MPL Lab 5
Mayur Jaiswal
D15B
24

Aim: Implement navigation between multiple screens and incorporate gesture detection to enhance user interaction in a Flutter application.

Theory: Navigation and routing are fundamental for multi-screen applications. Flutter's `Navigator` widget manages a stack of routes, enabling seamless transitions between screens. Named routes provide a cleaner way to define and navigate to different screens. Additionally, gestures like taps, swipes, and pinches are integral to modern app interfaces. Flutter's GestureDetector widget allows developers to capture and respond to various user gestures, making the app more interactive and responsive.

Steps to Perform:

1. **Set Up a New Flutter Project:**
   - Create a new Flutter application: flutter create navigation_app.
   - Navigate to the project directory: cd navigation_app.Flutter documentation+2Firebase+2Firebase+2
2. **Define Named Routes:**

In lib/main.dart, set up the MaterialApp with named routes:

```
void main() {
  runApp(MaterialApp(
    initialRoute: '/',
    routes: {
      '/': (context) => HomeScreen(),
      '/second': (context) => SecondScreen(),
    },
  ));
}
```

   -

3. **Create Screens:**
   - Define two stateless widgets, HomeScreen and SecondScreen, each returning a Scaffold with an AppBar and body content.
4. **Implement Navigation:**

In HomeScreen, add a button that navigates to SecondScreen when pressed:

```
ElevatedButton(
  onPressed: () {
    Navigator.pushNamed(context, '/second');
  },
  child: Text('Go to Second Screen'),
),
```

   -

In SecondScreen, add a button to navigate back:

```
ElevatedButton(
  onPressed: () {
    Navigator.pop(context);
  },
  child: Text('Go Back'),
),
```

○

5. **Handle Gestures:**

Wrap widgets with GestureDetector to detect gestures:

```
GestureDetector(
  onTap: () {
    // Handle tap
  },
  child: Container(
    color: Colors.blue,
    height: 100,
    width: 100,
  ),
),
```

---

2. Applying Navigation, Routing, and Gestures in a Flutter App *(continued)*

5. Handle Gestures (continued):
You can use GestureDetector to capture various gesture events such as onTap, onDoubleTap, onLongPress, onPanUpdate, etc. This can be useful for creating interactive UI components

```
GestureDetector(
  onTap: () {
    print('Box tapped!');
  },
  onDoubleTap: () {
    print('Box double tapped!');
  },
  child: Container(
    color: Colors.amber,
    height: 100,
    width: 100,
    child: Center(child: Text("Tap Me")),
  ),
),
```

6. Navigate with Parameters (Optional):

```
Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => SecondScreen(data: 'Hello'),
  ),
);
```

In the second screen:

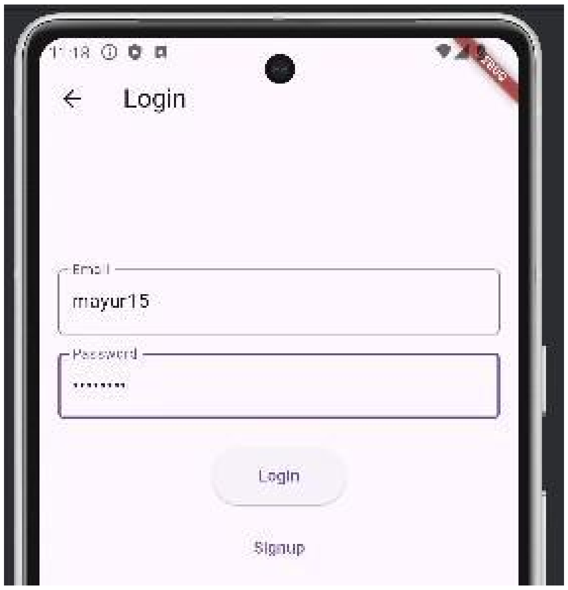
```
class SecondScreen extends StatelessWidget {
  final String data;
  SecondScreen({required this.data});
  ...
}
```

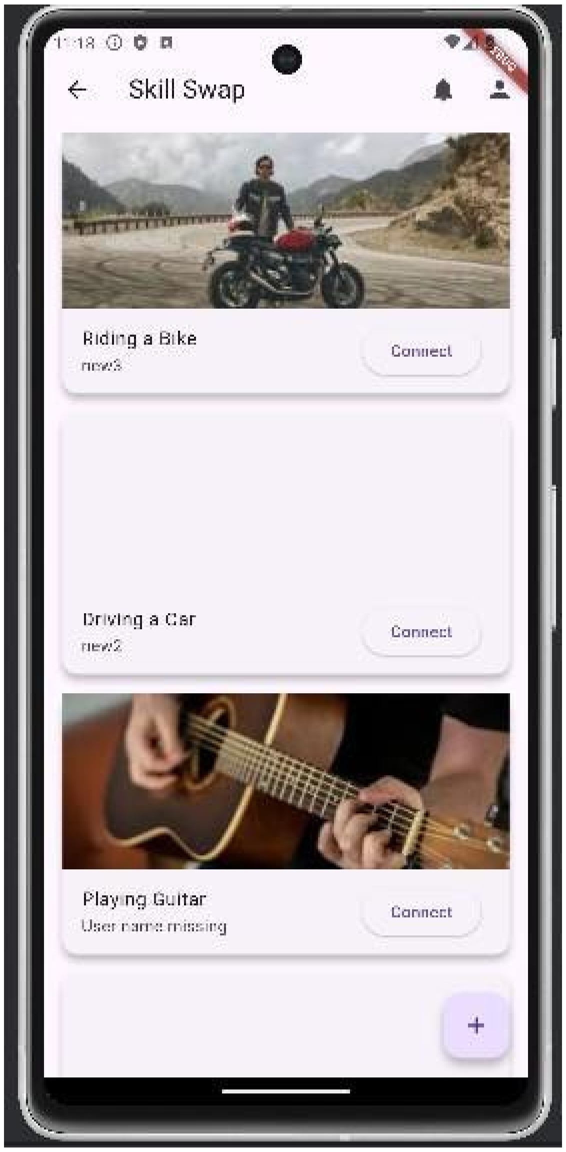
Key Features:

- **Named & Anonymous Routing:** Manage multiple pages easily.
- **Gesture Detection:** Supports tap, swipe, drag, long press, etc.
- **Flexible Navigation Stack:** Push, pop, replace, and popUntil available.
- **Parameter Passing:** Allows context-aware navigation.

Once Logged in the app gets navigated to the homescreen and “addskill” button navigates to the addskillscreen

```
// screens/login_screen.dart
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'home_screen.dart';
import 'signup_screen.dart';

class LoginScreen extends StatefulWidget {
 @override
 _LoginScreenState createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
 final _formKey = GlobalKey<FormState>();
 final _emailController = TextEditingController();
 final _passwordController =
 TextEditingController();

 Future<void> _login() async {
  if (_formKey.currentState!.validate()) {
   try {
    await FirebaseAuth.instance.signInWithEmailAndPassword(
     email: _emailController.text,
     password: _passwordController.text,
    );
    Navigator.pushReplacement(
      context, MaterialPageRoute(builder: (context) => HomeScreen()));
   } on FirebaseAuthException catch (e) {
    if (mounted) {
     ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Failed to login: ${e.message}')),
     );
    }
   }
  }
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(title: Text('Login')),
body: Padding(
  padding: const EdgeInsets.all(16.0),
  child: Form(
    key: _formKey,
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        TextFormField(
          controller: _emailController,
          decoration: InputDecoration(
          labelText: 'Email',
            border: OutlineInputBorder(),
          ),
          validator: (value) => value!.isEmpty ? 'Enter email' : null,
        ),
        SizedBox(height: 16),
        TextFormField(
          controller: _passwordController,
          obscureText: true,
          decoration: InputDecoration(
            labelText: 'Password',
            border: OutlineInputBorder(),
          ),
          validator: (value) => value!.isEmpty ? 'Enter password' : null,
        ),
        SizedBox(height: 24),
        ElevatedButton(
        onPressed: _login,
        child: Text('Login'),
          style: ElevatedButton.styleFrom(
            padding: EdgeInsets.symmetric(horizontal: 40, vertical: 15),
          ),
        ),
        SizedBox(height: 12),
        TextButton(
            onPressed: () {
              Navigator.push(context,
                  MaterialPageRoute(builder: (context) => SignupScreen()));
            },
            child: Text("Signup"))
      ],
    ),
  ),
),
    );
  }
}
// screens/add_skill_screen.dart
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'skill.dart';
import 'database_service.dart';
import 'user_model.dart';

class AddSkillPage extends StatefulWidget {
  final String userName;

  const AddSkillPage({Key? key, required this.userName}) : super(key: key);

  @override
  _AddSkillPageState createState() => _AddSkillPageState();
}

class _AddSkillPageState extends State<AddSkillPage> {
  final _skillController = TextEditingController();
  final DatabaseService _databaseService = DatabaseService();
  UserModel? user;
  String? address;

  @override
  void initState() {
    super.initState();
    _loadUserData();
  }

  Future<void> _loadUserData() async {
    final currentUser = FirebaseAuth.instance.currentUser;
    if (currentUser != null) {
      try {
        user = await _databaseService.getUser(currentUser.uid);
        setState(() {
          address = user?.address;
        });

      } catch (error) {
        print('Error loading user data: $error');
      }
    }
  }

  Future<void> _addSkill() async {
    try {
      final userId = FirebaseAuth.instance.currentUser!.uid;
      final skill = Skill(
          userId: userId,
          userName: widget.userName,
          skillName: _skillController.text,
          address: address?? "No Address"
      );
      await _databaseService.addSkill(skill);
      if (mounted) {
        Navigator.pop(context);
      }
    } catch (e) {
      if (mounted) {
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(content: Text('Failed to add skill: ${e.toString()}')),
        );
      }
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Add Skill')),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: [
            TextField(
              controller: _skillController,
              decoration: InputDecoration(labelText: 'Skill'),
            ),
            SizedBox(height: 16),
            ElevatedButton(onPressed: _addSkill, child: Text('Add')),
          ],
        ),
      ),
    );
  }
}
```


1:18
Login
Email
mayur15
Password
••••••••
Login
Signup


11:18
Skill Swap
Riding a Bike
new3
Connect
Driving a Car
new2
Connect
Playing Guitar
User name missing
Connect
+