Experiment 5

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

Theory :

Navigation and routing are some of the core concepts of all mobile applications, 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.

Let us take a simple example to understand the navigation between two routes:

Code:

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

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

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: [
           Image(
             image: AssetImage('images/download.png'),
             height: 150.0,
             width: 150.0,
           ),
           SizedBox(height: 20),
           TextFormField(
             decoration: InputDecoration(
               labelText: 'Email',
               border: OutlineInputBorder(),
             ),
           ),
           SizedBox(height: 20),
           TextFormField(
             decoration: InputDecoration(
               labelText: 'Password',
               border: OutlineInputBorder(),
             ),
             obscureText: true,
           ),
           SizedBox(height: 20),
           ElevatedButton(
             onPressed: () {
               // Add login logic here
             },
             child: Text('Login'),
           ),
           SizedBox(height: 10),
           Row(
             mainAxisAlignment: MainAxisAlignment.center,
             children: [
               Text("Don't have an account? "),
               TextButton(
                 onPressed: () {
                   Navigator.pushNamed(context, '/form');
                 },
                 child: Text('Sign Up'),
               ),
             ],
           ),
         ],
       ),
     ),
   );
 }
}

class SignUpPage extends StatefulWidget {
 @override
 _SignUpPageState createState() => _SignUpPageState();
}

class _SignUpPageState extends State<SignUpPage> {
 final _formKey = GlobalKey<FormState>();
 String? _username;
 String? _email;
 String? _phoneNumber;
 String? _password;
 String? _foodPreference;
 List<String> _foodPreferences = [
   'Vegetarian',
   'Non Vegetarian',
   'Vegan',
 ];

 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: Text('Sign Up Page'),
     ),
     body: Padding(
       padding: EdgeInsets.all(16.0),
       child: Form(
         key: _formKey,
         child: SingleChildScrollView(
           child: Column(
             crossAxisAlignment: CrossAxisAlignment.stretch,
             children: <Widget>[
               Image(
                 image: AssetImage('images/download.png'),
                 height: 150.0,
                 width: 150.0,
               ),
               TextFormField(
                 decoration: InputDecoration(labelText: 'Username'),
                 validator: (value) {
                   if (value == null || value.isEmpty) {
                     return 'Please enter a username';
                   }
                   return null;
                 },
                 onSaved: (value) => _username = value,
               ),
               TextFormField(
                 decoration: InputDecoration(labelText: 'Email'),
                 validator: (value) {
                   if (value == null || value.isEmpty) {
                     return 'Please enter an email address';
                   }
                   return null;
                 },
                 onSaved: (value) => _email = value,
               ),
               TextFormField(
                 decoration: InputDecoration(labelText: 'Phone Number'),
                 validator: (value) {
                   if (value == null || value.isEmpty) {
                     return 'Please enter a phone number';
                   }
                   return null;
                 },
                 onSaved: (value) => _phoneNumber = value,
               ),
               TextFormField(
                 decoration: InputDecoration(labelText: 'Password'),
                 obscureText: true,
                 validator: (value) {
                   if (value == null || value.isEmpty) {
                     return 'Please enter a password';
                   }
                   return null;
                 },
                 onSaved: (value) => _password = value,
               ),
               DropdownButtonFormField<String>(
                 value: _foodPreference,
                 items: _foodPreferences.map((preference) {
                   return DropdownMenuItem(
                     value: preference,
                     child: Text(preference),
                   );
                 }).toList(),
                 onChanged: (value) {
                   setState(() {
                     _foodPreference = value;
                   });
                 },
                 decoration: InputDecoration(labelText: 'Food Preference'),
                 validator: (value) {
                   if (value == null) {
                     return 'Please select a food preference';
                   }
                   return null;
                 },
               ),
               SizedBox(height: 20),
               ElevatedButton(
                 onPressed: () {
                   if (_formKey.currentState!.validate()) {
                     _formKey.currentState!.save();
                     // After successful form submission, navigate back to
login page
                     Navigator.pop(context);
                   }
                 },
                 child: Text('Submit'),
               ),
             ],
           ),
         ),
       ),
     ),
   );
 }
}
```

Explanation of the code:
This code is a Flutter application that implements two screens: a login page and a sign-up page. Here's an explanation of the code:

1. Main Function and MaterialApp:
   - The **main()** function is the entry point of the application. It calls **runApp()** with a **MaterialApp** widget.
   - **MaterialApp** is a convenience widget that provides several features commonly required in mobile applications, such as navigation and theme management.

- **initialRoute** specifies the initial route of the application, which is '/' (the login page).
- **routes** defines the named routes of the application, mapping route names to corresponding builder functions.

2. Login Page (MobileLoginPage):

- This page displays a login form with fields for email and password.
- It uses a **Scaffold** widget to provide the basic structure of the page, including an AppBar with the title "Login".
- The form fields are wrapped in a **Column** widget to arrange them vertically.
- A **TextFormField** widget is used for email and password input.
- An **ElevatedButton** widget is used for the login button, which currently has no functionality.
- A **TextButton** widget is used to navigate to the sign-up page when clicked.

3. **Sign-Up Page (SignUpPage)**:

- This page allows users to sign up with a username, email, phone number, password, and food preference.
- It also uses a **Scaffold** widget for the basic page structure.
- The form fields are wrapped in a **Form** widget, allowing for form validation and submission.
- Each form field is represented by a **TextFormField** widget with appropriate decoration and validation logic.
- The food preference field uses a **DropdownButtonFormField** widget to display a dropdown menu of options.
- An **ElevatedButton** widget is used to submit the form. Upon successful validation, the form data is saved and the sign-up page is closed (popped from the navigation stack).

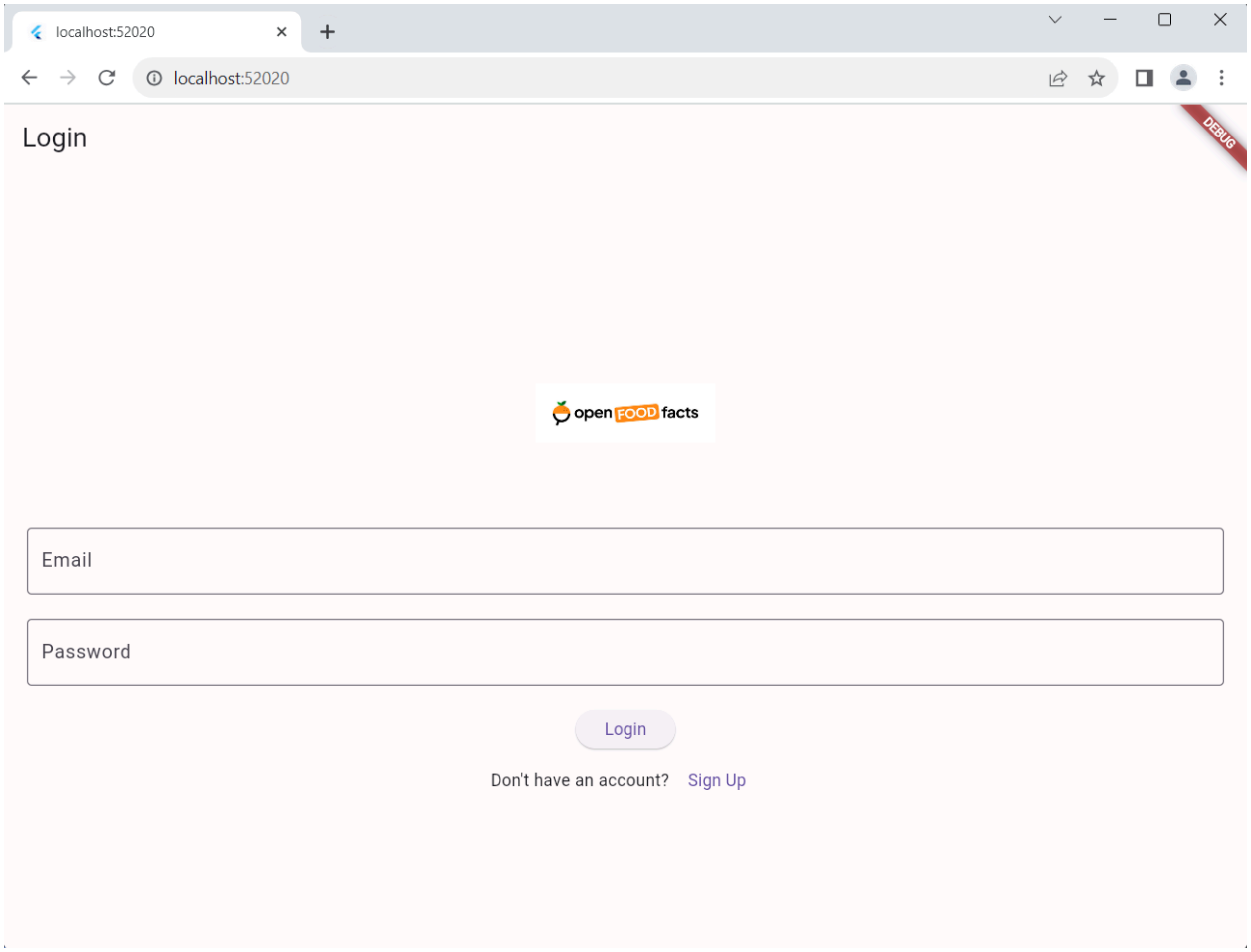
4. State Management:

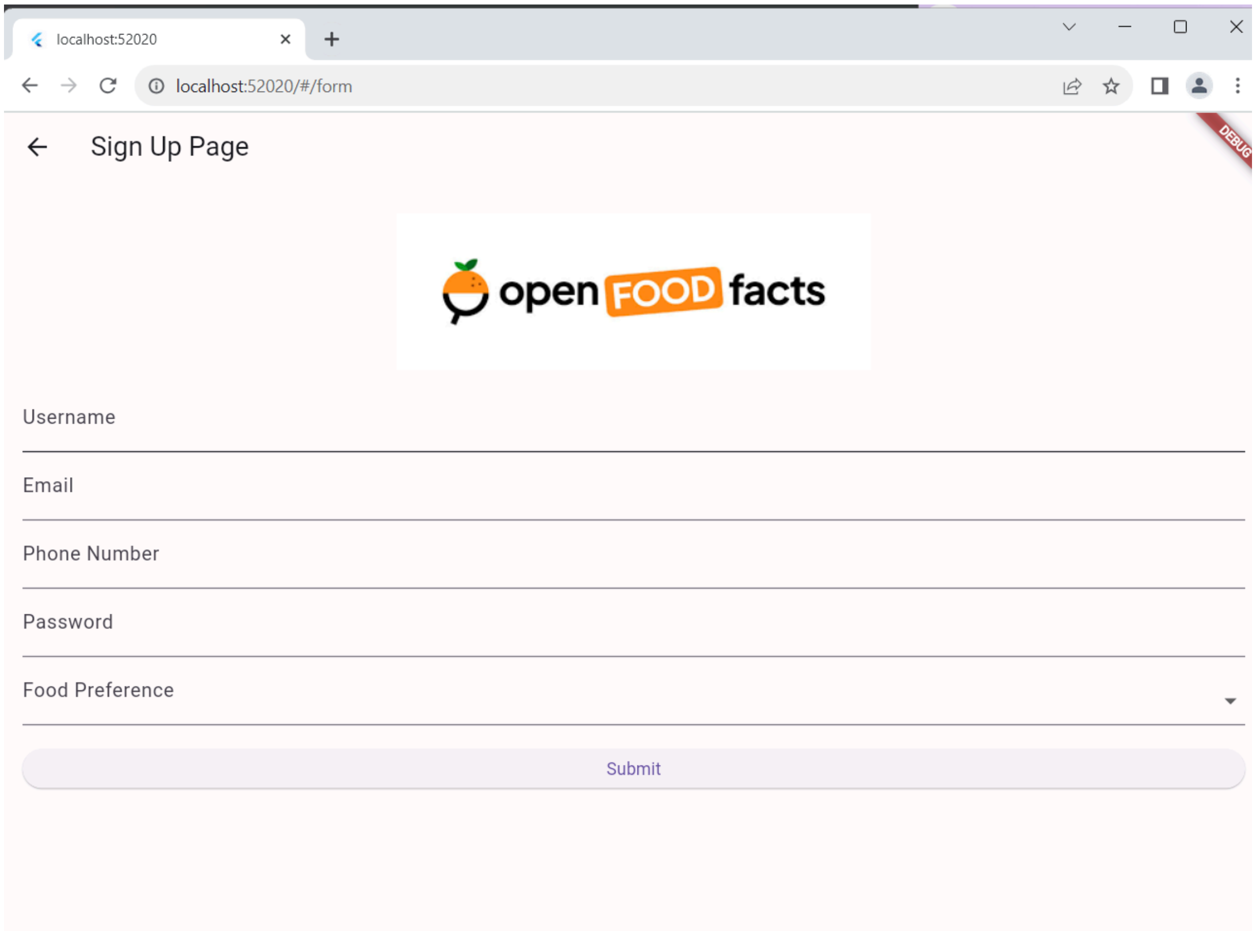
- The sign-up page (**SignUpPage**) is a stateful widget (**StatefulWidget**) because it needs to maintain the form state.
- It uses a GlobalKey<FormState> to access the state of the form for validation and saving.

5. Navigation:
- Navigation between the login page and sign-up page is handled using named routes. When the "Sign Up" button is pressed on the login page, it navigates to the sign-up page ('/form'). After successful sign-up, the sign-up page is closed and the user is taken back to the login page.

Overall, this code provides a basic login and sign-up functionality for a mobile application built with Flutter. It demonstrates how to create forms, handle form submission, perform form validation, and navigate between screens using named routes.

Output:




localhost:52020

localhost:52020/#/form

Sign Up Page

Username

Email

Phone Number

Password

Food Preference

Submit

Conclusion:

In this practical, we developed a Flutter application featuring login and sign-up functionalities. Leveraging named routes, we enabled smooth navigation between the login and sign-up pages, ensuring a structured approach to screen transitions. Using `TextFormField`, we created input fields for email, password, username, phone number, and food preference, implementing validation to maintain data integrity. Through `GlobalKey<FormState>`, we managed form state, facilitating validation and submission processes. The user interface was designed with visually appealing elements such as `Scaffold`, `AppBar`, and images, enhancing the overall user experience.