Yogeshchandra Paranjpe
D15B-52

Experiment No: 05

Aim: To apply navigation, routing and gestures in Flutter Application.

THEORY :

Flutter's navigation, routing, and gesture systems are critical components of building smooth, intuitive, and interactive mobile applications.

1. Navigation:

Navigation in Flutter refers to the process of moving between different screens or pages within an application. Flutter offers a `Navigator` class that manages a stack of `Route` objects. Each route represents a screen or page in the application. The `Navigator` facilitates pushing new routes onto the stack, popping existing routes off the stack, and managing transitions between routes.

2. Routing:

Routing in Flutter involves defining the routes available within an application and mapping them to specific widgets. Flutter uses a routing table to define the mapping between route names and corresponding widget builders. Developers can define routes using named routes or on-the-fly route creation using anonymous routes.

Here's a basic example of route definition in Flutter:

```
dart
MaterialApp(
  routes: {
    '/home': (context) => HomePage(),
    '/profile': (context) => ProfilePage(),
  },
  initialRoute: '/home',
)
```

3. Gestures:

Gestures in Flutter enable touch-based interaction with widgets. Flutter provides a rich set of gesture recognizers that detect various touch events such as taps, drags, swipes, pinches, etc. These recognizers allow developers to create custom touch-based interactions within their applications. Some commonly used gesture recognizers in Flutter include `GestureDetector`, `InkWell`, `GestureDetector`, and `Draggable`.

Here's a simple example of using GestureDetector to detect taps:

```
dart
  GestureDetector(
   onTap: () {
    print('Widget tapped');
   },
   child: Container(
    width: 200,
    height: 200,
    color: Colors.blue,
    child: Center(
     child: Text('Tap me'),
    ),
   ),
  )
```

4. Integration: Flutter's navigation, routing, and gesture systems are often integrated to create seamless user experiences. For example, developers can use gesture recognizers to detect swipes or taps on certain widgets and trigger navigation events accordingly. Similarly, navigation events can be used to push or pop routes within the application. Here's an example of using a gesture to navigate to a new screen:

```
dart
  GestureDetector(
   onTap: () {
    Navigator.push(
     context,
```

```
      MaterialPageRoute(builder: (context) => SecondScreen()),
    );
  },
  child: Text('Go to second screen'),
)
```

Input:

a. **Navigation and Routing**

```
import 'package:flutter/material.dart';
import
'package:news_app/src/features/authentication/presentation/
account/screens/changename_screen.dart';
import
'package:news_app/src/features/authentication/presentation/
account/screens/changepasword_screen.dart';
import
'package:news_app/src/features/authentication/presentation/
account/screens/email_screen.dart';
import
'package:news_app/src/features/authentication/presentation/
account/screens/profile_screen.dart';
import
'package:news_app/src/features/authentication/presentation/
register/screens/forget_password_screen.dart';
import
'package:news_app/src/features/authentication/presentation/
register/screens/login_screen.dart';
import
'package:news_app/src/features/authentication/presentation/
register/screens/signup_screen.dart';
import
'package:news_app/src/features/daily_news/presentation/arti
cle_detail/article_screen.dart';
```

```
import
'package:news_app/src/features/daily_news/presentation/disc
over/screens/discover_screen.dart';
import
'package:news_app/src/features/daily_news/presentation/home
/screens/home_screen.dart';
import
'package:news_app/src/features/daily_news/presentation/save
d_article/saved_article.dart';

class AppRoutes {
  static Map<String, Widget Function(BuildContext)> routes
= {
    LoginScreen.routeName: (context) => const
LoginScreen(),
    SignUpScreen.routeName: (context) => const
SignUpScreen(),
    ForgetPasswordScreen.routeName: (context) => const
ForgetPasswordScreen(),
    HomeScreen.routeName: (context) => const HomeScreen(),
    DiscoverScreen.routeName: (context) => const
DiscoverScreen(),
    ProfileScreen.routeName: (context) => const
ProfileScreen(),
    ChangeNameScreen.routeName: (context) => const
ChangeNameScreen(),
    EmailScreen.routeName: (context) => const
EmailScreen(),
    ChangePasswordScreen.routeName: (context) => const
ChangePasswordScreen(),
    ArticleScreen.routeName: (context) => const
ArticleScreen(),
    SavedArticles.routeName: (context) => const
SavedArticles(),
```

```
  };
}
```

b. Gesture Control and Routing

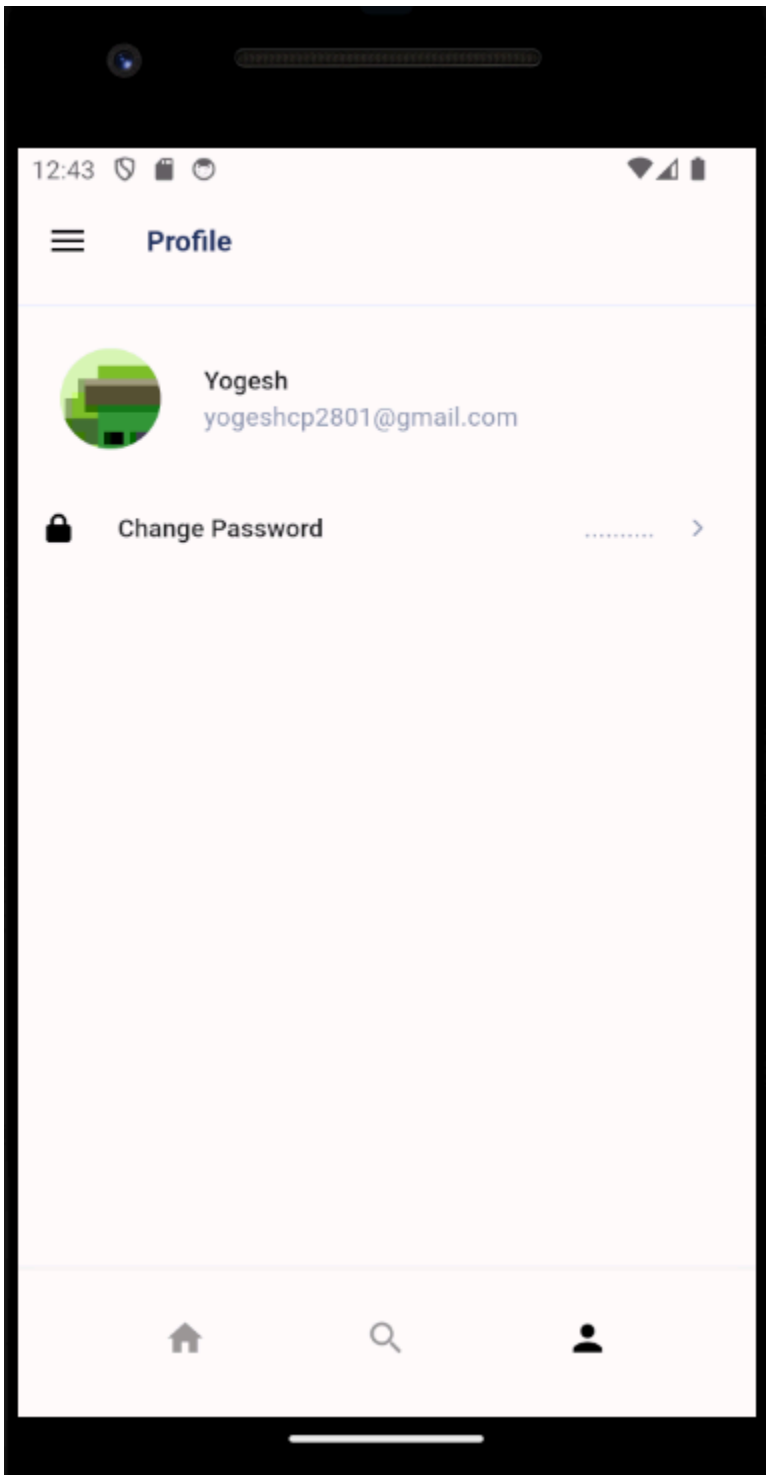
```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import
'package:news_app/src/core/components/bottom_nav_bar.dart';
import 'package:news_app/src/core/components/drawer.dart';
import
'package:news_app/src/core/components/shimmer/shimmer_scree
n_widget.dart';
import
'package:news_app/src/features/daily_news/domain/enums/news
_category_enum.dart';
import 'package:news_app/src/injection_container.dart';

import '../bloc/discover_bloc.dart';
import '../bloc/discover_event.dart';
import '../bloc/discover_state.dart';
import '../components/category_news.dart';
import '../components/discover_news.dart';

class DiscoverScreen extends StatelessWidget {
  const DiscoverScreen({super.key});

  static const routeName = '/discover';

  @override
  Widget build(BuildContext context) {
    return BlocProvider<DiscoverBloc>(
```

```
      create: (context) =>
sl()..add(GetArticlesDiscoverEvent(initialCategory:
NewsCategoryEnum.health.category)),
      child: const _DiscoverScreen(),
    );
  }
}

class _DiscoverScreen extends StatelessWidget {
  const _DiscoverScreen({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    List<NewsCategoryEnum> tabs = [
      NewsCategoryEnum.health,
      NewsCategoryEnum.business,
      NewsCategoryEnum.entertainment,
      NewsCategoryEnum.science,
      NewsCategoryEnum.sports,
      NewsCategoryEnum.technology,
      NewsCategoryEnum.general,
    ];

    return DefaultTabController(
      initialIndex: 0,
      length: tabs.length,
      child: Scaffold(
        appBar: AppBar(
          backgroundColor: Colors.transparent,
          elevation: 0,
          leading: Builder(
            builder: (context) => IconButton(
              onPressed: () =>
Scaffold.of(context).openDrawer(),
```

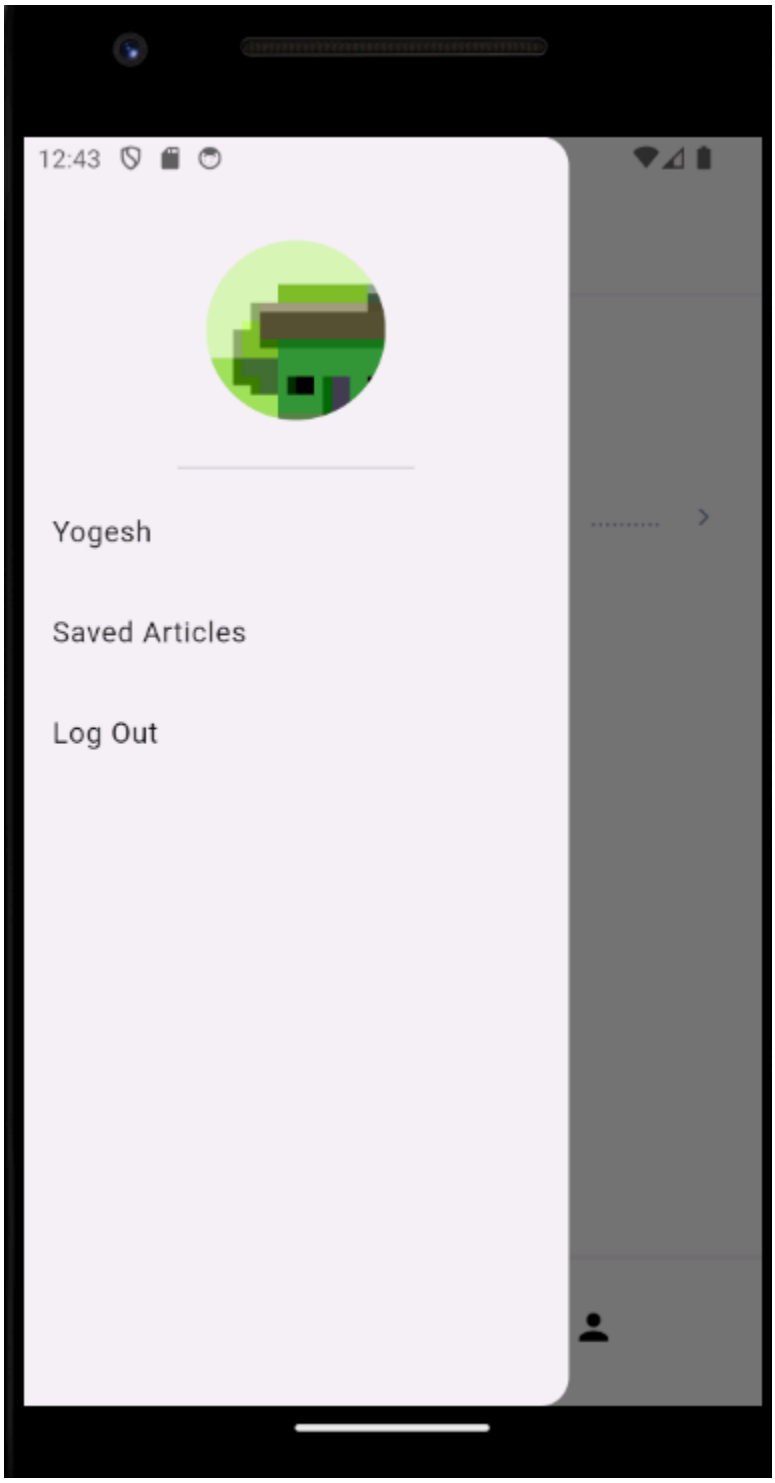
```
              icon: const Icon(
                Icons.menu,
                color: Colors.black,
              ),
            ),
          ),
        ),
        drawer: const AppDrawer(),
        bottomNavigationBar: const BottomNavBar(index: 1),
        body: BlocBuilder<DiscoverBloc, DiscoverState>(
          builder: (_, state) {
            return ListView(
              padding: const EdgeInsets.all(20.0),
              children: [
                const DiscoverNews(),
                AnimatedSwitcher(
                  duration: const Duration(milliseconds:
350),
                  child: (state is LoadingDiscoverState)
                      ? ShimmerScreen(
                          enabled: true,
                          child: CategoryNews(tabs: tabs),
                        )
                      : CategoryNews(tabs: tabs),
                )
              ],
            );
          },
        ),
      ),
    );
  }
}
```

Yogeshchandra Paranjpe
D15B-52
MPL LAB EXP 5

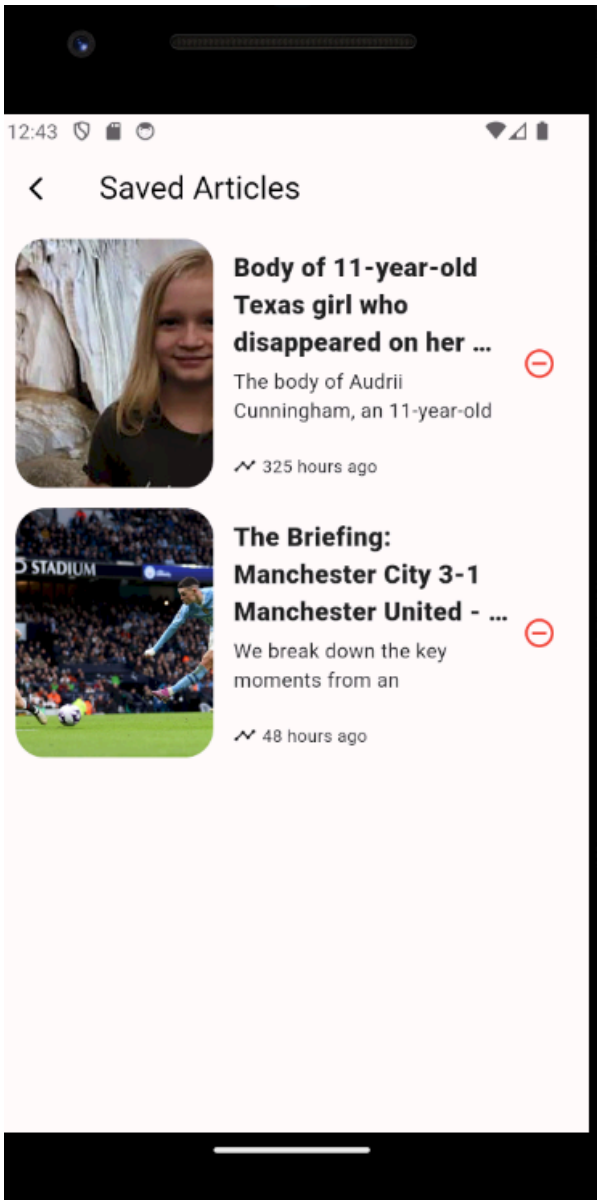
Output:



Yogeshchandra Paranjpe
D15B-52
MPL LAB EXP 5



Yogeshchandra Paranjpe
D15B-52
MPL LAB EXP 5



Conclusions: Through this experiment, we have acquired valuable insights into Flutter's routing, navigation, and gesture implementation. Our exploration involved mastering the art of fluidly moving between screens using named routes, elevating the overall user experience. This practical experience has enriched my comprehension of Flutter's versatile functionalities in creating dynamic and captivating mobile applications.