Anket Kadam
D15B/26

Experiment No 5

Title: Implementing Navigation, Routing, and Gestures in a Flutter App

Abstract:

In the realm of Flutter app development, integrating navigation, routing, and gestures is essential for crafting intuitive user experiences. Navigation encompasses the movement between different screens or "routes" within an app, facilitated by the Navigator widget. Routing, on the other hand, focuses on defining and managing named routes for screens, enhancing code organization and readability. Meanwhile, gestures empower users to interact with the app through touch-based actions. This experiment explores the implementation of these concepts in a Flutter application, aiming to enhance user engagement and navigation fluidity.

Introduction:

Flutter, with its robust framework, offers developers versatile tools for building engaging mobile applications. Navigation, routing, and gestures play pivotal roles in shaping the user experience. Navigating seamlessly between screens, organizing routes efficiently, and incorporating intuitive touch interactions are key aspects of a well-designed Flutter app.

Objective:

The objective of this experiment is to implement navigation, routing, and gestures effectively within a Flutter application. By understanding the principles and techniques involved, developers can create apps that are both user-friendly and aesthetically pleasing.

Methods:

Defining Routes: Begin by establishing a mapping between route names and corresponding widget classes. This ensures clarity and structure in the app's navigation flow.

Navigating Between Screens:

Anket Kadam
D15B/26

Utilize navigation methods like Navigator.push() to transition between screens based on user interactions. By pushing new routes onto the stack or popping existing routes, users can navigate through the app seamlessly.

Passing Data Between Screens: Harness route parameters to transmit data between screens, enabling dynamic content updates and personalized user experiences.

Handling Navigation Events: Implement logic to manage navigation events effectively. This may involve validating inputs, displaying loading indicators, or executing other tasks before transitioning to a new screen.

Managing Navigation Back: Consider user behavior when navigating back to previous screens. Ensure that the app responds correctly to system back buttons or gestures, maintaining coherence in the navigation flow.

CODE:

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

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Navigation and Routing Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      initialRoute: '/',
      routes: {
        '/': (context) => HomeScreen(),
        '/details': (context) => DetailsScreen(),
      },
    );
  }
```

Anket Kadam
D15B/26

```
}

class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Home'),
      ),
      body: GestureDetector(
        onHorizontalDragUpdate: (details) {
          if (details.delta.dx > 10) {
            Navigator.pushNamed(context, '/details');
          }
        },
        child: Container(
          color: Colors.blue[50],
          child: Center(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: <Widget>[
                Icon(
                  Icons.home,
                  size: 100,
                  color: Colors.blue,
                ),
                SizedBox(height: 20),
                Text(
                  'Welcome to the Home Screen!',
                  style: TextStyle(fontSize: 24, fontWeight:
FontWeight.bold),
                ),
                SizedBox(height: 40),
                ElevatedButton.icon(
                  onPressed: () {
                    Navigator.pushNamed(context, '/details');
                  },
                  icon: Icon(Icons.arrow_forward),
```

Anket Kadam
D15B/26

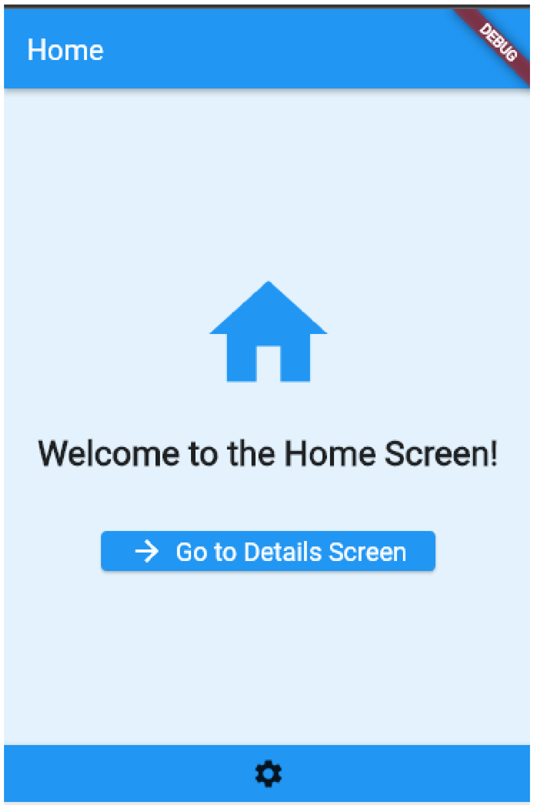
```
                  label: Text('Go to Details Screen'),
                  style: ElevatedButton.styleFrom(
                    padding: EdgeInsets.symmetric(horizontal: 20,
vertical: 10),
                    textStyle: TextStyle(fontSize: 18),
                  ),
                ),
              ],
            ),
          ),
        ),
      ),
      bottomNavigationBar: BottomAppBar(
        color: Colors.blue,
        child: Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            IconButton(
              icon: Icon(Icons.settings),
              onPressed: () {
                // Add your settings navigation logic here
              },
            ),
          ],
        ),
      ),
    );
  }
}

class DetailsScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Details'),
      ),
      body: GestureDetector(
```

Anket Kadam
D15B/26

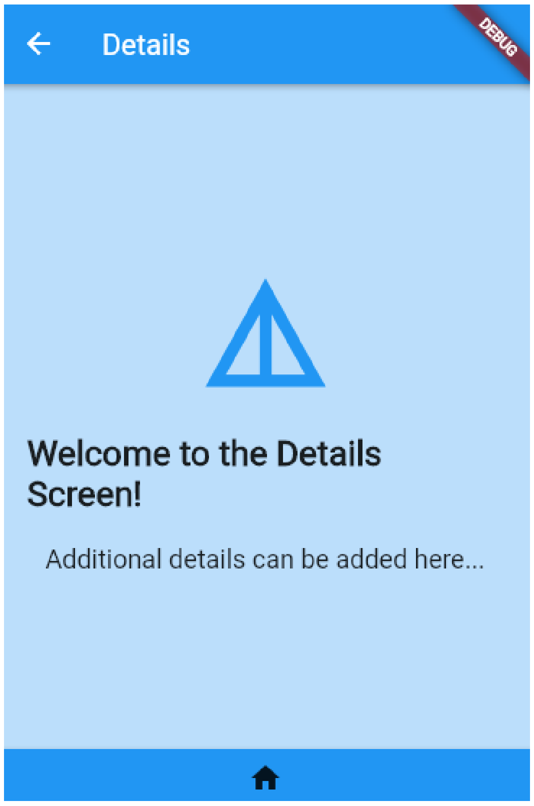
```
        onHorizontalDragUpdate: (details) {
          if (details.delta.dx < -10) {
            Navigator.pop(context);
          }
        },
        child: Container(
          color: Colors.blue[100],
          child: Padding(
            padding: EdgeInsets.all(20),
            child: Center(
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: <Widget>[
                  Icon(
                    Icons.details,
                    size: 100,
                    color: Colors.blue,
                  ),
                  SizedBox(height: 20),
                  Text(
                    'Welcome to the Details Screen!',
                    style: TextStyle(fontSize: 24, fontWeight:
FontWeight.bold),
                  ),
                  SizedBox(height: 20),
                  Text(
                    'Additional details can be added here...',
                    style: TextStyle(fontSize: 18),
                  ),
                ],
              ),
            ),
          ),
        ),
      ),
      bottomNavigationBar: BottomAppBar(
        color: Colors.blue,
        child: Row(
```

Anket Kadam
D15B/26

```
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            IconButton(
              icon: Icon(Icons.home),
              onPressed: () {
                Navigator.popUntil(context,
ModalRoute.withName('/'));
              },
            ),
          ],
        ),
      ),
    );
  }
}
```


Home

Welcome to the Home Screen!


Details

Welcome to the Details Screen!

Additional details can be added here...

Anket Kadam
D15B/26

Conclusion:
Navigation, routing, and gestures in Flutter are crucial for creating engaging user experiences. By mastering these elements, developers can craft standout apps that captivate users and stand out in today's competitive market.