Experiment No : 05

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

Theory:

Navigation, routing, and gestures are essential concepts in Flutter app development, contributing to user interaction, app structure, and overall user experience. Here's a theoretical overview of each:

1. Navigation:

Navigation refers to moving between different screens or pages within a Flutter app. It allows users to explore different parts of the app's content and functionality. Navigation can be triggered by various user actions, such as tapping buttons, swiping, or using gestures.

In Flutter, navigation is typically managed by the Navigator class, which maintains a stack of routes. Each route represents a screen or page in the app. Common navigation patterns include:

Stack-based Navigation: New routes are pushed onto a stack when navigating to a new screen, and popped off the stack when navigating back.

Named Routes: Routes are assigned unique names, allowing developers to navigate to specific screens using their names.

Modal Routes: Routes that appear as overlays on top of the current screen, often used for dialogs, pop-up menus, or settings screens.

2. Routing:

Routing in Flutter refers to the process of defining and managing the app's navigation paths or routes. Routes specify the mapping between route names and corresponding screen widgets. Flutter provides the MaterialApp widget as the root of the app's widget tree, allowing developers to define routes using the routes parameter or by passing a Navigator widget directly.

Key concepts in Flutter routing include:

Initial Route: The route that the app displays when it starts.

Named Routes: Routes identified by unique names that can be navigated to directly using Navigator.pushNamed().

Route Transitions: Animations that define how routes transition on and off the screen, such as sliding, fading, or scaling.

3. Gestures:

Gestures enable users to interact with the app through touch-based inputs, such as tapping, swiping, dragging, pinching, and more. Flutter provides the GestureDetector widget, which listens for various gestures and invokes callback functions when those gestures occur.

Common gestures in Flutter include:

Tap: A quick touch on the screen with a finger.

Double Tap: Two quick taps in succession.

Long Press: Holding a finger on the screen for an extended period.

Swipe: Dragging a finger across the screen in a specific direction.

Pinch: Using two fingers to zoom in or out by bringing them closer together or farther apart.

Code:

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Food App',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: HomePage(),
    );
  }
```

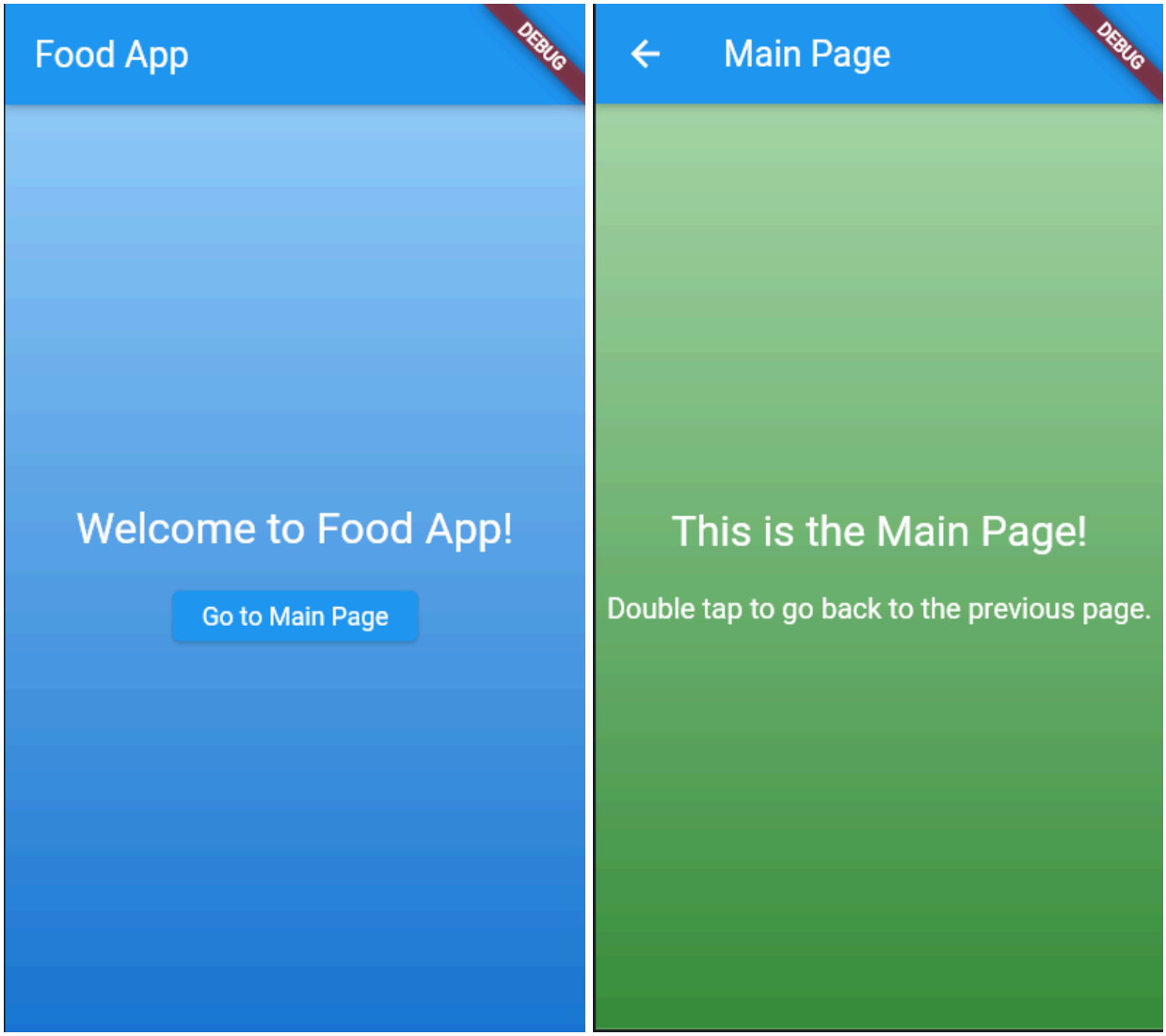
```
}

class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Food App'),
      ),
      body: Container(
        decoration: BoxDecoration(
          gradient: LinearGradient(
            begin: Alignment.topCenter,
            end: Alignment.bottomCenter,
            colors: [Colors.blue[200]!, Colors.blue[700]!],
          ),
        ),
        child: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              Text(
                'Welcome to Food App!',
                style: TextStyle(fontSize: 24, color: Colors.white),
              ),
              SizedBox(height: 20),
              ElevatedButton(
                onPressed: () {
                  // Navigate to the main page of the food app
                  Navigator.push(
                    context,
                    MaterialPageRoute(builder: (context) => MainPage()),
                  );
                },
                child: Text('Go to Main Page'),
              ),
            ],
          ),
        ),
      ),
```

```
    );
  }
}

class MainPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Main Page'),
      ),
      body: Container(
        decoration: BoxDecoration(
          gradient: LinearGradient(
            begin: Alignment.topCenter,
            end: Alignment.bottomCenter,
            colors: [Colors.green[200]!, Colors.green[700]!],
          ),
        ),
        child: GestureDetector(
          onDoubleTap: () {
            // Navigate back to the home page of the food app
            Navigator.pop(context);
          },
          child: Center(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: <Widget>[
                Text(
                  'This is the Main Page!',
                  style: TextStyle(fontSize: 24, color: Colors.white),
                ),
                SizedBox(height: 20),
                Text(
                  'Double tap to go back to the previous page.',
                  style: TextStyle(fontSize: 16, color: Colors.white),
                  textAlign: TextAlign.center,
                ),
              ],
            ),
```

```
          ),
        ),
      ),
    );
  }
}
```



Conclusion: Navigation, routing, and gestures are fundamental concepts in Flutter app development, allowing developers to create engaging user experiences and intuitive interfaces. By understanding these concepts and utilizing Flutter's built-in widgets developers can implement effective navigation patterns, manage app routes efficiently, and enhance user interaction through gestures, resulting in more polished and user-friendly Flutter apps.