Name: Yash Uskelwar Roll No: 69 Div: D15B

MAD Lab Experiment No 5

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

Theory:

Navigation

- Purpose: Navigation refers to the process of guiding users through the different screens (or "routes") within your app, allowing them to explore information and complete tasks. It establishes a clear flow and user experience.
- Techniques in Flutter:
  a. Navigator: Flutter's built-in Navigator class is the central component for managing navigation. It provides methods for pushing new routes onto the navigation stack (moving forward), popping routes from the stack (going back), and replacing the current route.
  b. Named Routes: Named routes associate a unique string identifier with each route, making navigation more readable and maintainable. You define these routes in the MaterialApp or WidgetsApp constructor's routes property.

Routing

- Purpose: Routing defines the logic behind how the app determines which route (screen) to display based on user actions or data. It establishes the mapping between events and destinations.
- Implementation in Flutter:
  a. Basic Navigation: For simple navigation, you can use the Navigator.push() and Navigator.pop() methods directly.
  b. Declarative Routing (Packages): For complex applications with deep linking or advanced navigation patterns, consider using third-party routing packages like go_router or fluro. These packages provide a more declarative approach to defining routes and handling transitions.

Gestures

- Purpose: Gestures are user interactions with the touch screen that control the app's behavior. They allow users to navigate, interact with UI elements, and manipulate data.
- Handling Gestures in Flutter:
  a. Gesture Detectors: Flutter provides various gesture detectors (like

GestureDetector, TapGestureRecognizer, SwipeGestureRecognizer) to capture and interpret user gestures. These detectors trigger callbacks based on the type and timing of the gesture.

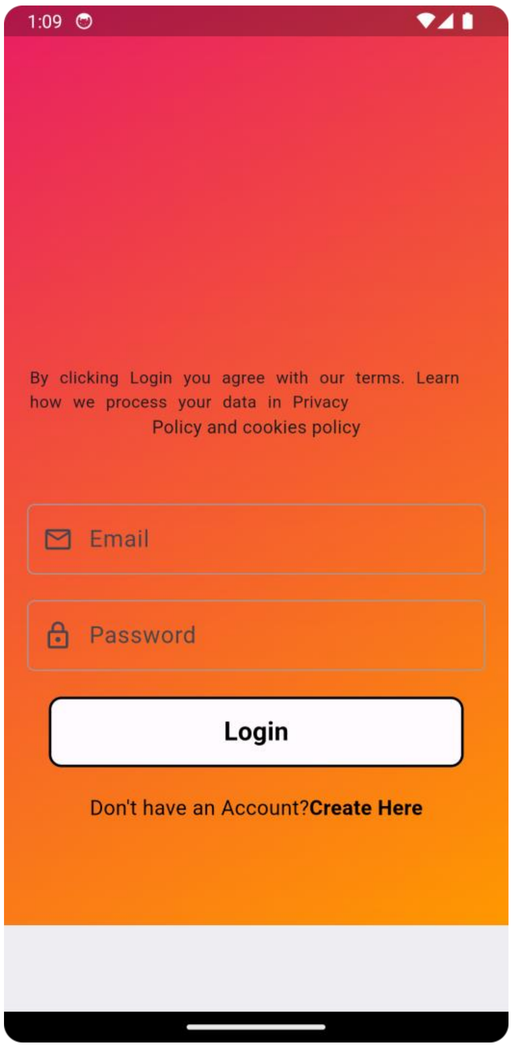
Code:

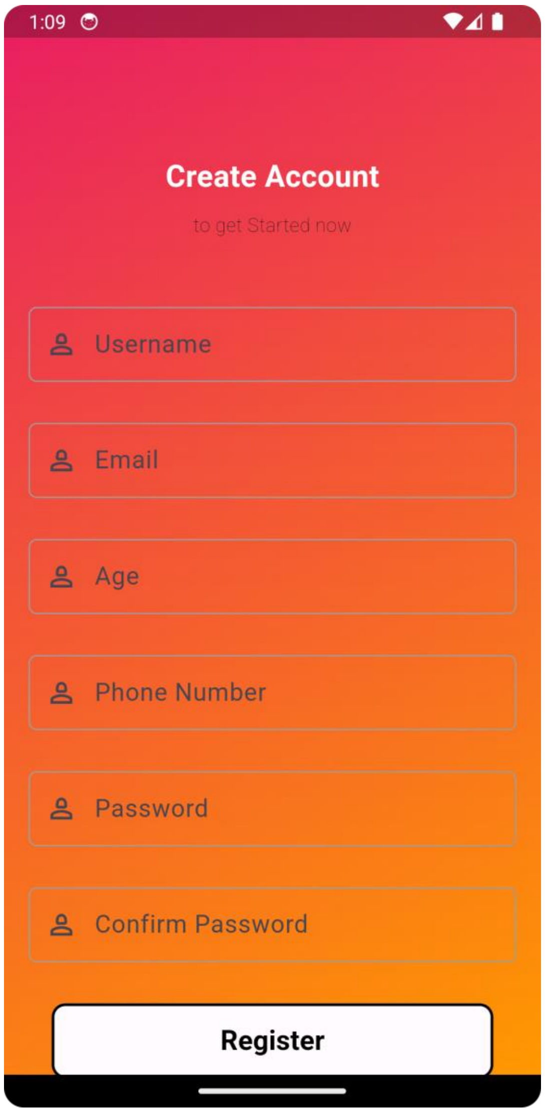
WelcomePage.dart:

```
import 'package:flutter/material.dart'; import
'package:flashcards_quiz/views/login_page.dart';

class WelcomePage extends StatelessWidget {
  const WelcomePage({super.key});

  @override
  Widget build(BuildContext context) { return Scaffold(
    backgroundColor: const Color.fromARGB(255, 215, 1, 11),
    body: Center(
        child: Column( mainAxisAlignment:
          MainAxisAlignment.center, children: [
           // Add your app logo image here
           Image.asset("assets/dictionary_img.png"),
           const SizedBox(height: 20), const Text(
           "Vocabulary Builder", style: TextStyle(
               fontSize: 30, fontWeight:
               FontWeight.bold, color:
               Colors.white,
             ),
           ),
           const SizedBox(height: 20),
           Padding( padding: const EdgeInsets.only(bottom: 30,
             right: 30), child: FloatingActionButton(
               onPressed: () => Navigator.push(
                 context,
                 MaterialPageRoute(builder: (context) => const LoginPage()),
               ),
               child: const Icon(Icons.arrow_forward),
             ),
           ),
          ],
      ),
    ),
  );
 }
}
```

<u>LoginPage.dart:</u> import 'package:flutter/material.dart';

```
import 'package:firebase_auth/firebase_auth.dart';
import
'package:flashcards_quiz/views/home_page.dart';

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

 @override
 State<LoginPage> createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
 final _usernameController = TextEditingController();
 final _passwordController = TextEditingController();
 final FirebaseAuth _auth = FirebaseAuth.instance;
 String _errorText = '';

 @override
 void dispose()
 {
  _usernameController.dispose();
  _passwordController.dispose();
  super.dispose();
 }

 Future<void> _signInWithEmailAndPassword() async {
  try {
   final UserCredential userCredential = await
    _auth.signInWithEmailAndPassword( email:
      _usernameController.text.trim(), password:
      _passwordController.text,
    );
    if (userCredential.user != null) {
      // Navigate to home page if authentication successful
      // ignore: use_build_context_synchronously
      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => const HomePage1()),
      );
    }
  } catch (e) {
    setState(() {
      _errorText =
          'Invalid email or password'; // Set error message for invalid credentials
    });
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text("Login"), backgroundColor: const
      Color.fromARGB(255, 215, 1, 11),
    ),
    backgroundColor: const Color.fromARGB(255, 215, 1, 11),
    body: SingleChildScrollView( // Make content scrollable
    padding: const EdgeInsets.all(20.0), child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
         // Add your image widget here
         Image.asset(
           "assets/login_imgg.png"
           , width: 250, height:
           250,
         ),
            const SizedBox(height: 20),
            TextField( controller:
              _usernameController, decoration:
              const InputDecoration(
                labelText: "Username",
                labelStyle: TextStyle(
                  color: Colors.white, // Set text color
                ),
              ),
            ),
            const SizedBox(height: 10),
            TextField( controller:
              _passwordController,
              obscureText: true, decoration:
              InputDecoration( labelText:
              "Password", labelStyle: const
              TextStyle(
                  color: Colors.white, // Set text color
                ),
                errorText: _errorText.isNotEmpty ? _errorText : null,
                errorStyle: const TextStyle(
                    color: Colors.white), // Set error text color
              ),
            ),
            const SizedBox(height: 20),
            ElevatedButton(
              onPressed:
                  _signInWithEmailAndPassword, // Call method for authentication
              child: const Text("Login"),
            ),
          ],
        ),
      ),
    );
  }
}
```
Output:


1:09
By clicking Login you agree with our terms. Learn how we process your data in Privacy Policy and cookies policy
Email
Password
Login
Don't have an Account?Create Here



Conclusion: The integration of navigation, routing, and gestures in Flutter app development significantly enhances user experience and interaction. Leveraging Flutter's robust navigation system, hierarchical routing, and diverse gesture recognizers, developers can create intuitive, seamless, and engaging applications. By prioritizing user-centric design and functionality.