EXPERIMENT NO. 5

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

Theory:

1. Navigation:
Navigation in Flutter refers to the movement or transition between different screens or pages within an app. It involves going from one UI view to another, allowing users to navigate through different sections or features of the application.

2. Routing:
Routing in Flutter is the mechanism that enables navigation. It defines how the app should move from one screen to another. Flutter uses a routing system to manage the flow of screens and organize the hierarchy of widgets in the app.

- Navigator: The `Navigator` class in Flutter manages a stack of pages or routes. Each page is represented by a widget. You can push a new route onto the stack to navigate to a different screen and pop a route off the stack to go back.

3. Gestures:
Gestures in Flutter refer to user interactions such as taps, swipes, pinches, etc. Flutter provides a rich set of gesture recognizers that allow you to respond to user input in a variety of ways.

-GestureDetector: The `GestureDetector` widget in Flutter is used to recognize various gestures. It wraps around other widgets and can trigger callbacks based on user interactions.

Code:

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

void main() {

  runApp(const MyApp());

}

class MyApp extends StatelessWidget {

  const MyApp({Key? key}) : super(key: key);

  @override

  Widget build(BuildContext context) {
```

```
    return MaterialApp(
      title: 'Todo App',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      home: const HomePage(),
    );
  }
}
class HomePage extends StatefulWidget {
  const HomePage({Key? key}) : super(key: key);
  @override
  _HomePageState createState() => _HomePageState();
}
class _HomePageState extends State<HomePage> {
  final List<String> tasks = [];
  void _addTask(String task) {
    setState(() {
      tasks.add(task);
    });
  }
  void _removeTask(int index) {
    setState(() {
      tasks.removeAt(index);
```

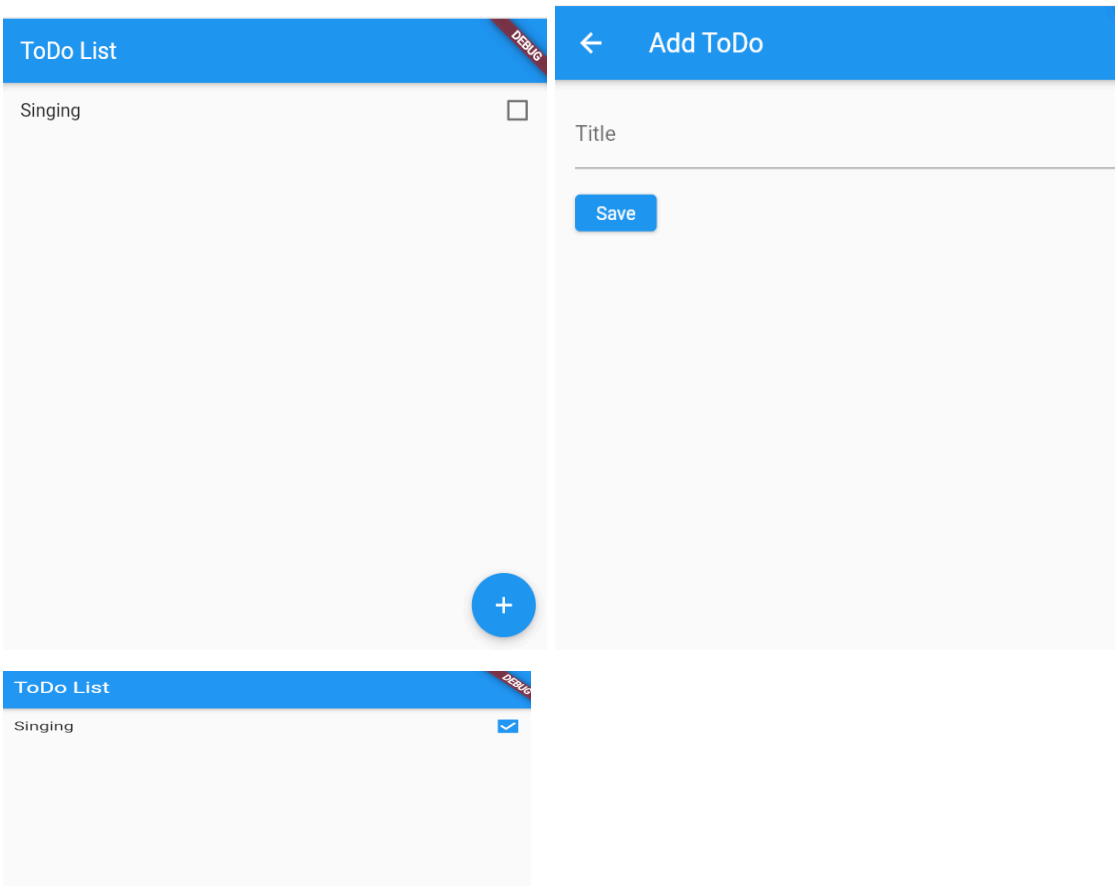
```
  });
}
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Todo List'),
    ),
    body: ListView.builder(
      itemCount: tasks.length,
      itemBuilder: (context, index) {
        return ListTile(
          title: Text(tasks[index]),
          trailing: IconButton(
            icon: Icon(Icons.delete),
            onPressed: () => _removeTask(index),
          ),
        );
      },
    ),
    floatingActionButton: FloatingActionButton(
      onPressed: () {
        Navigator.push(
          context,
          MaterialPageRoute(builder: (context) => AddTaskPage(addTask: _addTask)),
```

```
          );
        },   child: Icon(Icons.add),
      ),
    );
  }}
class AddTaskPage extends StatelessWidget {
  final Function(String) addTask;
  const AddTaskPage({Key? key, required this.addTask}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    String newTask = '';
   return Scaffold(
      appBar: AppBar(
        title: const Text('Add Task'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          children: [
            TextField(
              onChanged: (value) {
                newTask = value;
              },
              decoration: InputDecoration(labelText: 'Task'),
            ),
```

```
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: () {
                if (newTask.isNotEmpty) {
                  addTask(newTask);
                  Navigator.pop(context);
                }
              },
              child: const Text('Add Task'),
            ),
          ],
        ),
      ),
    );
  }
}
```

Output:

Navigation Implementation into the Project



Conclusion:

This example showcases basic navigation between two pages using the `Navigator` and `MaterialPageRoute`. The `GestureDetector` is used to respond to user gestures for navigation. Flutter offers more advanced routing options and a wide range of gestures for creating interactive and engaging user interfaces.