// lib/modules/user\_management/models/user\_model.dart

class UserModel {

final String id;

final String name;

final String email;

final String phoneNumber;

final String role; // "customer", "provider", "admin"

final bool isEmailVerified;

final bool isPhoneVerified;

final List<Map<String, dynamic>> serviceHistory;

UserModel({

required this.id,

required this.name,

required this.email,

required this.phoneNumber,

required this.role,

this.isEmailVerified = false,

this.isPhoneVerified = false,

this.serviceHistory = const [],

});

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

return UserModel(

id: json['id'],

name: json['name'],

email: json['email'],

phoneNumber: json['phoneNumber'],

role: json['role'],

isEmailVerified: json['isEmailVerified'] ?? false,

isPhoneVerified: json['isPhoneVerified'] ?? false,

serviceHistory: List<Map<String, dynamic>>.from(json['serviceHistory'] ?? []),

);

}

Map<String, dynamic> toJson() {

return {

'id': id,

'name': name,

'email': email,

'phoneNumber': phoneNumber,

'role': role,

'isEmailVerified': isEmailVerified,

'isPhoneVerified': isPhoneVerified,

'serviceHistory': serviceHistory,

};

}

UserModel copyWith({

String? id,

String? name,

String? email,

String? phoneNumber,

String? role,

bool? isEmailVerified,

bool? isPhoneVerified,

List<Map<String, dynamic>>? serviceHistory,

}) {

return UserModel(

id: id ?? this.id,

name: name ?? this.name,

email: email ?? this.email,

phoneNumber: phoneNumber ?? this.phoneNumber,

role: role ?? this.role,

isEmailVerified: isEmailVerified ?? this.isEmailVerified,

isPhoneVerified: isPhoneVerified ?? this.isPhoneVerified,

serviceHistory: serviceHistory ?? this.serviceHistory,

);

}

}

// lib/modules/user\_management/services/auth\_service.dart

import 'dart:async';

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

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

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

class AuthService {

final FirebaseAuth \_auth = FirebaseAuth.instance;

final FirebaseFirestore \_firestore = FirebaseFirestore.instance;

final \_otpExpirationTime = 5; // minutes

// Stream to listen to auth state changes

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

// Get current user

User? get currentUser => \_auth.currentUser;

// Get user profile data

Future<UserModel?> getUserProfile() async {

try {

if (currentUser == null) return null;

DocumentSnapshot doc = await \_firestore

.collection('users')

.doc(currentUser!.uid)

.get();

if (doc.exists) {

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

}

return null;

} catch (e) {

print('Error getting user profile: $e');

return null;

}

}

// Register with email and password

Future<UserModel?> registerWithEmail(

String email,

String password,

String name,

String phoneNumber,

String role,

) 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,

role: role,

);

await \_firestore

.collection('users')

.doc(result.user!.uid)

.set(user.toJson());

// Send email verification

await result.user!.sendEmailVerification();

return user;

} catch (e) {

print('Error registering user: $e');

return null;

}

}

// Sign in with email and password

Future<UserModel?> signInWithEmail(String email, String password) async {

try {

await \_auth.signInWithEmailAndPassword(

email: email,

password: password

);

return await getUserProfile();

} catch (e) {

print('Error signing in: $e');

return null;

}

}

// Sign out

Future<void> signOut() async {

try {

await \_auth.signOut();

} catch (e) {

print('Error signing out: $e');

}

}

// Send email verification

Future<bool> sendEmailVerification() async {

try {

if (currentUser != null && !currentUser!.emailVerified) {

await currentUser!.sendEmailVerification();

return true;

}

return false;

} catch (e) {

print('Error sending email verification: $e');

return false;

}

}

// Verify email OTP

Future<bool> verifyEmailOTP(String otp) async {

try {

// In a real app, you would validate the OTP against one stored in your database

// For demo purposes, we'll just update the user's verification status

if (currentUser == null) return false;

await \_firestore

.collection('users')

.doc(currentUser!.uid)

.update({'isEmailVerified': true});

return true;

} catch (e) {

print('Error verifying email OTP: $e');

return false;

}

}

// Send phone verification OTP

Future<bool> sendPhoneVerificationOTP(String phoneNumber) async {

try {

if (currentUser == null) return false;

// Generate OTP

String otp = (100000 + (DateTime.now().millisecondsSinceEpoch % 900000)).toString();

// Store OTP with expiration time

await \_firestore.collection('otps').doc(phoneNumber).set({

'otp': otp,

'expiresAt': FieldValue.serverTimestamp(),

'userId': currentUser!.uid,

});

// In a real app, you would send this OTP via SMS

print('OTP for $phoneNumber: $otp'); // For demo purposes

return true;

} catch (e) {

print('Error sending phone verification OTP: $e');

return false;

}

}

// Verify phone OTP

Future<bool> verifyPhoneOTP(String phoneNumber, String otp) async {

try {

if (currentUser == null) return false;

// Get stored OTP

DocumentSnapshot otpDoc = await \_firestore

.collection('otps')

.doc(phoneNumber)

.get();

if (!otpDoc.exists) return false;

Map<String, dynamic> otpData = otpDoc.data() as Map<String, dynamic>;

// Check if OTP is expired

Timestamp expiresAt = otpData['expiresAt'];

DateTime expiryTime = expiresAt.toDate();

if (DateTime.now().isAfter(expiryTime.add(Duration(minutes: \_otpExpirationTime)))) {

return false; // OTP expired

}

// Verify OTP

if (otpData['otp'] != otp || otpData['userId'] != currentUser!.uid) {

return false; // Invalid OTP

}

// Mark phone as verified

await \_firestore

.collection('users')

.doc(currentUser!.uid)

.update({'isPhoneVerified': true});

// Delete used OTP

await \_firestore.collection('otps').doc(phoneNumber).delete();

return true;

} catch (e) {

print('Error verifying phone OTP: $e');

return false;

}

}

// Update user profile

Future<bool> updateUserProfile({

String? name,

String? email,

String? phoneNumber,

}) async {

try {

if (currentUser == null) return false;

Map<String, dynamic> updates = {};

if (name != null) updates['name'] = name;

if (email != null) {

updates['email'] = email;

updates['isEmailVerified'] = false;

await currentUser!.updateEmail(email);

await sendEmailVerification();

}

if (phoneNumber != null) {

updates['phoneNumber'] = phoneNumber;

updates['isPhoneVerified'] = false;

}

await \_firestore

.collection('users')

.doc(currentUser!.uid)

.update(updates);

return true;

} catch (e) {

print('Error updating user profile: $e');

return false;

}

}

// Add service to history

Future<bool> addServiceToHistory(Map<String, dynamic> serviceData) async {

try {

if (currentUser == null) return false;

await \_firestore

.collection('users')

.doc(currentUser!.uid)

.update({

'serviceHistory': FieldValue.arrayUnion([serviceData]),

});

return true;

} catch (e) {

print('Error adding service to history: $e');

return false;

}

}

}

// lib/modules/user\_management/providers/auth\_provider.dart

import 'package:flutter/material.dart';

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

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

class AuthProvider extends ChangeNotifier {

final AuthService \_authService = AuthService();

UserModel? \_currentUser;

bool \_isLoading = false;

String? \_error;

UserModel? get currentUser => \_currentUser;

bool get isLoading => \_isLoading;

String? get error => \_error;

bool get isAuthenticated => \_currentUser != null;

AuthProvider() {

\_init();

}

void \_init() {

\_authService.authStateChanges.listen((user) async {

if (user != null) {

\_isLoading = true;

notifyListeners();

\_currentUser = await \_authService.getUserProfile();

\_isLoading = false;

notifyListeners();

} else {

\_currentUser = null;

notifyListeners();

}

});

}

// Register with email and password

Future<bool> register(String email, String password, String name, String phoneNumber, String role) async {

\_isLoading = true;

\_error = null;

notifyListeners();

try {

\_currentUser = await \_authService.registerWithEmail(

email,

password,

name,

phoneNumber,

role,

);

\_isLoading = false;

notifyListeners();

return \_currentUser != null;

} catch (e) {

\_error = e.toString();

\_isLoading = false;

notifyListeners();

return false;

}

}

// Sign in with email and password

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

\_isLoading = true;

\_error = null;

notifyListeners();

try {

\_currentUser = await \_authService.signInWithEmail(email, password);

\_isLoading = false;

notifyListeners();

return \_currentUser != null;

} catch (e) {

\_error = e.toString();

\_isLoading = false;

notifyListeners();

return false;

}

}

// Sign out

Future<void> signOut() async {

\_isLoading = true;

notifyListeners();

await \_authService.signOut();

\_currentUser = null;

\_isLoading = false;

notifyListeners();

}

// Send email verification OTP

Future<bool> sendEmailOTP() async {

\_isLoading = true;

\_error = null;

notifyListeners();

bool result = await \_authService.sendEmailVerification();

\_isLoading = false;

if (!result) {

\_error = 'Failed to send email verification';

}

notifyListeners();

return result;

}

// Verify email OTP

Future<bool> verifyEmailOTP(String otp) async {

\_isLoading = true;

\_error = null;

notifyListeners();

bool result = await \_authService.verifyEmailOTP(otp);

if (result && \_currentUser != null) {

\_currentUser = \_currentUser!.copyWith(isEmailVerified: true);

} else if (!result) {

\_error = 'Invalid or expired OTP';

}

\_isLoading = false;

notifyListeners();

return result;

}

// Send phone verification OTP

Future<bool> sendPhoneOTP(String phoneNumber) async {

\_isLoading = true;

\_error = null;

notifyListeners();

bool result = await \_authService.sendPhoneVerificationOTP(phoneNumber);

\_isLoading = false;

if (!result) {

\_error = 'Failed to send phone verification';

}

notifyListeners();

return result;

}

// Verify phone OTP

Future<bool> verifyPhoneOTP(String phoneNumber, String otp) async {

\_isLoading = true;

\_error = null;

notifyListeners();

bool result = await \_authService.verifyPhoneOTP(phoneNumber, otp);

if (result && \_currentUser != null) {

\_currentUser = \_currentUser!.copyWith(isPhoneVerified: true);

} else if (!result) {

\_error = 'Invalid or expired OTP';

}

\_isLoading = false;

notifyListeners();

return result;

}

// Update user profile

Future<bool> updateProfile({

String? name,

String? email,

String? phoneNumber,

}) async {

\_isLoading = true;

\_error = null;

notifyListeners();

bool result = await \_authService.updateUserProfile(

name: name,

email: email,

phoneNumber: phoneNumber,

);

if (result) {

\_currentUser = await \_authService.getUserProfile();

} else {

\_error = 'Failed to update profile';

}

\_isLoading = false;

notifyListeners();

return result;

}

// Add service to history

Future<bool> addServiceToHistory(Map<String, dynamic> serviceData) async {

bool result = await \_authService.addServiceToHistory(serviceData);

if (result) {

\_currentUser = await \_authService.getUserProfile();

notifyListeners();

}

return result;

}

}

// lib/modules/user\_management/screens/registration\_screen.dart

import 'package:flutter/material.dart';

import 'package:provider/provider.dart';

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

import 'otp\_verification\_screen.dart';

class RegistrationScreen extends StatefulWidget {

@override

\_RegistrationScreenState createState() => \_RegistrationScreenState();

}

class \_RegistrationScreenState extends State<RegistrationScreen> {

final \_formKey = GlobalKey<FormState>();

final \_nameController = TextEditingController();

final \_emailController = TextEditingController();

final \_phoneController = TextEditingController();

final \_passwordController = TextEditingController();

String \_selectedRole = 'customer';

@override

void dispose() {

\_nameController.dispose();

\_emailController.dispose();

\_phoneController.dispose();

\_passwordController.dispose();

super.dispose();

}

@override

Widget build(BuildContext context) {

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

return Scaffold(

appBar: AppBar(

title: Text('Create Account'),

),

body: SingleChildScrollView(

padding: EdgeInsets.all(16.0),

child: Form(

key: \_formKey,

child: Column(

crossAxisAlignment: CrossAxisAlignment.stretch,

children: [

TextFormField(

controller: \_nameController,

decoration: InputDecoration(

labelText: 'Full Name',

prefixIcon: Icon(Icons.person),

border: OutlineInputBorder(),

),

validator: (value) {

if (value == null || value.isEmpty) {

return 'Please enter your name';

}

return null;

},

),

SizedBox(height: 16),

TextFormField(

controller: \_emailController,

decoration: InputDecoration(

labelText: 'Email',

prefixIcon: Icon(Icons.email),

border: OutlineInputBorder(),

),

keyboardType: TextInputType.emailAddress,

validator: (value) {

if (value == null || value.isEmpty) {

return 'Please enter your email';

}

if (!RegExp(r'^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$').hasMatch(value)) {

return 'Please enter a valid email';

}

return null;

},

),

SizedBox(height: 16),

TextFormField(

controller: \_phoneController,

decoration: InputDecoration(

labelText: 'Phone Number',

prefixIcon: Icon(Icons.phone),

border: OutlineInputBorder(),

),

keyboardType: TextInputType.phone,

validator: (value) {

if (value == null || value.isEmpty) {

return 'Please enter your phone number';

}

if (!RegExp(r'^\+?[0-9]{10,15}$').hasMatch(value)) {

return 'Please enter a valid phone number';

}

return null;

},

),

SizedBox(height: 16),

TextFormField(

controller: \_passwordController,

decoration: InputDecoration(

labelText: 'Password',

prefixIcon: Icon(Icons.lock),

border: OutlineInputBorder(),

),

obscureText: true,

validator: (value) {

if (value == null || value.isEmpty) {

return 'Please enter a password';

}

if (value.length < 6) {

return 'Password must be at least 6 characters long';

}

return null;

},

),

SizedBox(height: 16),

DropdownButtonFormField<String>(

decoration: InputDecoration(

labelText: 'Role',

prefixIcon: Icon(Icons.work),

border: OutlineInputBorder(),

),

value: \_selectedRole,

items: [

DropdownMenuItem(

value: 'customer',

child: Text('Customer'),

),

DropdownMenuItem(

value: 'provider',

child: Text('Service Provider'),

),

],

onChanged: (value) {

setState(() {

\_selectedRole = value!;

});

},

),

SizedBox(height: 24),

ElevatedButton(

onPressed: authProvider.isLoading

? null

: () async {

if (\_formKey.currentState!.validate()) {

bool success = await authProvider.register(

\_emailController.text,

\_passwordController.text,

\_nameController.text,

\_phoneController.text,

\_selectedRole,

);

if (success) {

// Navigate to OTP verification

Navigator.push(

context,

MaterialPageRoute(

builder: (context) => OTPVerificationScreen(),

),

);

}

}

},

child: authProvider.isLoading

? CircularProgressIndicator()

: Text('REGISTER'),

style: ElevatedButton.styleFrom(

padding: EdgeInsets.symmetric(vertical: 16),

),

),

SizedBox(height: 8),

if (authProvider.error != null)

Text(

authProvider.error!,

style: TextStyle(color: Colors.red),

textAlign: TextAlign.center,

),

SizedBox(height: 16),

TextButton(

onPressed: () {

Navigator.pop(context);

},

child: Text('Already have an account? Sign In'),

),

],

),

),

),

);

}

}

// lib/modules/user\_management/screens/login\_screen.dart

import 'package:flutter/material.dart';

import 'package:provider/provider.dart';

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

import 'registration\_screen.dart';

import 'profile\_screen.dart';

class LoginScreen extends StatefulWidget {

@override

\_LoginScreenState createState() => \_LoginScreenState();

}

class \_LoginScreenState extends State<LoginScreen> {

final \_formKey = GlobalKey<FormState>();

final \_emailController = TextEditingController();

final \_passwordController = TextEditingController();

@override

void dispose() {

\_emailController.dispose();

\_passwordController.dispose();

super.dispose();

}

@override

Widget build(BuildContext context) {

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

return Scaffold(

appBar: AppBar(

title: Text('Sign In'),

),

body: SingleChildScrollView(

padding: EdgeInsets.all(16.0),

child: Form(

key: \_formKey,

child: Column(

crossAxisAlignment: CrossAxisAlignment.stretch,

children: [

SizedBox(height: 40),

Icon(

Icons.account\_circle,

size: 80,

color: Theme.of(context).primaryColor,

),

SizedBox(height: 40),

TextFormField(

controller: \_emailController,

decoration: InputDecoration(

labelText: 'Email',

prefixIcon: Icon(Icons.email),

border: OutlineInputBorder(),

),

keyboardType: TextInputType.emailAddress,

validator: (value) {

if (value == null || value.isEmpty) {

return 'Please enter your email';

}

return null;

},

),

SizedBox(height: 16),

TextFormField(

controller: \_passwordController,

decoration: InputDecoration(

labelText: 'Password',

prefixIcon: Icon(Icons.lock),

border: OutlineInputBorder(),

),

obscureText: true,

validator: (value) {

if (value == null || value.isEmpty) {

return 'Please enter your password';

}

return null;

},

),

SizedBox(height: 8),

Align(

alignment: Alignment.centerRight,

child: TextButton(

onPressed: () {

// Navigate to password reset screen

},

child: Text('Forgot Password?'),

),

),

SizedBox(height: 24),

ElevatedButton(

onPressed: authProvider.isLoading

? null

: () async {

if (\_formKey.currentState!.validate()) {

bool success = await authProvider.signIn(

\_emailController.text,

\_passwordController.text,

);

if (success) {

Navigator.pushReplacement(

context,

MaterialPageRoute(

builder: (context) => ProfileScreen(),

),

);

}

}

},

child: authProvider.isLoading

? CircularProgressIndicator()

: Text('SIGN IN'),

style: ElevatedButton.styleFrom(

padding: EdgeInsets.symmetric(vertical: 16),

),

),

SizedBox(height: 8),

if (authProvider.error != null)

Text(

authProvider.error!,

style: TextStyle(color: Colors.red),

textAlign: TextAlign.center,

),

SizedBox(height: 24),

TextButton(

onPressed: () {

Navigator.push(

context,

MaterialPageRoute(

builder: (context) => RegistrationScreen(),

),

);

},

child: Text('Don\'t have an account? Register'),

),

],

),

),

),

);

}

}

// lib/modules/user\_management/screens/otp\_verification\_screen.dart

import 'package:flutter/material.dart';

import 'package:provider/provider.dart';

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

import 'profile\_screen.dart';

class OTPVerificationScreen extends StatefulWidget {

@override

\_OTPVerificationScreenState createState() => \_OTPVerificationScreenState();

}

class \_OTPVerificationScreenState extends State<OTPVerificationScreen> {

final \_emailOtpController = TextEditingController();

final \_phoneOtpController = TextEditingController();

bool \_isEmailOtpSent = false;

bool \_isPhoneOtpSent = false;

bool \_isEmailVerified = false;

bool \_isPhoneVerified = false;

@override

void dispose() {

\_emailOtpController.dispose();

\_phoneOtpController.dispose();

super.dispose();

}

@override

void initState() {

super.initState();

// Check if already verified

WidgetsBinding.instance.addPostFrameCallback((\_) {

final authProvider = Provider.of<AuthProvider>(context, listen: false);

if (authProvider.currentUser != null) {

setState(() {

\_isEmailVerified = authProvider.currentUser!.isEmailVerified;

\_isPhoneVerified = authProvider.currentUser!.isPhoneVerified;

});

}

});

}

@override

Widget build(BuildContext context) {

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

final user = authProvider.currentUser;

if (user == null) {

return Scaffold(

appBar: AppBar(

title: Text('Verification'),

),

body: Center(

child: Text('Please sign in first'),

),

);

}

return Scaffold(

appBar: AppBar(

title: Text('Verify Your Account'),

),

body: SingleChildScrollView(

padding: EdgeInsets.all(16.0),

child: Column(

crossAxisAlignment: CrossAxisAlignment.stretch,

children: [

Card(

child: Padding(

padding: EdgeInsets.all(16.0),

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

Text(

'Email Verification',

style: Theme.of(context).textTheme.headline6,

),

SizedBox(height: 8),

Text(

'Verify your email: ${user.email}',

style: TextStyle(color: Colors.grey),

),

SizedBox(height: 16),

if (\_isEmailVerified)

Row(

children: [

Icon(Icons.check\_circle, color: Colors.green),

SizedBox(width: 8),

Text(

'Email verified',

style: TextStyle(color: Colors.green),

),

],

)

else if (!\_isEmailOtpSent)

ElevatedButton(

onPressed: () async {

bool success = await authProvider.sendEmailOTP();

if (success) {

setState(() {

\_isEmailOtpSent = true;

});

ScaffoldMessenger.of(context).showSnackBar(

SnackBar(

content: Text('Verification email sent. Check your inbox.'),

),

);

}

},

child: Text('SEND VERIFICATION EMAIL'),

)

else

Column(

crossAxisAlignment: CrossAxisAlignment.stretch,

children: [

TextFormField(

controller: \_emailOtpController,

decoration: InputDecoration(

labelText: 'Enter OTP from Email',

border: OutlineInputBorder(),

),

keyboardType: TextInputType.number,

),

SizedBox(height: 8),

ElevatedButton(

onPressed: () async {

bool success = await authProvider.verifyEmailOTP(

\_emailOtpController.text,

);

if (success) {

setState(() {

\_isEmailVerified = true;

});

ScaffoldMessenger.of(context).showSnackBar(

SnackBar(

content: Text('Email verified successfully'),

),

);

}

},

child: Text('VERIFY EMAIL'),

),

TextButton(

onPressed: () async {

await authProvider.sendEmailOTP();

ScaffoldMessenger.of(context).showSnackBar(

SnackBar(

content: Text('New OTP sent to your email'),

),

);

},

child: Text('Resend OTP'),

),

],

),

],

),

),

),

SizedBox(height: 16),

Card(

child: Padding(

padding: EdgeInsets.all(16.0),

child: Column