// main.dart

import 'package:flutter/material.dart';

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

import 'package:provider/provider.dart';

import 'firebase\_options.dart';

import 'providers/auth\_provider.dart';

import 'providers/note\_provider.dart';

import 'screens/auth\_screen.dart';

import 'screens/home\_screen.dart';

void main() async {

WidgetsFlutterBinding.ensureInitialized();

await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform);

runApp(const MyApp());

}

class MyApp extends StatelessWidget {

const MyApp({super.key});

@override

Widget build(BuildContext context) {

return MultiProvider(

providers: [

ChangeNotifierProvider(create: (context) => AuthProvider()),

ChangeNotifierProvider(create: (context) => NoteProvider()),

],

child: MaterialApp(

debugShowCheckedModeBanner: false,

title: 'Note App',

theme: ThemeData(primarySwatch: Colors.blue),

home: Consumer<AuthProvider>(

builder: (context, authProvider, \_) {

return authProvider.isAuthenticated ? HomeScreen() : AuthScreen();

},

),

),

);

}

}

// auth\_provider.dart

import 'package:flutter/material.dart';

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

class AuthProvider with ChangeNotifier {

final FirebaseAuth \_auth = FirebaseAuth.instance;

User? \_user;

User? get user => \_user;

bool get isAuthenticated => \_user != null;

AuthProvider() {

\_auth.authStateChanges().listen((user) {

\_user = user;

notifyListeners();

});

}

Future<void> signIn(String email, String password) async {

await \_auth.signInWithEmailAndPassword(email: email, password: password);

}

Future<void> signUp(String email, String password) async {

await \_auth.createUserWithEmailAndPassword(email: email, password: password);

}

Future<void> signOut() async {

await \_auth.signOut();

}

}

// note\_provider.dart

import 'package:flutter/material.dart';

import '../models/note\_model.dart';

import '../services/firebase\_service.dart';

class NoteProvider with ChangeNotifier {

final FirebaseService \_firebaseService = FirebaseService();

List<Note> \_notes = [];

List<Note> get notes => \_notes;

void fetchNotes() {

\_firebaseService.getNotes().listen((notes) {

\_notes = notes;

notifyListeners();

});

}

Future<void> addNote(Note note) async {

await \_firebaseService.addNote(note);

}

Future<void> updateNote(Note note) async {

await \_firebaseService.updateNote(note);

}

Future<void> deleteNote(String noteId) async {

await \_firebaseService.deleteNote(noteId);

}

}

// note\_model.dart

class Note {

final String id;

final String title;

final String content;

final DateTime createdAt;

Note({required this.id, required this.title, required this.content, required this.createdAt});

Map<String, dynamic> toMap() {

return {'id': id, 'title': title, 'content': content, 'createdAt': createdAt.millisecondsSinceEpoch};

}

factory Note.fromMap(Map<String, dynamic> map) {

return Note(

id: map['id'],

title: map['title'],

content: map['content'],

createdAt: DateTime.fromMillisecondsSinceEpoch(map['createdAt']),

);

}

}

// firebase\_service.dart

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

import '../models/note\_model.dart';

class FirebaseService {

final CollectionReference notesCollection = FirebaseFirestore.instance.collection('notes');

Future<void> addNote(Note note) async {

await notesCollection.doc(note.id).set(note.toMap());

}

Future<void> updateNote(Note note) async {

await notesCollection.doc(note.id).update(note.toMap());

}

Future<void> deleteNote(String noteId) async {

await notesCollection.doc(noteId).delete();

}

Stream<List<Note>> getNotes() {

return notesCollection.snapshots().map((snapshot) {

return snapshot.docs.map((doc) => Note.fromMap(doc.data() as Map<String, dynamic>)).toList();

});

}

}

// auth\_screen.dart

import 'package:flutter/material.dart';

import 'package:provider/provider.dart';

import '../providers/auth\_provider.dart';

class AuthScreen extends StatelessWidget {

final TextEditingController emailController = TextEditingController();

final TextEditingController passwordController = TextEditingController();

@override

Widget build(BuildContext context) {

final authProvider = Provider.of<AuthProvider>(context);

return Scaffold(

appBar: AppBar(title: Text("Login / Signup")),

body: Column(

children: [

TextField(controller: emailController, decoration: InputDecoration(labelText: "Email")),

TextField(controller: passwordController, decoration: InputDecoration(labelText: "Password"), obscureText: true),

ElevatedButton(onPressed: () => authProvider.signIn(emailController.text, passwordController.text), child: Text("Login")),

TextButton(onPressed: () => authProvider.signUp(emailController.text, passwordController.text), child: Text("Signup")),

],

),

);

}

}