Name: Rushikesh Yeole Roll no: 72

MAD Experiment 5

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

Theory :

Flutter Navigation and Routing :
Navigation and routing are some of the core concepts of all mobile application, which allows the user to move between different pages. We know that every mobile application contains several screens for displaying different types of information. For example, an app can have a screen that contains various products. When the user taps on that product, immediately it will display detailed information about that product.
In Flutter, the screens and pages are known as routes, and these routes are just a widget. In Android, a route is similar to an Activity, whereas, in iOS, it is equivalent to a ViewController. In any mobile app, navigating to different pages defines the workflow of the application, and the way to handle the navigation is known as routing. Flutter provides a basic routing class MaterialPageRoute and two methods Navigator.push() and Navigator.pop() that shows how to navigate between two routes. The following steps are required to start navigation in your application.
Step 1: First, you need to create two routes.
Step 2: Then, navigate to one route from another route by using the Navigator.push() method.
Step 3: Finally, navigate to the first route by using the Navigator.pop() method.

Gestures :

Gestures are used to interact with an application. It is generally used in touch- based devices to physically interact with the application. It can be as simple as a single tap on the screen to a more complex physical interaction like swiping in a specific direction to scrolling down an application. It is heavily used in gaming and more or less every application requires it to function as devices turn more touch-based than ever. In this article, we will discuss them in detail.
Some widely used gestures are mentioned here :

- Tap: Touching the surface of the device with the fingertip for a small duration of time period and finally releasing the fingertip.
- Double Tap: Tapping twice in a short time.
- Drag: Touching the surface of the device with the fingertip and then moving the fingertip in a steadily and finally releasing the fingertip.

- Flick: Similar to dragging, but doing it in a speedier way.

- Pinch: Pinching the surface of the device using two fingers.
- Zoom: Opposite of pinching.
- Panning: Touching the device surface with the fingertip and moving it in the desired direction without releasing the fingertip.

The GestureDetector widget in flutter is used to detect physical interaction with the application on the UI. If a widget is supposed to experience a gesture, it is kept inside the GestureDetector widget. The same widget catches the gesture and returns the appropriate action or response.

Below is the list of gestures and their corresponding events :

Tap

- onTapDown
- onTapUp
- onTap
- onTapCancel Double tap
- onDoubleTap Long press
- onLongPress Vertical drag
- onVerticalDragStart
- onVerticalDragUpdate
- onVerticalDragEnd Horizontal drag
- onHorizontalDragStart
- onHorizontalDragUpdate
- onHorizontalDragEnd Pan
- onPanStart
- onPanUpdate
- onPanEnd

Code:

```
class NutritionTrackerPage extends StatefulWidget {
  @override
  _NutritionTrackerPageState createState() => _NutritionTrackerPageState();
}

class _NutritionTrackerPageState extends State<NutritionTrackerPage> {
  TextEditingController mealTypeController = TextEditingController();
  TextEditingController caloriesController = TextEditingController();
  List<String> nutritionEntries = [];

  void _saveNutrition() {
    String mealType = mealTypeController.text;
    int calories = int.tryParse(caloriesController.text) ?? 0;
```

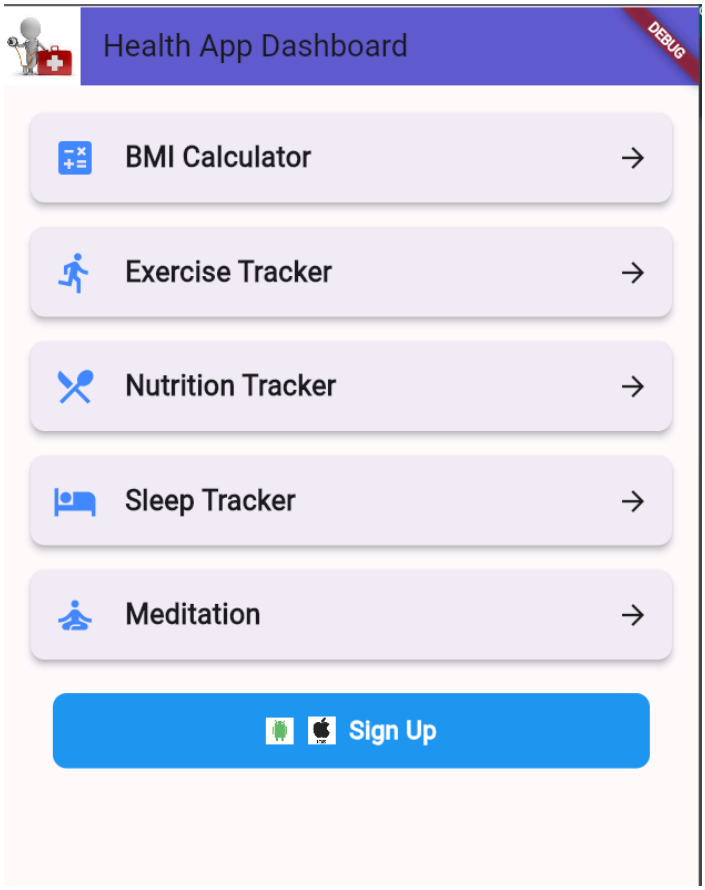
```
  // Build the nutrition entry string
  String entry = '$mealType - $calories calories';

  // Add the entry to the list
  setState(() {
    nutritionEntries.add(entry);
  });

  // Clear the text fields after saving
  mealTypeController.clear();
  caloriesController.clear();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Nutrition Tracker'),
    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.start,
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          TextField(
            controller: mealTypeController,
            decoration: InputDecoration(
              labelText: 'Meal Type',
            ),
          ),
          SizedBox(height: 20),
          TextField(
            controller: caloriesController,
            keyboardType: TextInputType.number,
            decoration: InputDecoration(
              labelText: 'Calories',
            ),
          ),
          SizedBox(height: 20),
          ElevatedButton(
            onPressed: _saveNutrition,
            child: Text('Save Nutrition'),
```

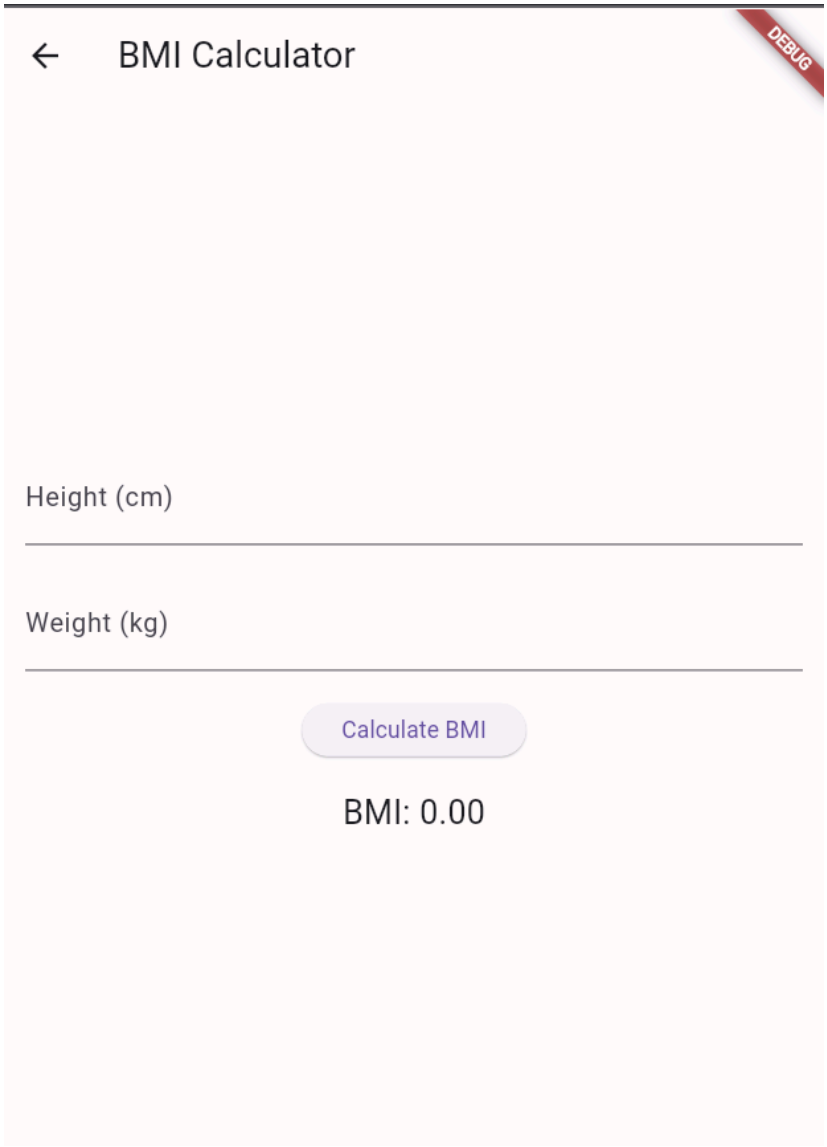
```
            ),
            SizedBox(height: 20),
            Text(
              'Nutrition Entries:',
              style: TextStyle(
                fontSize: 18,
                fontWeight: FontWeight.bold,
              ),
            ),
            SizedBox(height: 10),
            Expanded(
              child: ListView.builder(
                itemCount: nutritionEntries.length,
                itemBuilder: (context, index) {
                  return ListTile(
                    title: Text(nutritionEntries[index]),
                  );
                },
              ),
            ),
          ],
        ),
      ),
    );
  }
}


class SleepTrackerPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Sleep Tracker'),
      ),
      body: Center(
        child: Text(
          'This is the Sleep Tracker page.',
          style: TextStyle(fontSize: 20),
        ),
      ),
    );
  }
}
```

```
class MeditationPage extends StatefulWidget {
  @override
  _MeditationPageState createState() => _MeditationPageState();
}

class _MeditationPageState extends State<MeditationPage> {
  int _durationInSeconds = 0;
  bool _isTimerRunning = false;

  void _startTimer(int duration) {
    setState(() {
      _isTimerRunning = true;
    });

    Future.delayed(Duration(seconds: duration), () {
      setState(() {
        _isTimerRunning = false;
      });
      _showTimerFinishedDialog();
    });
  }

  void _showTimerFinishedDialog() {
    showDialog(
      context: context,
      builder: (context) => AlertDialog(
        title: Text('Meditation Timer'),
        content: Text('Your meditation session has ended.'),
        actions: [
          TextButton(
            onPressed: () => Navigator.pop(context),
            child: Text('OK'),
          ),
        ],
      ),
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Meditation'),
```

```
),
body: Center(
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    crossAxisAlignment: CrossAxisAlignment.center,
    children: [
      Text(
        'Meditation Timer',
        style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
      ),
      SizedBox(height: 20),
      Text(
        'Set Duration (minutes)',
        style: TextStyle(fontSize: 18),
      ),
      SizedBox(height: 10),
      Slider(
        value: _durationInSeconds.toDouble(),
        onChanged: (value) {
          setState(() {
            _durationInSeconds = value.toInt();
          });
        },
        min: 0,
        max: 60,
        divisions: 12,
        label: _durationInSeconds.toString(),
      ),
      SizedBox(height: 20),
      ElevatedButton(
        onPressed: () {
          _startTimer(_durationInSeconds * 60);
        },
        child: Text('Start Meditation'),
      ),
      SizedBox(height: 20),
      if (_isTimerRunning)
        CircularProgressIndicator()
      else
        Text(
          'Tap "Start Meditation" to begin your session.',
          textAlign: TextAlign.center,
          style: TextStyle(fontSize: 16),
        ),
```

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

class SignUpForm extends StatelessWidget {
  const SignUpForm({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Sign Up'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: <Widget>[
            // Your sign-up form fields go here
            TextFormField(
              decoration: InputDecoration(labelText: 'Name'),
            ),
            TextFormField(
              decoration: InputDecoration(labelText: 'Email'),
            ),
            TextFormField(
              decoration: InputDecoration(labelText: 'Password'),
              obscureText: true,
            ),
            const SizedBox(height: 20),
            ElevatedButton(
              onPressed: () {
                // Implement sign-up logic here
              },
              child: const Text('Sign Up'),
            ),
          ],
        ),
      ),
    );
  }
```


Health App Dashboard
DEBUG
BMI Calculator
Exercise Tracker
Nutrition Tracker
Sleep Tracker
Meditation
Sign Up



Conclusion:

We have understood the concept of gestures, their use and implemented it in our flutter app as a search bar. Also, we created two pages and routed them in our app and further enabled navigation. The flow is smooth for our app.