Experiment-5

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

Theory :

Navigation and routing are fundamental concepts in mobile applications, enabling users to transition between different pages or screens. In Flutter, screens and pages are referred to as routes, each being represented by a widget.

In Flutter, routes are akin to Activities in Android and ViewControllers in iOS. They facilitate the movement between pages, defining the workflow of the application.
Navigating between pages is handled through routing, with Flutter providing tools like MaterialPageRoute, Navigator.push(), and Navigator.pop() for basic 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.

Navigation:
Navigation in Flutter involves moving between different screens or pages within the app. Flutter offers the Navigator widget to manage the navigation stack and execute common navigation tasks.

Routing:
Routing in Flutter entails defining paths or routes leading to different screens, enabling the organization and structuring of the application flow. Flutter supports both named and unnamed (default) routes.

Named Routes:

Named routes are identified by unique string identifiers, offering a more structured and maintainable way to navigate between screens. You can define named routes within the MaterialApp widget using the routes property.

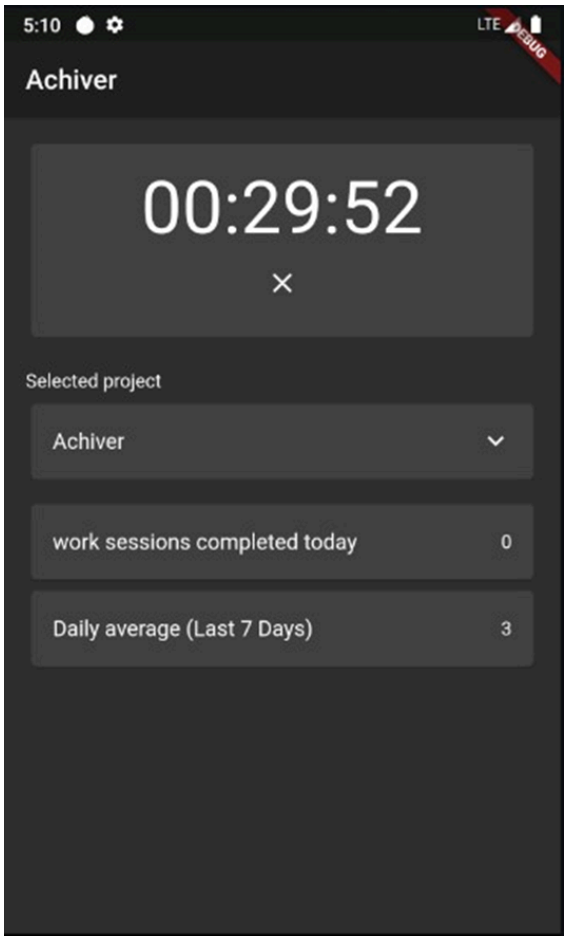
Code :

```
import 'package:flutter/material.dart' ; import
'package:flutter_achiver/core/presentation/notifiers/theme_notifier.dart' ; import
'package:flutter_achiver/features/auth/presentation/notifiers/us
er_repository.dart' ; import
'package:flutter_achiver/features/auth/presentation/pages/main_s creen.dart' ;
import
'package:flutter_achiver/features/projects/data/services/firestore_project_service
.dart' ; import
'package:flutter_achiver/features/projects/presentation/pages/add_project.dart' ;
import
'package:flutter_achiver/features/projects/presentation/pages/pr
oject_details.dart' ; import
'package:flutter_achiver/features/projects/presentation/pages/pr ojects.dart' ;
import
'package:flutter_achiver/features/stat/data/service/firestore_log_service.dart' ;
import
'package:flutter_achiver/features/stat/presentation/pages/add_work_log.dart' ;
import
'package:flutter_achiver/features/stat/presentation/pages/log_list.dart' ; import
'package:flutter_achiver/features/timer/presentation/notifiers/timer_state.dart' ;
import 'package:provider/provider.dart' ; void main() =>
runApp(ProvidedApp());

class ProvidedApp extends StatelessWidget {
  @override
    Widget build(BuildContext context) { return
      MultiProvider(
       providers: [
            ChangeNotifierProvider( builder: (_) =>
              ThemeNotifier(),
```

```
            ),
              ChangeNotifierProvider( builder: (_) =>
                UserRepository.instance(),
            ),
              ChangeNotifierProxyProvider<UserRepository, TimerState>(
                builder: (context, user, timerState) { if (timerState != null ) {
                    timerState.setUser = user.fsUser; return
                   timerState;
                } else {
                        return TimerState(user: user.fsUser);
                }
              },
            ),
          ],
          child: MyApp(),
        );
    }
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    ThemeData theme =
    Provider.of<ThemeNotifier>(context).currentTheme; return
        Consumer<UserRepository>(builder: (context, user,
      child) { if (user.user != null )
              {
       initWorkLogDBS(user.user.uid);
      initProjectDBS(user.user.uid);
      }
      return MultiProvider(
        providers: [
            StreamProvider.value( value:
              projectDBS?.streamList(),
          ),
        ],
          child: MaterialApp(
            title: 'Achiver' ,
            theme: theme, home:
```

```
                MainScreen(),
                routes: {
                      "projects" : (_) => ProjectsPage(),
                 "add_project" : (_) => AddProjectPage(),
                 "add_work_log" : (_) => AddWorkLogPage(),
               },
               onGenerateRoute: (RouteSettings settings) {
                     return  MaterialPageRoute(builder: (_){  switch
                       (settings.name)  {
                        case  "edit_project" :
                              return  AddProjectPage(
                                project: settings.arguments,
                              );  case  "project_details" :
                                 return  ProjectDetailsPage(
                                 project: settings.arguments,
                               );
                              case  "log_list" :
                               return  LogListPage(
                                         logs: settings.arguments,
                               );  default :
                               return  MainScreen();
                           }
                       });
               },
            ),
         );
    });
}
```

```
}
```



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