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

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

import 'user\_model.dart';

class UserService {

final FirebaseFirestore \_firestore = FirebaseFirestore.instance;

final FirebaseAuth \_auth = FirebaseAuth.instance;

// Fetch user profile

Future<UserModel?> getUserProfile(String uid) async {

try {

var userDoc = await \_firestore.collection('users').doc(uid).get();

if (userDoc.exists) {

return UserModel.fromMap(userDoc.data()!);

}

} catch (e) {

print("Error fetching user profile: $e");

}

return null;

}

// Update user profile

Future<void> updateUserProfile(UserModel user) async {

try {

await \_firestore.collection('users').doc(user.uid).update(user.toMap());

} catch (e) {

print("Error updating profile: $e");

}

}

}

import 'package:flutter/material.dart';

import 'package:image\_picker/image\_picker.dart';

import 'dart:io';

import 'user\_service.dart';

import 'user\_model.dart';

class ProfileScreen extends StatefulWidget {

final String uid;

ProfileScreen({required this.uid});

@override

\_ProfileScreenState createState() => \_ProfileScreenState();

}

class \_ProfileScreenState extends State<ProfileScreen> {

final UserService \_userService = UserService();

UserModel? \_user;

File? \_image;

final picker = ImagePicker();

@override

void initState() {

super.initState();

\_loadUserProfile();

}

void \_loadUserProfile() async {

var userData = await \_userService.getUserProfile(widget.uid);

if (userData != null) {

setState(() {

\_user = userData;

});

}

}

Future<void> \_pickImage() async {

final pickedFile = await picker.pickImage(source: ImageSource.gallery);

if (pickedFile != null) {

setState(() {

\_image = File(pickedFile.path);

});

// Upload image to Firebase Storage (to be implemented)

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text("User Profile")),

body: \_user == null

? Center(child: CircularProgressIndicator())

: Padding(

padding: const EdgeInsets.all(16.0),

child: Column(

children: [

GestureDetector(

onTap: \_pickImage,

child: CircleAvatar(

radius: 50,

backgroundImage: \_image != null

? FileImage(\_image!)

: NetworkImage(\_user!.profilePic) as ImageProvider,

),

),

SizedBox(height: 20),

TextField(

controller: TextEditingController(text: \_user!.name),

decoration: InputDecoration(labelText: "Name"),

onChanged: (value) {

\_user = \_user!.copyWith(name: value);

},

),

SizedBox(height: 20),

ElevatedButton(

onPressed: () async {

await \_userService.updateUserProfile(\_user!);

ScaffoldMessenger.of(context).showSnackBar(

SnackBar(content: Text("Profile updated!")),

);

},

child: Text("Save Changes"),

),

],

),

),

);

}

}

dependencies:

flutter:

sdk: flutter

firebase\_core: latest\_version

firebase\_auth: latest\_version

cloud\_firestore: latest\_version

provider: latest\_version

flutter\_stripe: latest\_version # For online payments

uuid: latest\_version # For generating unique invoice IDs

class PricingModel {

String serviceId;

String serviceName;

double baseVisitCharge;

double serviceCharge;

double distanceCharge;

double nightCharge;

bool isHourlyRate;

double hourlyRate;

PricingModel({

required this.serviceId,

required this.serviceName,

required this.baseVisitCharge,

required this.serviceCharge,

required this.distanceCharge,

required this.nightCharge,

required this.isHourlyRate,

required this.hourlyRate,

});

factory PricingModel.fromMap(Map<String, dynamic> data) {

return PricingModel(

serviceId: data['serviceId'],

serviceName: data['serviceName'],

baseVisitCharge: data['baseVisitCharge'].toDouble(),

serviceCharge: data['serviceCharge'].toDouble(),

distanceCharge: data['distanceCharge'].toDouble(),

nightCharge: data['nightCharge'].toDouble(),

isHourlyRate: data['isHourlyRate'],

hourlyRate: data['hourlyRate'].toDouble(),

);

}

Map<String, dynamic> toMap() {

return {

'serviceId': serviceId,

'serviceName': serviceName,

'baseVisitCharge': baseVisitCharge,

'serviceCharge': serviceCharge,

'distanceCharge': distanceCharge,

'nightCharge': nightCharge,

'isHourlyRate': isHourlyRate,

'hourlyRate': hourlyRate,

};

}

}