Experiment 5

Name:Sonali Makhijani
Div: D15A
Roll no: 34

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

Theory:

1. Navigation:
    - Navigation refers to the process of moving between different screens or pages within a Flutter app.
    - In Flutter, navigation is typically managed using the Navigator class, which maintains a stack of routes.
    - Each route represents a screen or page in the app, and the navigator manages the navigation stack, allowing users to move forward and backward between routes.
    - Navigation can be triggered by user actions such as tapping buttons, selecting items from lists, or swiping between pages.

2. Routing:
    - Routing is the mechanism used to define and manage the routes within a Flutter app.
    - Routes are defined using route names and associated with corresponding widgets or screens.
    - Flutter provides several routing mechanisms, including named routes, on-the-fly routes, and nested routes.
    - Named routes allow developers to define routes with unique names and navigate to them using the Navigator based on these names.
    - On-the-fly routes are created dynamically at runtime and pushed onto the navigation stack as needed.
    - Nested routes involve embedding navigators within other navigators to create complex navigation structures, such as tab-based navigation or drawer navigation.

3. Gestures:
    - Gestures refer to user interactions such as tapping, dragging, swiping, pinching, and rotating on the screen.
    - Flutter provides a rich set of gesture recognition widgets and APIs to handle user gestures effectively.
    - Common gesture recognition widgets include GestureDetector, InkWell, InkResponse, Draggable, Dismissible, etc.
    - These widgets allow developers to detect various user gestures and trigger corresponding actions or animations in response.

- Gestures can be used to implement interactive UI elements, such as buttons, sliders, swipers, drag-and-drop interfaces, and more.

4. Gesture Detection:

    - Gesture detection in Flutter involves registering gesture recognizers on widgets to detect specific user interactions.
    - Gesture recognizers analyze touch input and determine whether a specific gesture has occurred, such as a tap, double-tap, long-press, drag, etc.
    - Once a gesture is detected, Flutter invokes the corresponding callback function associated with the gesture recognizer.
    - Developers can customize gesture detection by configuring properties such as gesture sensitivity, velocity thresholds, and touch area boundaries.

5. Gesture Handling:
    - After a gesture is detected, developers can handle it by performing various actions, such as updating UI state, navigating between screens, triggering animations, or executing business logic.
    - Gesture handling involves responding to user interactions in a way that provides feedback and enhances the user experience.
    - Flutter's declarative programming model makes it easy to update UI elements in response to user gestures, ensuring a smooth and responsive user interface.

home_screen.dart

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

import '/core/constants/styles.dart';
import '/domain/bloc/expenses_bloc/expenses_bloc.dart';
import '../add_or_edit_expense/add_or_edit_expense.dart';
import '/presentation/screens/search/search_page.dart';
import '/data/repositories/common_interfaces/expenses_repo_interface.dart';
import '/presentation/screens/homescreen/components/expense_tile.dart';
import '/presentation/screens/homescreen/components/category_tile.dart';
import '/presentation/screens/homescreen/components/app_drawer.dart';
import '/presentation/widgets/error_widget.dart';
import '/presentation/widgets/loader.dart';
import '../view_expenses/view_expenses_list.dart';

class HomeScreen extends StatefulWidget {
  const HomeScreen({super.key});

  @override
  State<HomeScreen> createState() => _HomeScreenState();
}
```

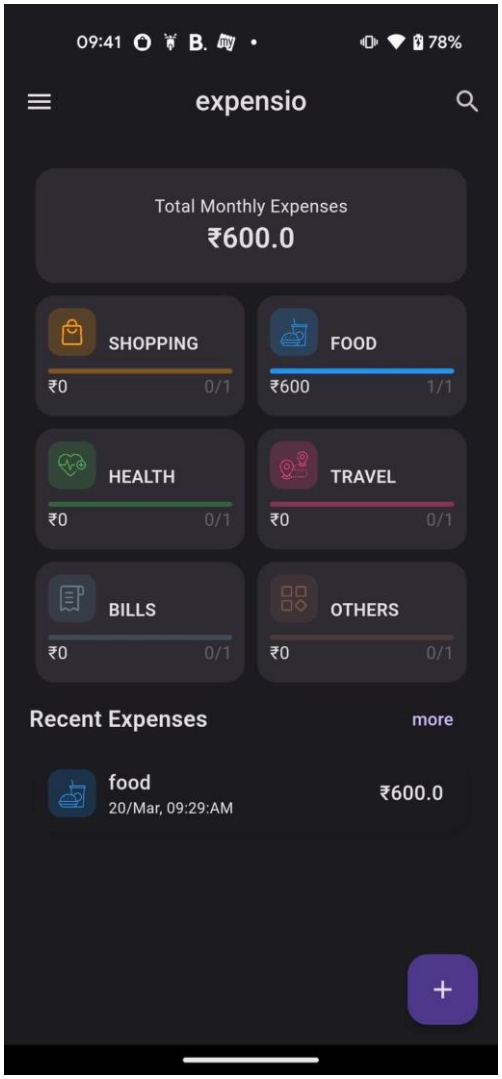
```
class _HomeScreenState extends State<HomeScreen> {
  @override
  void didChangeDependencies() {
    context.read<ExpensesBloc>().add(ExpensesLoadEvent());
    super.didChangeDependencies();
  }

  final GlobalKey<ScaffoldState> _key = GlobalKey();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      key: _key,
      appBar: AppBar(
        title: const Text(
          'expensio',
          style: TextStyle(
            fontWeight: FontWeight.w500,
          ),
        ),
        centerTitle: true,
        leading: IconButton(
          icon: const Icon(Icons.menu),
          onPressed: () => _key.currentState!.openDrawer(),
        ),
        actions: [
          IconButton(
            onPressed: () {
              Navigator.push(context,
                  MaterialPageRoute(builder: (context) => const SearchPage()));
            },
            icon: const Icon(Icons.search_rounded),
          )
        ],
      ),
      drawer: const AppDrawer(),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          Navigator.push(
            context,
```

```
      MaterialPageRoute(
        builder: (_) => const AddOrEditExpensePage(),
        fullscreenDialog: true,
        barrierDismissible: true,
      ),
    );
  },
  child: const Icon(Icons.add),
),
body: BlocBuilder<ExpensesBloc, ExpensesState>(
  builder: (context, state) {
    if (state is ExpensesLoadingState) {
      return const Loader();
    } else if (state is ExpensesErrorState) {
      return CustomErrorWidget(
        exceptionCaught: state.exception,
        onPressed: () {
          context.read<ExpensesBloc>().add(ExpensesLoadEvent());
        },
      );
    } else {
      return ListView(
        padding: const EdgeInsets.all(paddingDefault),
        physics: const BouncingScrollPhysics(),
        children: [
          Container(
            padding: const EdgeInsets.all(paddingDefault),
            margin: const EdgeInsets.all(5),
            decoration: BoxDecoration(
              color: Theme.of(context)
                  .colorScheme
                  .surfaceVariant
                  .withOpacity(0.4),
              borderRadius: borderRadiusDefault,
            ),
            child: Column(
              children: [
                const Text('Total Monthly Expenses'),
                Text(
                  "₹${context.read<ExpensesBloc>().allExpensesSum.toString()}",
                  style: const TextStyle(
                      fontSize: 22, fontWeight: FontWeight.w600),
                )
```

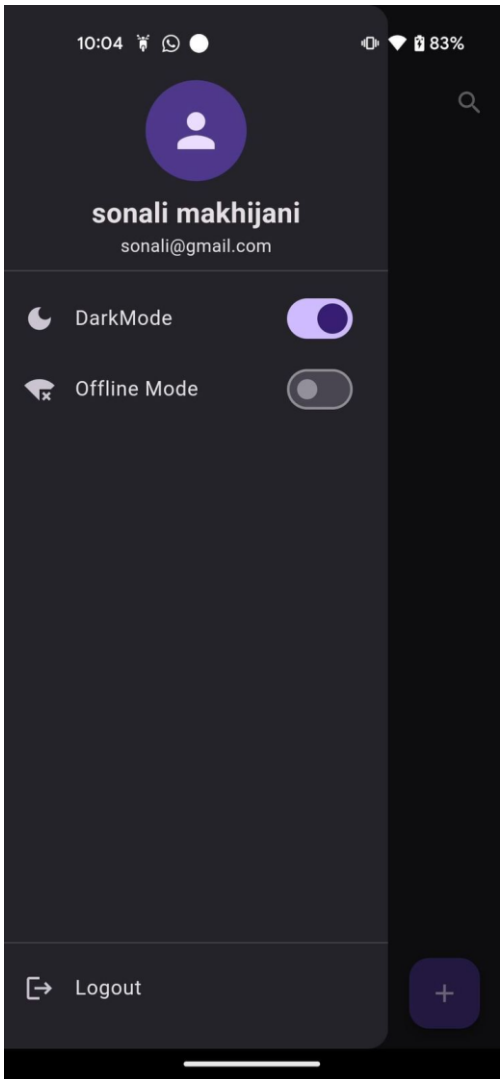
```
    ],
  ),
),
ConstrainedBox(
  constraints:
      const BoxConstraints(maxHeight: 500, maxWidth: 300),
  child: GridView.count(
    physics: const NeverScrollableScrollPhysics(),
    shrinkWrap: true,
    crossAxisCount: 2,
    childAspectRatio: 4 / 2.4,
    children: List.generate(
      ExpenseCategory.values.length,
      (index) => ConstrainedBox(
        constraints:
            const BoxConstraints(maxHeight: 150, maxWidth: 300),
        child: CategoryTile(index),
      ),
    ),
  ),
),
Row(
  mainAxisAlignment: MainAxisAlignment.spaceBetween,
  children: [
    const Text(
      "Recent Expenses",
      style:
          TextStyle(fontWeight: FontWeight.w600, fontSize: 18),
    ),
    TextButton(
      onPressed: () {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => const ViewExpenses(),
          ),
        );
      },
      child: const Text('more'),
    )
  ],
),
BlocBuilder<ExpensesBloc, ExpensesState>(
    builder: (context, state) {
```

```
                return ListView.builder(
                  physics: const NeverScrollableScrollPhysics(),
                  shrinkWrap: true,
                  itemCount:
                      context.watch<ExpensesBloc>().listOfExpenses.length,
                  itemBuilder: (context, index) {
                    final expense =
                        context.watch<ExpensesBloc>().listOfExpenses[index];
                    return ExpenseTile(expense: expense);
                  },
                );
              }),
            ],
          );
        }
      },
    ),
  );
 }
}
```

App UI:



Home Page



Side Bar

Widgets used: Text, Side nav bar, Icons

Conclusion: Therefore understood navigation, routing, gesture detection and gesture handling in Flutter and implemented the same in my Flutter application to route different pages.