dependencies:

flutter:

sdk: flutter

firebase\_core: ^2.10.0

firebase\_auth: ^4.6.2

cloud\_firestore: ^4.6.0

provider: ^6.0.5

import 'package:flutter/material.dart';

import 'package:firebase\_core/firebase\_core.dart';

import 'package:provider/provider.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

import 'todo\_screen.dart';

import 'auth\_screen.dart';

void main() async {

WidgetsFlutterBinding.ensureInitialized();

await Firebase.initializeApp();

runApp(const MyApp());

}

class MyApp extends StatelessWidget {

const MyApp({super.key});

@override

Widget build(BuildContext context) {

return StreamProvider<User?>.value(

value: FirebaseAuth.instance.authStateChanges(),

initialData: null,

child: MaterialApp(

title: 'Get It Done',

theme: ThemeData(primarySwatch: Colors.blue),

home: const AuthWrapper(),

),

);

}

}

class AuthWrapper extends StatelessWidget {

const AuthWrapper({super.key});

@override

Widget build(BuildContext context) {

final user = context.watch<User?>();

if (user != null) {

return const TodoScreen();

} else {

return const AuthScreen();

}

}

}

import 'package:flutter/material.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

class AuthScreen extends StatefulWidget {

const AuthScreen({super.key});

@override

State<AuthScreen> createState() => \_AuthScreenState();

}

class \_AuthScreenState extends State<AuthScreen> {

final \_formKey = GlobalKey<FormState>();

final \_emailController = TextEditingController();

final \_passwordController = TextEditingController();

bool \_isLogin = true;

String? \_errorMessage;

bool \_isLoading = false;

Future<void> \_submit() async {

if (!\_formKey.currentState!.validate()) return;

setState(() {

\_isLoading = true;

\_errorMessage = null;

});

try {

if (\_isLogin) {

await FirebaseAuth.instance.signInWithEmailAndPassword(

email: \_emailController.text.trim(),

password: \_passwordController.text.trim(),

);

} else {

await FirebaseAuth.instance.createUserWithEmailAndPassword(

email: \_emailController.text.trim(),

password: \_passwordController.text.trim(),

);

}

} on FirebaseAuthException catch (e) {

setState(() {

\_errorMessage = e.message;

});

} finally {

if (mounted) {

setState(() {

\_isLoading = false;

});

}

}

}

@override

void dispose() {

\_emailController.dispose();

\_passwordController.dispose();

super.dispose();

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text(\_isLogin ? 'Sign In' : 'Sign Up')),

body: Padding(

padding: const EdgeInsets.all(16),

child: Center(

child: SingleChildScrollView(

child: Form(

key: \_formKey,

child: Column(

children: [

if (\_errorMessage != null)

Text(

\_errorMessage!,

style: const TextStyle(color: Colors.red),

),

TextFormField(

controller: \_emailController,

decoration: const InputDecoration(labelText: 'Email'),

keyboardType: TextInputType.emailAddress,

validator: (val) {

if (val == null || !val.contains('@')) {

return 'Enter a valid email';

}

return null;

},

),

const SizedBox(height: 16),

TextFormField(

controller: \_passwordController,

decoration: const InputDecoration(labelText: 'Password'),

obscureText: true,

validator: (val) {

if (val == null || val.length < 6) {

return 'Password must be at least 6 characters';

}

return null;

},

),

const SizedBox(height: 20),

if (\_isLoading)

const CircularProgressIndicator()

else

ElevatedButton(

onPressed: \_submit,

child: Text(\_isLogin ? 'Sign In' : 'Sign Up'),

),

TextButton(

onPressed: () {

setState(() {

\_isLogin = !\_isLogin;

\_errorMessage = null;

});

},

child: Text(

\_isLogin

? "Don't have an account? Sign Up"

: "Already have an account? Sign In",

),

),

],

),

),

),

),

),

);

}

}

import 'package:flutter/material.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

class TodoScreen extends StatefulWidget {

const TodoScreen({super.key});

@override

State<TodoScreen> createState() => \_TodoScreenState();

}

class \_TodoScreenState extends State<TodoScreen> {

final \_taskController = TextEditingController();

CollectionReference get \_tasksRef => FirebaseFirestore.instance

.collection('users')

.doc(FirebaseAuth.instance.currentUser!.uid)

.collection('tasks');

Future<void> \_addTask() async {

final task = \_taskController.text.trim();

if (task.isEmpty) return;

await \_tasksRef.add({'task': task, 'done': false, 'timestamp': FieldValue.serverTimestamp()});

\_taskController.clear();

}

Future<void> \_toggleDone(String id, bool currentDone) async {

await \_tasksRef.doc(id).update({'done': !currentDone});

}

Future<void> \_deleteTask(String id) async {

await \_tasksRef.doc(id).delete();

}

Future<void> \_signOut() async {

await FirebaseAuth.instance.signOut();

}

@override

void dispose() {

\_taskController.dispose();

super.dispose();

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: const Text('Get It Done'),

actions: [

IconButton(

icon: const Icon(Icons.logout),

onPressed: \_signOut,

tooltip: 'Sign Out',

)

],

),

body: Column(

children: [

Padding(

padding: const EdgeInsets.all(12),

child: Row(

children: [

Expanded(

child: TextField(

controller: \_taskController,

decoration: const InputDecoration(

hintText: 'Add a new task',

border: OutlineInputBorder(),

),

),

),

const SizedBox(width: 8),

ElevatedButton(

onPressed: \_addTask,

child: const Text('Add'),

),

],

),

),

Expanded(

child: StreamBuilder<QuerySnapshot>(

stream: \_tasksRef.orderBy('timestamp', descending: true).snapshots(),

builder: (context, snapshot) {

if (snapshot.hasError) {

return const Center(child: Text('Error loading tasks'));

}

if (snapshot.connectionState == ConnectionState.waiting) {

return const Center(child: CircularProgressIndicator());

}

final tasks = snapshot.data?.docs ?? [];

if (tasks.isEmpty) {

return const Center(child: Text('No tasks yet. Add some!'));

}

return ListView.builder(

itemCount: tasks.length,

itemBuilder: (context, index) {

final doc = tasks[index];

final data = doc.data()! as Map<String, dynamic>;

final task = data['task'] ?? '';

final done = data['done'] ?? false;

return ListTile(

title: Text(

task,

style: TextStyle(

decoration: done ? TextDecoration.lineThrough : null,

color: done ? Colors.grey : Colors.black,

),

),

leading: Checkbox(

value: done,

onChanged: (\_) => \_toggleDone(doc.id, done),

),

trailing: IconButton(

icon: const Icon(Icons.delete),

onPressed: () => \_deleteTask(doc.id),

),

);

},

);

},

),

),

],

),

);

}

}