Experiment No: 4

Experiment No 5 5: Exploring Flutter Widgets.	
ROLL NO	**03**
NAME	**Ansari Mohammed Sharjeel**
CLASS	**D15B**
SUBJECT	**MAD & PWA Lab**
LO-MAPPED	

Experiment 5

Aim: To demonstrate navigation, routing, and gestures in a Flutter application.

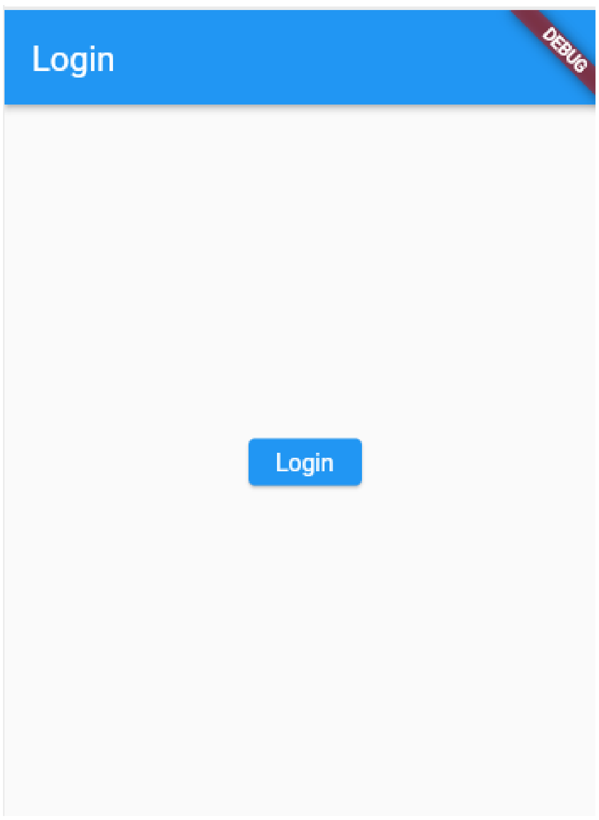
Theory:

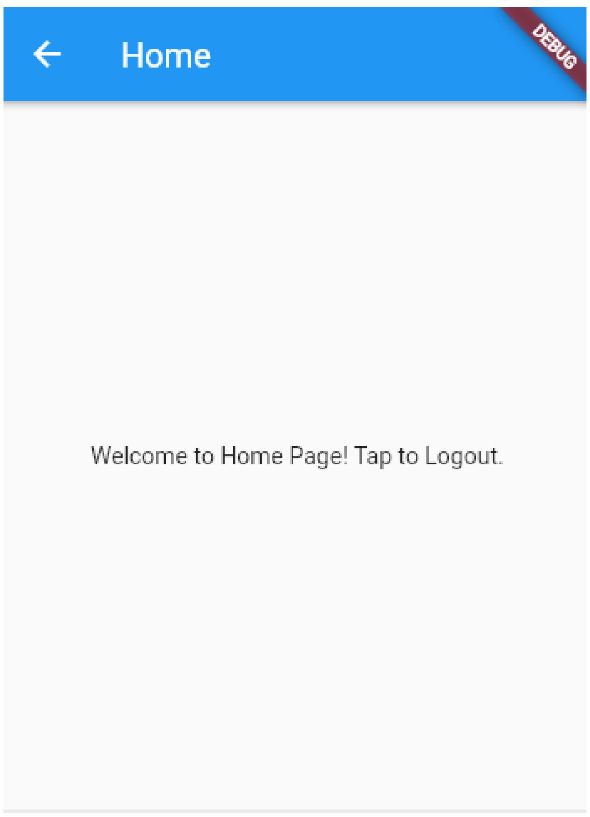
- Navigation is the action of moving between different screens or pages within a mobile or web application. It enables users to explore various sections of the app, access different functionalities, and engage with content seamlessly. In Flutter, navigation is typically managed using the Navigator widget, which maintains a stack of pages. Each page is represented by a widget, and you can push new pages onto the stack to move forward and pop them off to go backward. The Navigator widget handles transitions between pages, including animations and route management.
- Routing involves defining and managing the paths (or routes) that users can follow through an application. In Flutter, routing requires specifying a set of named routes and linking them to specific page widgets. This enables navigation to different pages by referencing their route names. Flutter facilitates routing through the MaterialApp widget's routes parameter. Here, you can establish a map of route names to builder functions that produce the corresponding page widgets.
- Gestures pertain to touch-based interactions with the user interface elements of an application. In Flutter, gestures are managed using gesture recognizer widgets such as GestureDetector. These widgets identify various types of user input, such as taps, drags, swipes, and pinches, and allow you to respond to them with tailored actions or behaviors.

Code:

```dart
// Importing the necessary packages
import 'package:flutter/material.dart';

// Main function to run the Flutter app
void main() {
  runApp(MyApp());
}

// MyApp class, defining the root widget of the application
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Navigation Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      // Initial route set to the Login page
      initialRoute: '/login',
      // Route definitions
      routes: {
        '/login': (context) => LoginPage(),
        '/home': (context) => HomePage(),
      },
    );
  }
}

// LoginPage class, representing the login screen
class LoginPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Login'),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            // Navigate to the Home page
            Navigator.pushNamed(context, '/home');
          },
          child: Text('Login'),
        ),
      ),
    );
  }
}

// HomePage class, representing the home screen
class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Home'),
      ),
      body: GestureDetector(
        // Gesture detector for detecting taps
        onTap: () {
          // Pop the current route (Home page) from the stack to go back to the
previous page (Login page)
          Navigator.pop(context);
        },
        child: Center(
          child: Text('Welcome to Home Page! Tap to Logout.'),
        ),
      ),
    );
  }
}
```

<u>Output:</u>





Welcome to Home Page! Tap to Logout.

<u>Conclusion</u>:
In this experiment, we demonstrated how to implement navigation, routing, and gestures in a Flutter application. Navigation was achieved using the Navigator widget, routing was defined through named routes in the MaterialApp widget, and gestures were managed using the GestureDetector widget. These features are essential for creating a smooth and intuitive user experience in Flutter applications.