// Project Structure:

//

// lib/

// ├── main.dart

// ├── app.dart

// ├── config/

// │ ├── routes.dart

// │ └── theme.dart

// ├── models/

// │ ├── user\_model.dart

// │ ├── worker\_model.dart

// │ ├── service\_model.dart

// │ ├── booking\_model.dart

// │ └── review\_model.dart

// ├── services/

// │ ├── auth\_service.dart

// │ ├── location\_service.dart

// │ ├── booking\_service.dart

// │ ├── payment\_service.dart

// │ ├── chat\_service.dart

// │ └── storage\_service.dart

// ├── views/

// │ ├── common/

// │ │ ├── splash\_screen.dart

// │ │ ├── onboarding\_screen.dart

// │ │ └── auth/

// │ │ ├── login\_screen.dart

// │ │ └── signup\_screen.dart

// │ ├── customer/

// │ │ ├── customer\_home\_screen.dart

// │ │ ├── search\_workers\_screen.dart

// │ │ ├── worker\_details\_screen.dart

// │ │ ├── booking\_screen.dart

// │ │ ├── booking\_history\_screen.dart

// │ │ ├── payment\_screen.dart

// │ │ └── profile\_screen.dart

// │ ├── worker/

// │ │ ├── worker\_home\_screen.dart

// │ │ ├── add\_service\_screen.dart

// │ │ ├── job\_requests\_screen.dart

// │ │ ├── earnings\_screen.dart

// │ │ └── worker\_profile\_screen.dart

// │ └── admin/

// │ ├── admin\_dashboard.dart

// │ ├── worker\_verification\_screen.dart

// │ ├── user\_management\_screen.dart

// │ └── complaint\_management\_screen.dart

// └── widgets/

// ├── worker\_card.dart

// ├── service\_card.dart

// ├── review\_card.dart

// ├── booking\_card.dart

// └── chat\_bubble.dart

// main.dart

import 'package:flutter/material.dart';

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

import 'app.dart';

void main() async {

WidgetsFlutterBinding.ensureInitialized();

await Firebase.initializeApp();

runApp(const SkillLinkApp());

}

// app.dart

import 'package:flutter/material.dart';

import 'package:provider/provider.dart';

import 'config/routes.dart';

import 'config/theme.dart';

import 'services/auth\_service.dart';

class SkillLinkApp extends StatelessWidget {

const SkillLinkApp({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return MultiProvider(

providers: [

ChangeNotifierProvider(create: (\_) => AuthService()),

// Add other providers here

],

child: MaterialApp(

title: 'SkillLink',

theme: appTheme,

initialRoute: AppRoutes.splash,

routes: AppRoutes.routes,

),

);

}

}

// config/theme.dart

import 'package:flutter/material.dart';

final ThemeData appTheme = ThemeData(

primaryColor: const Color(0xFF2E7D32), // Green

colorScheme: ColorScheme.fromSwatch().copyWith(

secondary: const Color(0xFF1976D2), // Blue

background: Colors.white,

),

fontFamily: 'Poppins',

textTheme: const TextTheme(

headlineLarge: TextStyle(fontSize: 28.0, fontWeight: FontWeight.bold),

headlineMedium: TextStyle(fontSize: 24.0, fontWeight: FontWeight.w600),

titleLarge: TextStyle(fontSize: 20.0, fontWeight: FontWeight.w600),

bodyLarge: TextStyle(fontSize: 16.0),

bodyMedium: TextStyle(fontSize: 14.0),

),

elevatedButtonTheme: ElevatedButtonThemeData(

style: ElevatedButton.styleFrom(

padding: const EdgeInsets.symmetric(vertical: 12.0, horizontal: 24.0),

shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(8.0)),

),

),

);

// config/routes.dart

import 'package:flutter/material.dart';

import '../views/common/splash\_screen.dart';

import '../views/common/onboarding\_screen.dart';

import '../views/common/auth/login\_screen.dart';

import '../views/common/auth/signup\_screen.dart';

import '../views/customer/customer\_home\_screen.dart';

import '../views/worker/worker\_home\_screen.dart';

import '../views/admin/admin\_dashboard.dart';

class AppRoutes {

static const String splash = '/';

static const String onboarding = '/onboarding';

static const String login = '/login';

static const String signup = '/signup';

static const String customerHome = '/customer/home';

static const String workerHome = '/worker/home';

static const String adminDashboard = '/admin/dashboard';

static Map<String, WidgetBuilder> get routes => {

splash: (context) => const SplashScreen(),

onboarding: (context) => const OnboardingScreen(),

login: (context) => const LoginScreen(),

signup: (context) => const SignupScreen(),

customerHome: (context) => const CustomerHomeScreen(),

workerHome: (context) => const WorkerHomeScreen(),

adminDashboard: (context) => const AdminDashboard(),

};

}

// models/user\_model.dart

class UserModel {

final String id;

final String name;

final String email;

final String phoneNumber;

final String profilePicture;

final String address;

final double latitude;

final double longitude;

final String userType; // 'customer', 'worker', 'admin'

final DateTime createdAt;

UserModel({

required this.id,

required this.name,

required this.email,

required this.phoneNumber,

required this.profilePicture,

required this.address,

required this.latitude,

required this.longitude,

required this.userType,

required this.createdAt,

});

factory UserModel.fromJson(Map<String, dynamic> json) {

return UserModel(

id: json['id'],

name: json['name'],

email: json['email'],

phoneNumber: json['phoneNumber'],

profilePicture: json['profilePicture'],

address: json['address'],

latitude: json['latitude'],

longitude: json['longitude'],

userType: json['userType'],

createdAt: DateTime.parse(json['createdAt']),

);

}

Map<String, dynamic> toJson() {

return {

'id': id,

'name': name,

'email': email,

'phoneNumber': phoneNumber,

'profilePicture': profilePicture,

'address': address,

'latitude': latitude,

'longitude': longitude,

'userType': userType,

'createdAt': createdAt.toIso8601String(),

};

}

}

// models/worker\_model.dart

class WorkerModel {

final String userId;

final List<String> skills;

final String bio;

final bool isVerified;

final bool isAvailable;

final double rating;

final int reviewCount;

final List<String> serviceIds;

final double hourlyRate;

WorkerModel({

required this.userId,

required this.skills,

required this.bio,

required this.isVerified,

required this.isAvailable,

required this.rating,

required this.reviewCount,

required this.serviceIds,

required this.hourlyRate,

});

factory WorkerModel.fromJson(Map<String, dynamic> json) {

return WorkerModel(

userId: json['userId'],

skills: List<String>.from(json['skills']),

bio: json['bio'],

isVerified: json['isVerified'],

isAvailable: json['isAvailable'],

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

reviewCount: json['reviewCount'],

serviceIds: List<String>.from(json['serviceIds']),

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

);

}

Map<String, dynamic> toJson() {

return {

'userId': userId,

'skills': skills,

'bio': bio,

'isVerified': isVerified,

'isAvailable': isAvailable,

'rating': rating,

'reviewCount': reviewCount,

'serviceIds': serviceIds,

'hourlyRate': hourlyRate,

};

}

}

// models/service\_model.dart

class ServiceModel {

final String id;

final String workerId;

final String title;

final String description;

final String category;

final double price;

final String priceType; // 'hourly', 'fixed'

final List<String> images;

ServiceModel({

required this.id,

required this.workerId,

required this.title,

required this.description,

required this.category,

required this.price,

required this.priceType,

required this.images,

});

factory ServiceModel.fromJson(Map<String, dynamic> json) {

return ServiceModel(

id: json['id'],

workerId: json['workerId'],

title: json['title'],

description: json['description'],

category: json['category'],

price: json['price'].toDouble(),

priceType: json['priceType'],

images: List<String>.from(json['images']),

);

}

Map<String, dynamic> toJson() {

return {

'id': id,

'workerId': workerId,

'title': title,

'description': description,

'category': category,

'price': price,

'priceType': priceType,

'images': images,

};

}

}

// models/booking\_model.dart

class BookingModel {

final String id;

final String customerId;

final String workerId;

final String serviceId;

final DateTime bookingDate;

final String timeSlot;

final String status; // 'pending', 'accepted', 'rejected', 'completed', 'cancelled'

final DateTime createdAt;

final double amount;

final String paymentStatus; // 'pending', 'completed'

final String address;

final double latitude;

final double longitude;

final String description;

BookingModel({

required this.id,

required this.customerId,

required this.workerId,

required this.serviceId,

required this.bookingDate,

required this.timeSlot,

required this.status,

required this.createdAt,

required this.amount,

required this.paymentStatus,

required this.address,

required this.latitude,

required this.longitude,

required this.description,

});

factory BookingModel.fromJson(Map<String, dynamic> json) {

return BookingModel(

id: json['id'],

customerId: json['customerId'],

workerId: json['workerId'],

serviceId: json['serviceId'],

bookingDate: DateTime.parse(json['bookingDate']),

timeSlot: json['timeSlot'],

status: json['status'],

createdAt: DateTime.parse(json['createdAt']),

amount: json['amount'].toDouble(),

paymentStatus: json['paymentStatus'],

address: json['address'],

latitude: json['latitude'],

longitude: json['longitude'],

description: json['description'],

);

}

Map<String, dynamic> toJson() {

return {

'id': id,

'customerId': customerId,

'workerId': workerId,

'serviceId': serviceId,

'bookingDate': bookingDate.toIso8601String(),

'timeSlot': timeSlot,

'status': status,

'createdAt': createdAt.toIso8601String(),

'amount': amount,

'paymentStatus': paymentStatus,

'address': address,

'latitude': latitude,

'longitude': longitude,

'description': description,

};

}

}

// models/review\_model.dart

class ReviewModel {

final String id;

final String bookingId;

final String customerId;

final String workerId;

final double rating;

final String comment;

final DateTime createdAt;

ReviewModel({

required this.id,

required this.bookingId,

required this.customerId,

required this.workerId,

required this.rating,

required this.comment,

required this.createdAt,

});

factory ReviewModel.fromJson(Map<String, dynamic> json) {

return ReviewModel(

id: json['id'],

bookingId: json['bookingId'],

customerId: json['customerId'],

workerId: json['workerId'],

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

comment: json['comment'],

createdAt: DateTime.parse(json['createdAt']),

);

}

Map<String, dynamic> toJson() {

return {

'id': id,

'bookingId': bookingId,

'customerId': customerId,

'workerId': workerId,

'rating': rating,

'comment': comment,

'createdAt': createdAt.toIso8601String(),

};

}

}

// services/auth\_service.dart

import 'package:flutter/material.dart';

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

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

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

class AuthService with ChangeNotifier {

final FirebaseAuth \_auth = FirebaseAuth.instance;

final FirebaseFirestore \_firestore = FirebaseFirestore.instance;

User? get currentUser => \_auth.currentUser;

Stream<User?> get authStateChanges => \_auth.authStateChanges();

// Sign up with email and password

Future<UserModel> signUp({

required String email,

required String password,

required String name,

required String phoneNumber,

required String userType,

required String address,

required double latitude,

required double longitude,

}) async {

try {

// Create user in Firebase Auth

UserCredential result = await \_auth.createUserWithEmailAndPassword(

email: email,

password: password

);

// Create user document in Firestore

UserModel user = UserModel(

id: result.user!.uid,

name: name,

email: email,

phoneNumber: phoneNumber,

profilePicture: '',

address: address,

latitude: latitude,

longitude: longitude,

userType: userType,

createdAt: DateTime.now(),

);

await \_firestore.collection('users').doc(result.user!.uid).set(user.toJson());

if (userType == 'worker') {

// Create worker profile

await \_firestore.collection('workers').doc(result.user!.uid).set({

'userId': result.user!.uid,

'skills': [],

'bio': '',

'isVerified': false,

'isAvailable': true,

'rating': 0.0,

'reviewCount': 0,

'serviceIds': [],

'hourlyRate': 0.0,

});

}

notifyListeners();

return user;

} catch (e) {

throw Exception('Sign up failed: $e');

}

}

// Sign in with email and password

Future<UserModel> signIn({required String email, required String password}) async {

try {

UserCredential result = await \_auth.signInWithEmailAndPassword(

email: email,

password: password

);

DocumentSnapshot doc = await \_firestore.collection('users').doc(result.user!.uid).get();

UserModel user = UserModel.fromJson(doc.data() as Map<String, dynamic>);

notifyListeners();

return user;

} catch (e) {

throw Exception('Sign in failed: $e');

}

}

// Sign out

Future<void> signOut() async {

await \_auth.signOut();

notifyListeners();

}

// Get user data

Future<UserModel> getUserData(String userId) async {

DocumentSnapshot doc = await \_firestore.collection('users').doc(userId).get();

return UserModel.fromJson(doc.data() as Map<String, dynamic>);

}

// Update user profile

Future<void> updateProfile({

required String userId,

required Map<String, dynamic> data,

}) async {

await \_firestore.collection('users').doc(userId).update(data);

notifyListeners();

}

}

// services/location\_service.dart

import 'package:geolocator/geolocator.dart';

import 'package:geocoding/geocoding.dart';

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

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

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

class LocationService {

final FirebaseFirestore \_firestore = FirebaseFirestore.instance;

// Get current location

Future<Position> getCurrentLocation() async {

bool serviceEnabled;

LocationPermission permission;

// Check if location services are enabled

serviceEnabled = await Geolocator.isLocationServiceEnabled();

if (!serviceEnabled) {

throw Exception('Location services are disabled.');

}

permission = await Geolocator.checkPermission();

if (permission == LocationPermission.denied) {

permission = await Geolocator.requestPermission();

if (permission == LocationPermission.denied) {

throw Exception('Location permissions are denied');

}

}

if (permission == LocationPermission.deniedForever) {

throw Exception('Location permissions are permanently denied');

}

// When we reach here, permissions are granted and we can

// continue accessing the position of the device.

return await Geolocator.getCurrentPosition();

}

// Get address from coordinates

Future<String> getAddressFromCoordinates(double latitude, double longitude) async {

List<Placemark> placemarks = await placemarkFromCoordinates(latitude, longitude);

Placemark place = placemarks[0];

return "${place.street}, ${place.subLocality}, ${place.locality}, ${place.postalCode}";

}

// Get nearby workers

Future<List<Map<String, dynamic>>> getNearbyWorkers({

required double latitude,

required double longitude,

required double radiusInKm,

String? category,

}) async {

// Get all workers

QuerySnapshot workersSnapshot = await \_firestore.collection('workers').get();

List<WorkerModel> workers = workersSnapshot.docs

.map((doc) => WorkerModel.fromJson(doc.data() as Map<String, dynamic>))

.where((worker) => worker.isVerified && worker.isAvailable)

.toList();

List<Map<String, dynamic>> nearbyWorkers = [];

for (var worker in workers) {

DocumentSnapshot userDoc = await \_firestore.collection('users').doc(worker.userId).get();

UserModel user = UserModel.fromJson(userDoc.data() as Map<String, dynamic>);

// Calculate distance

double distance = Geolocator.distanceBetween(

latitude,

longitude,

user.latitude,

user.longitude,

) / 1000; // Convert to km

// Check if within radius

if (distance <= radiusInKm) {

// Filter by category if provided

if (category != null) {

QuerySnapshot servicesSnapshot = await \_firestore

.collection('services')

.where('workerId', isEqualTo: worker.userId)

.where('category', isEqualTo: category)

.get();

if (servicesSnapshot.docs.isNotEmpty) {

nearbyWorkers.add({

'worker': worker.toJson(),

'user': user.toJson(),

'distance': distance,

});

}

} else {

nearbyWorkers.add({

'worker': worker.toJson(),

'user': user.toJson(),

'distance': distance,

});

}

}

}

// Sort by distance

nearbyWorkers.sort((a, b) => (a['distance'] as double).compareTo(b['distance'] as double));

return nearbyWorkers;

}

}

// views/common/splash\_screen.dart

import 'package:flutter/material.dart';

import 'package:provider/provider.dart';

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

import '../../config/routes.dart';

class SplashScreen extends StatefulWidget {

const SplashScreen({Key? key}) : super(key: key);

@override

State<SplashScreen> createState() => \_SplashScreenState();

}

class \_SplashScreenState extends State<SplashScreen> {

@override

void initState() {

super.initState();

\_checkAuthState();

}

void \_checkAuthState() async {

await Future.delayed(const Duration(seconds: 2));

if (!mounted) return;

final authService = Provider.of<AuthService>(context, listen: false);

authService.authStateChanges.listen((user) {

if (user == null) {

Navigator.pushReplacementNamed(context, AppRoutes.onboarding);

} else {

\_navigateBasedOnUserType(user.uid);

}

});

}

void \_navigateBasedOnUserType(String uid) async {

final authService = Provider.of<AuthService>(context, listen: false);

final user = await authService.getUserData(uid);

switch (user.userType) {

case 'customer':

Navigator.pushReplacementNamed(context, AppRoutes.customerHome);

break;

case 'worker':

Navigator.pushReplacementNamed(context, AppRoutes.workerHome);

break;

case 'admin':

Navigator.pushReplacementNamed(context, AppRoutes.adminDashboard);

break;

default:

Navigator.pushReplacementNamed(context, AppRoutes.onboarding);

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Image.asset(

'assets/images/logo.png',

height: 150,

width: 150,

),

const SizedBox(height: 24),

const Text(

'SkillLink',

style: TextStyle(

fontSize: 32,

fontWeight: FontWeight.bold,

color: Color(0xFF2E7D32),

),

),

const SizedBox(height: 16),

const Text(

'Connect with skilled workers nearby',

style: TextStyle(

fontSize: 16,

color: Colors.grey,

),

),

const SizedBox(height: 40),

const CircularProgressIndicator(

valueColor: AlwaysStoppedAnimation<Color>(Color(0xFF2E7D32)),

),

],

),

),

);

}

}

// views/customer/customer\_home\_screen.dart

import 'package:flutter/material.dart';

import 'package:provider/provider.dart';

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

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

import '../../widgets/service\_card.dart';

class CustomerHomeScreen extends StatefulWidget {

const CustomerHomeScreen({Key? key}) : super(key: key);

@override

State<CustomerHomeScreen> createState() => \_CustomerHomeScreenState();

}

class \_CustomerHomeScreenState extends State<CustomerHomeScreen> {

final List<Map<String, dynamic>> \_categories = [

{'name': 'Plumbing', 'icon': Icons.plumbing},

{'name': 'Electrical', 'icon': Icons.electrical\_services},

{'name': 'Cleaning', 'icon': Icons.cleaning\_services},

{'name': 'Gardening', 'icon': Icons.yard},

{'name': 'Painting', 'icon': Icons.format\_paint},

{'name': 'Carpentry', 'icon': Icons.handyman},

{'name': 'Moving', 'icon': Icons.local\_shipping},

{'name': 'All Services', 'icon': Icons.more\_horiz},

];

String \_selectedCategory = 'All Services';

List<Map<String, dynamic>> \_nearbyWorkers = [];

bool \_isLoading = true;

String \_address = 'Locating...';

final LocationService \_locationService = LocationService();

@override

void initState() {

super.initState();

\_loadInitialData();

}

Future<void> \_loadInitialData() async {

setState(() {

\_isLoading = true;

});

try {

// Get current location

final position = await \_locationService.getCurrentLocation();

// Get address

final address = await \_locationService.getAddressFromCoordinates(

position.latitude,

position.longitude

);

// Get nearby workers

final nearbyWorkers = await \_locationService.getNearbyWorkers(

latitude: position.latitude,

longitude: position.longitude,

radiusInKm: 10,

category: \_selectedCategory == 'All Services' ? null : \_selectedCategory,

);

setState(() {

\_address = address;

\_nearbyWorkers = nearbyWorkers;

\_isLoading = false;

});

} catch (e) {

setState(() {

\_isLoading = false;

});

ScaffoldMessenger.of(context).showSnackBar(

SnackBar(content: Text('Error: $e')),

);

}

}

void \_filterByCategory(String category) {

setState(() {

\_selectedCategory = category;

});

\_loadInitialData();

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

const Text('SkillLink', style: TextStyle(fontWeight: FontWeight.bold)),

Text(

\_address,

style: const TextStyle(fontSize: 12),

overflow: TextOverflow.ellipsis,

),

],

),

actions: [

IconButton(

icon: const Icon(Icons.chat\_bubble\_outline),

onPressed: () {

// Navigate to messages screen

},

),

IconButton(

icon: const Icon(Icons.person\_outline),

onPressed: () {

// Navigate to profile screen

},

),

],

),

body: \_isLoading

? const Center(child: CircularProgressIndicator())

: SingleChildScrollView(

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

// Search bar

Padding(

padding: const EdgeInsets.all(16.0),

child: TextField(

decoration: InputDecoration(

hintText: 'Search for a service or worker',

prefixIcon: const Icon(Icons.search),

border: OutlineInputBorder(

borderRadius: BorderRadius.circular(8.0),

),

),

),

),

// Categories

SizedBox(

height: 100,

child: ListView.builder(

scrollDirection: Axis.horizontal,

itemCount: \_categories.length,

itemBuilder: (context, index) {

final category = \_categories[index];

return GestureDetector(

onTap: () => \_filterByCategory(category['name']),

child: Container(

width: 80,

margin: const EdgeInsets.only(left: 16.0),

child: Column(

children: [

Container(

height: 60,

width: 60,

decoration: BoxDecoration(

color: \_selectedCategory == category['name']

? Theme.of(context).primaryColor

: Colors.grey[200],

borderRadius: BorderRadius.circular(12),

),

child: Icon(

category['icon'],

color: \_selectedCategory == category['name']

? Colors.white

: Colors.grey[600],

size: 30,

),

),

const SizedBox(height: 8),

Text(

category['name'],

style: TextStyle(

fontWeight: \_selectedCategory == category['name']

? FontWeight.bold

: FontWeight.normal,

fontSize: 12,

),

textAlign: TextAlign.center,

),

],

),

),

);

},

),

),

// Nearby workers

Padding(

padding: const EdgeInsets.all(16.0),

child: Text(

'Nearby ${\_selectedCategory == 'All Services' ? 'Workers' : \_selectedCategory} (${\_nearbyWorkers.length})',

style: const TextStyle(

fontSize: 18,

fontWeight: FontWeight.bold,

),

),

),

\_nearbyWorkers.isEmpty

? Center(

child: Column(

children: [

const SizedBox(height: 40),

Icon(Icons.search\_off, size: 80, color: Colors.grey[400]),

const SizedBox(height: 16),

Text(

'No workers found nearby',

style: TextStyle(

fontSize: 16,

color: Colors.grey[600],

),

),

],

),

)

: ListView.builder(

shrinkWrap: true,

physics: const NeverScrollableScroll