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

import 'pricing\_model.dart';

class PaymentService {

final FirebaseFirestore \_firestore = FirebaseFirestore.instance;

// Fetch Pricing for a Service

Future<PricingModel?> getPricing(String serviceId) async {

var doc = await \_firestore.collection('pricing').doc(serviceId).get();

if (doc.exists) {

return PricingModel.fromMap(doc.data()!);

}

return null;

}

// Calculate Price

double calculatePrice(PricingModel pricing, double distance, int hours) {

double totalPrice = pricing.baseVisitCharge + pricing.serviceCharge;

totalPrice += pricing.distanceCharge \* distance;

if (pricing.isHourlyRate) {

totalPrice += pricing.hourlyRate \* hours;

}

return totalPrice;

}

}

import 'package:flutter\_stripe/flutter\_stripe.dart';

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

class StripePaymentService {

final FirebaseFirestore \_firestore = FirebaseFirestore.instance;

Future<bool> makePayment(double amount, String currency) async {

try {

PaymentIntentResult paymentIntent = await Stripe.instance.createPaymentIntent(

PaymentIntentParams(

amount: (amount \* 100).toInt(), // Convert to cents

currency: currency,

paymentMethodTypes: ['card'],

),

);

if (paymentIntent.status == PaymentIntentsStatus.Succeeded) {

print("Payment Successful!");

return true;

} else {

print("Payment Failed!");

return false;

}

} catch (e) {

print("Payment Error: $e");

return false;

}

}

}

import 'package:uuid/uuid.dart';

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

class InvoiceService {

final FirebaseFirestore \_firestore = FirebaseFirestore.instance;

final Uuid uuid = Uuid();

Future<void> generateInvoice(

String userId, String workerId, double amount, String paymentMethod) async {

String invoiceId = uuid.v4();

await \_firestore.collection('invoices').doc(invoiceId).set({

'invoiceId': invoiceId,

'userId': userId,

'workerId': workerId,

'amount': amount,

'paymentMethod': paymentMethod,

'timestamp': DateTime.now(),

});

}

}

import 'package:flutter/material.dart';

import 'payment\_service.dart';

import 'stripe\_payment\_service.dart';

import 'invoice\_service.dart';

class PaymentScreen extends StatefulWidget {

final String serviceId;

final double distance;

final int hours;

PaymentScreen({required this.serviceId, required this.distance, required this.hours});

@override

\_PaymentScreenState createState() => \_PaymentScreenState();

}

class \_PaymentScreenState extends State<PaymentScreen> {

final PaymentService \_paymentService = PaymentService();

final StripePaymentService \_stripeService = StripePaymentService();

final InvoiceService \_invoiceService = InvoiceService();

double? totalPrice;

@override

void initState() {

super.initState();

\_calculatePrice();

}

void \_calculatePrice() async {

var pricing = await \_paymentService.getPricing(widget.serviceId);

if (pricing != null) {

setState(() {

totalPrice = \_paymentService.calculatePrice(pricing, widget.distance, widget.hours);

});

}

}

void \_processPayment(String method) async {

bool success = false;

if (method == "Online") {

success = await \_stripeService.makePayment(totalPrice!, "usd");

} else {

success = true; // Assume success for cash payments

}

if (success) {

await \_invoiceService.generateInvoice("user123", "worker456", totalPrice!, method);

ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text("Payment Successful!")));

Navigator.pop(context);

} else {

ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text("Payment Failed!")));

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text("Payment")),

body: totalPrice == null

? Center(child: CircularProgressIndicator())

: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Text("Total Price: \$${totalPrice!.toStringAsFixed(2)}"),

SizedBox(height: 20),

ElevatedButton(

onPressed: () => \_processPayment("Online"),

child: Text("Pay Online"),

),

ElevatedButton(

onPressed: () => \_processPayment("Cash"),

child: Text("Pay with Cash"),

),

],

),

);

}

}

class ReviewModel {

String reviewId;

String reviewerId;

String reviewedUserId;

double rating;

String comment;

DateTime timestamp;

ReviewModel({

required this.reviewId,

required this.reviewerId,

required this.reviewedUserId,

required this.rating,

required this.comment,

required this.timestamp,

});

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

return ReviewModel(

reviewId: data['reviewId'],

reviewerId: data['reviewerId'],

reviewedUserId: data['reviewedUserId'],

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

comment: data['comment'],

timestamp: (data['timestamp'] as Timestamp).toDate(),

);

}

Map<String, dynamic> toMap() {

return {

'reviewId': reviewId,

'reviewerId': reviewerId,

'reviewedUserId': reviewedUserId,

'rating': rating,

'comment': comment,

'timestamp': timestamp,

};

}

}