

Experiment 5

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

Theory :

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 an interesting feature in Flutter that allows us to interact with the mobile app (or any touch-based device). Generally, gestures define any physical action or movement of a user in the intention of specific control of the mobile device. Some of the examples of gestures are:

- When the mobile screen is locked, you slide your finger across the screen to unlock it.
- Tapping a button on your mobile screen, and
- Tapping and holding an app icon on a touch-based device to drag it across screens.

We use all these gestures in everyday life to interact with your phone or touch-based device.

Flutter divides the gesture system into two different layers, which are given below:

1. Pointers
2. Gestures

Implementation:

Code for navigation and routes:

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

void main() {
  runApp(MaterialApp(
    initialRoute: '/',
    routes: {
      '/': (context) => MobileLoginPage(),
      '/form': (context) => FormScreen(),
    },
  ));
}

class MobileLoginPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Login'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            TextFormField(
              decoration: InputDecoration(
                labelText: 'Email',
                border: OutlineInputBorder(),
              ),
            ),
            SizedBox(height: 20),
            TextFormField(

```

```
        decoration: InputDecoration(
          labelText: 'Password',
          border: OutlineInputBorder(),
        ),
        obscureText: true,
      ),
      SizedBox(height: 20),
      ElevatedButton(
        onPressed: () {
          Navigator.pushNamed(context, '/form');
        },
        child: Text('Login'),
      ),
      SizedBox(height: 10),
      Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Text('Don\'t have an account?'),
          SizedBox(width: 5),
          TextButton(
            onPressed: () {
              // Add navigation to sign up page
            },
            child: Text('Sign Up'),
          ),
        ],
      ),
    ],
  ),
),
);
}

class FormScreen extends StatefulWidget {
```

```
@override
  _FormScreenState createState() => _FormScreenState();
}

class _FormScreenState extends State<FormScreen> {
  final _formKey = GlobalKey<FormState>();

  String? _name;
  int? _age;
  String? _gender;
  String? _city;
  String? _selectedArtForm;

  List<String> _artForms = [
    'Painting',
    'Sculpture',
    'Music',
    'Dance',
    'Literature',
    'Photography',
    'Film',
  ];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Artistic Form'),
      ),
      body: Padding(
        padding: EdgeInsets.all(16.0),
        child: Form(
          key: _formKey,
          child: SingleChildScrollView(
            child: Column(
```

```
crossAxisAlignment: CrossAxisAlignment.stretch,
children: <Widget>[
  TextFormField(
    decoration: InputDecoration(labelText: 'Name'),
    validator: (value) {
      if (value == null || value.isEmpty) {
        return 'Please enter your name';
      }
      return null;
    },
    onSave: (value) => _name = value,
  ),
  TextFormField(
    decoration: InputDecoration(labelText: 'Age'),
    keyboardType: TextInputType.number,
    validator: (value) {
      if (value == null || value.isEmpty) {
        return 'Please enter your age';
      }
      return null;
    },
    onSave: (value) => _age = int.tryParse(value!),
  ),
  TextFormField(
    decoration: InputDecoration(labelText: 'Gender'),
    validator: (value) {
      if (value == null || value.isEmpty) {
        return 'Please enter your gender';
      }
      return null;
    },
    onSave: (value) => _gender = value,
  ),
  TextFormField(
    decoration: InputDecoration(labelText: 'City'),
```

```
        validator: (value) {
          if (value == null || value.isEmpty) {
            return 'Please enter your city';
          }
          return null;
        },
        onSave: (value) => _city = value,
      ),
      DropdownButtonFormField<String>(
        value: _selectedArtForm,
        items: _artForms.map((artForm) {
          return DropdownMenuItem(
            value: artForm,
            child: Text(artForm),
          );
        }).toList(),
        onChanged: (value) {
          setState(() {
            _selectedArtForm = value;
          });
        },
        decoration: InputDecoration(labelText: 'Interested Art
Form'),
        validator: (value) {
          if (value == null) {
            return 'Please select an art form';
          }
          return null;
        },
      ),
      SizedBox(height: 20),
      ElevatedButton(
        onPressed: () {
          if (_formKey.currentState!.validate()) {
            _formKey.currentState!.save();
            // Process the data
```

```
        print('Name: $_name');
        print('Age: $_age');
        print('Gender: $_gender');
        print('City: $_city');
        print('Interested Art Form: $_selectedArtForm');
      }
    },
    child: Text('Submit'),
  ),
],
),
),
),
),
),
);
}
```

Explanation:

navigation and routes are used to handle the movement between different screens (or pages) within the app. Here's how it works:

Setting up Routes:

- In the main() function, the MaterialApp widget is used to define the root of the widget tree for the app. Within the MaterialApp, the initialRoute property is set to '/', indicating that the '/' route will be the initial route displayed when the app is launched.
- The routes property is a map that associates route names with builder functions that return the corresponding widgets. In this case, two routes are defined: '/' and '/form'.
- The '/' route is associated with the MobileLoginPage widget, and the '/form' route is associated with the FormScreen widget.

Navigation:

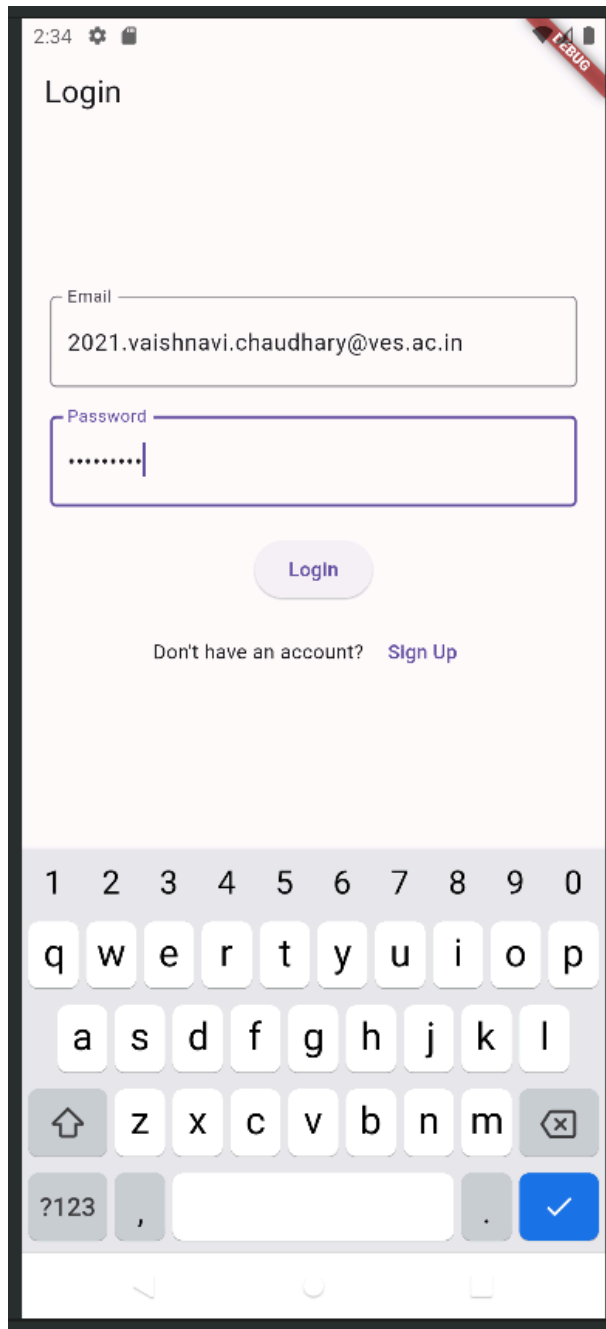
- In the MobileLoginPage widget, when the "Login" button is pressed (onPressed callback of the ElevatedButton), the Navigator.pushNamed method is used to navigate to the '/form' route. This method takes the BuildContext and the route name as arguments.

- Similarly, you could navigate to other routes using `Navigator.pushNamed(context, '/desired_route_name')`.

Building Widgets for Each Route:

- Each route defined in the `routes` property corresponds to a specific widget that will be displayed when the route is navigated to.
- The `MobileLoginPage` widget is associated with the `'/'` route, and it represents the login screen.
- The `FormScreen` widget is associated with the `'/form'` route, and it represents the form screen where the user can input their information.

By using navigation and routes, the app can easily transition between different screens based on user interactions, providing a seamless user experience.



2:34

Login

Email

2021.vaishnavi.chaudhary@ves.ac.in

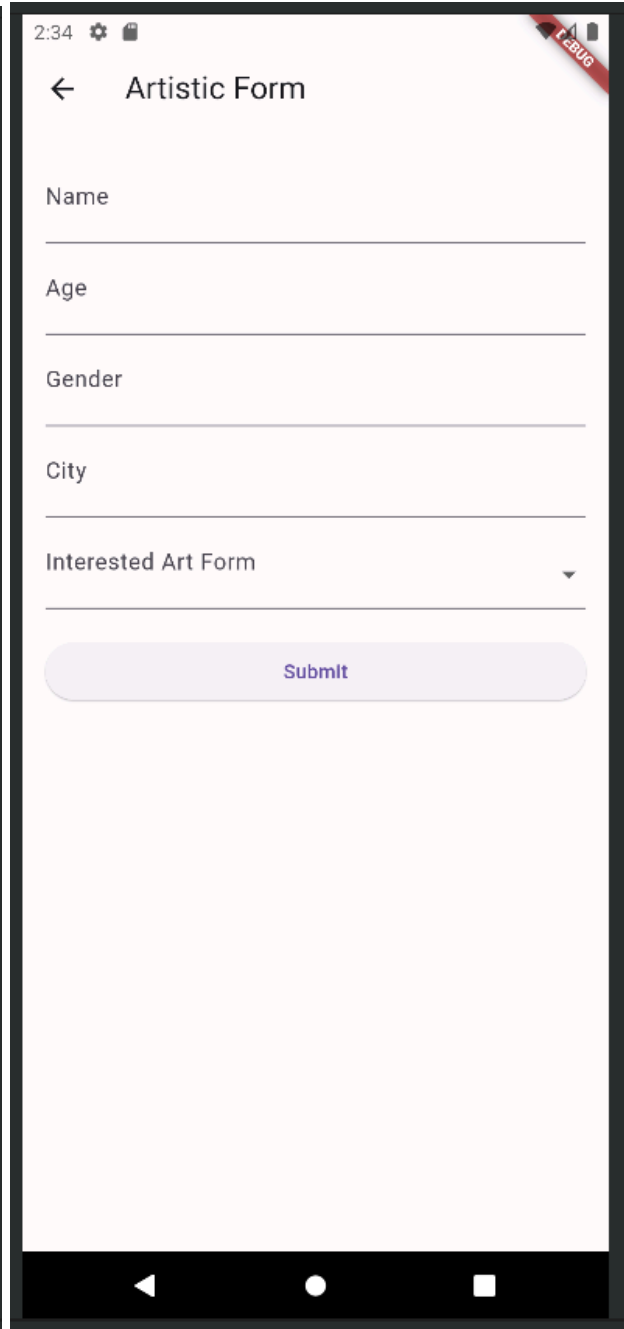
Password

.....

Login

Don't have an account? [Sign Up](#)

1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
↑ z x c v b n m ↵
?123 , . ✓



2:34

← Artistic Form

Name

Age

Gender

City

Interested Art Form ▼

Submit

Code for gestures:

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Gestures Example',
      home: GestureExample(),
    );
  }
}

class GestureExample extends StatefulWidget {
  @override
  _GestureExampleState createState() => _GestureExampleState();
}

class _GestureExampleState extends State<GestureExample> {
  String _gestureEvent = 'No gesture detected';

  void _updateGestureEvent(String event) {
    setState(() {
      _gestureEvent = event;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

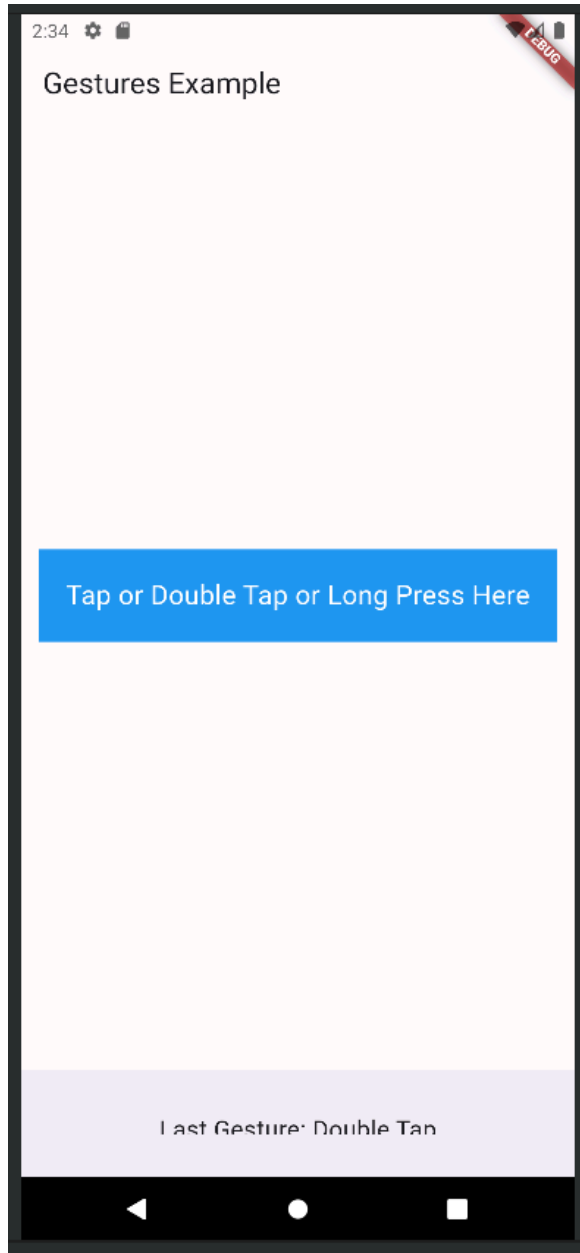
```
appBar: AppBar(  
  title: Text('Gestures Example'),  
) ,  
body: Center(  
  child: GestureDetector(  
    onTap: () {  
      _updateGestureEvent('Tap');  
    },  
    onDoubleTap: () {  
      _updateGestureEvent('Double Tap');  
    },  
    onLongPress: () {  
      _updateGestureEvent('Long Press');  
    },  
    child: Container(  
      padding: EdgeInsets.all(20.0),  
      color: Colors.blue,  
      child: Text(  
        'Tap or Double Tap or Long Press Here',  
        style: TextStyle(color: Colors.white, fontSize: 20.0),  
      ),  
    ),  
  ),  
) ,  
bottomNavigationBar: BottomAppBar(  
  child: Container(  
    padding: EdgeInsets.symmetric(vertical: 20.0),  
    alignment: Alignment.center,  
    child: Text(  
      'Last Gesture: $_gestureEvent',  
      style: TextStyle(fontSize: 16.0),  
    ),  
  ),  
) ,  
) ;
```

```
}  
}
```

Explanation:

In this example:

- We have a GestureDetector wrapped around a Container widget.
- The GestureDetector listens for different gestures like onTap, onDoubleTap, and onLongPress.
- When a gesture is detected, the corresponding callback function updates the `_gestureEvent` variable, which is displayed at the bottom of the screen.
- Try tapping, double-tapping, and long-pressing on the blue container to see the gesture events being detected and displayed at the bottom of the screen.

**Conclusion:**

We understood the concepts of navigation , routing and gestures in Flutter.

We implemented navigation and routing for the 2 pages ie. login page and Form page .

We implemented gestures in a basic Flutter application.