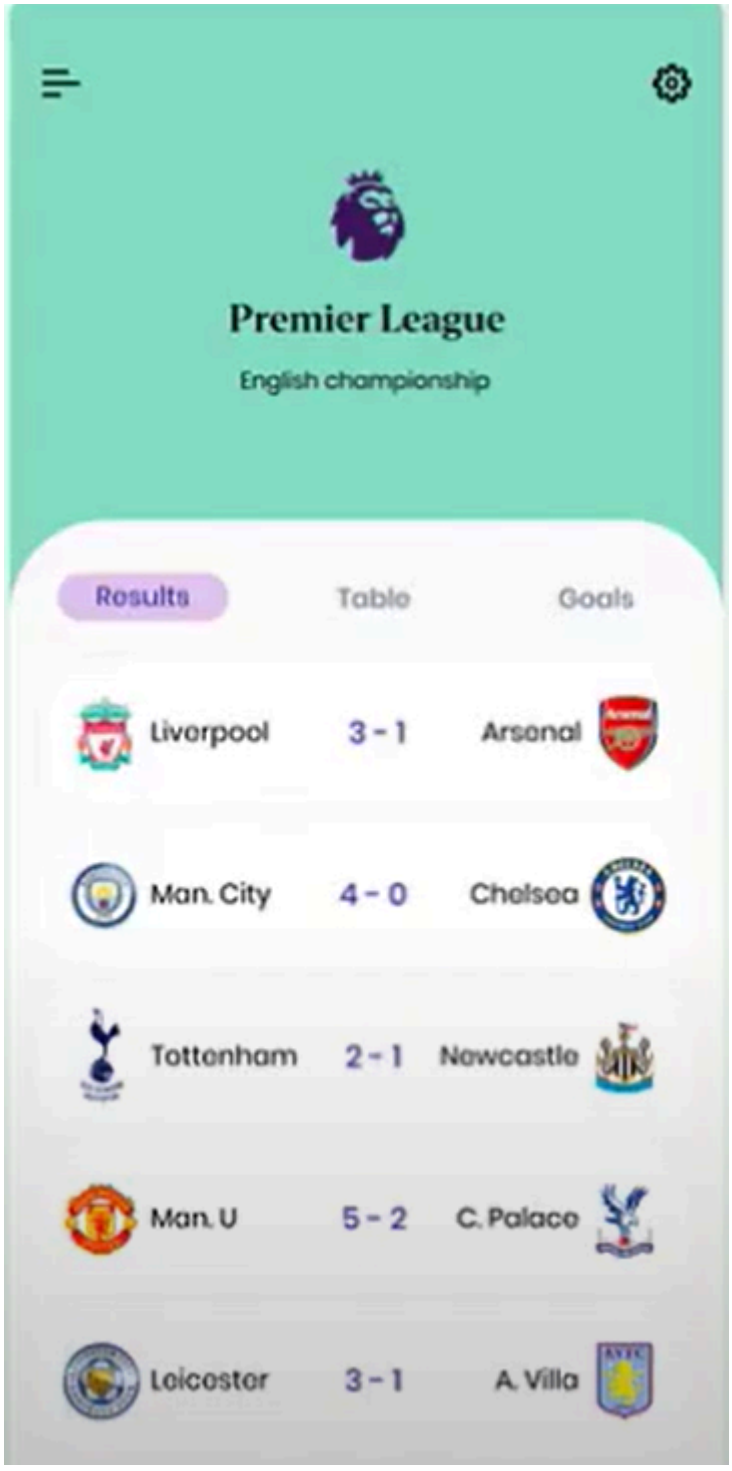
```
        //                   Text('${i['name']}'),
        //                 ],
        //               ),
        //               children: <Widget>[
        //                 // Content widgets inside the ExpansionTile
        //                 ListTile(
        //                   title: Text('Name: ${i['name']}\n\n'
        //                       'Position: ${i['position']}\n\n'
        //                       'Nationality: ${i['nationality']}'),
        //                 ),],
        //             ),
        //           );
        //         }
        //
        //         return Column(
        //           children: textWidgets,
        //         );
        //       } else {
        //         // Data fetch failed or is null, handle error or show loading indicator
        //         return Center(child: CircularProgressIndicator()); // Or display an error
message
        //       }
        //     } else {
        //       // Future is still loading, show a loading indicator
        //       return Center(child: CircularProgressIndicator());
        //     }
        //   },
        // ),
        // ]
        child: _getBodyWidget(selected_index),
      ),




      ]
      ),
      );
```

```
  }

  Widget _getBodyWidget(int index) {
    switch (index) {
      case 0:
        return Squad_details();
      case 1:
        return Stats();
      case 2:
        return Points_table();
      default:
        return SizedBox.shrink();
    }
  }
}
```


Premier League
English championship
Results
Table
Goals
Liverpool
3 - 1
Arsenal
Man. City
4 - 0
Chelsea
Tottenham
2 - 1
Newcastle
Man. U
5 - 2
C. Palace
Leicester
3 - 1
A. Villa

Conclusion: Hence, we have successfully implemented the required functionalities for our app.