SOURCE CODE

FLUTTER CODE

1. **Main.dart**

import 'package:flutter/material.dart';

import 'screens/login\_screen.dart';

void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {

  const MyApp({super.key});

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      title: 'Complaint Response Application with Geolocator Tracking',

      debugShowCheckedModeBanner: false,

      theme: ThemeData(

        primarySwatch: Colors.blue,

        brightness: Brightness.light,

      ),

      darkTheme: ThemeData.dark(),

      home: const LoginScreen(),

    );

  }

}

1. **Services/api\_services.dart**

import 'dart:convert';

import 'package:http/http.dart' as http;

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

import '../models/complaint.dart';

class ApiService {

  static const baseUrl = 'http://192.168.1.142/cra\_api/api';

  static Future<User?> login(String email, String password) async {

    final response = await http.post(

      Uri.parse('$baseUrl/login.php'),

      body: jsonEncode({"email": email, "password": password}),

      headers: {"Content-Type": "application/json"},

    );

    final data = jsonDecode(response.body);

    if (data['success']) {

      return User.fromJson(data['user']);

    }

    return null;

  }

  static Future<bool> register(

      String name, String email, String password) async {

    final response = await http.post(

      Uri.parse('$baseUrl/register.php'),

      body: jsonEncode({"name": name, "email": email, "password": password}),

      headers: {"Content-Type": "application/json"},

    );

    return jsonDecode(response.body)['success'];

  }

  static Future<List<Complaint>> fetchUserComplaints(int userId) async {

    final response = await http

        .get(Uri.parse('$baseUrl/get\_user\_complaints.php?user\_id=$userId'));

    final data = jsonDecode(response.body);

    return (data['complaints'] as List)

        .map((e) => Complaint.fromJson(e))

        .toList();

  }

  static Future<List<Complaint>> getAllComplaints() async {

    final response =

        await http.get(Uri.parse('$baseUrl/get\_all\_complaint.php'));

    if (response.statusCode == 200) {

      final data = jsonDecode(response.body);

      if (data['success']) {

        List complaintsJson = data['data'];

        return complaintsJson.map((e) => Complaint.fromJson(e)).toList();

      } else {

        throw Exception("Failed to load complaints");

      }

    } else {

      throw Exception("Server error");

    }

  }

  static Future<bool> createComplaint(Map<String, dynamic> complaint) async {

    final response = await http.post(

      Uri.parse('$baseUrl/create\_complaint.php'),

      body: jsonEncode(complaint),

      headers: {"Content-Type": "application/json"},

    );

    return jsonDecode(response.body)['success'];

  }

  static Future<bool> updateComplaintStatus(int id, String status) async {

    final response = await http.post(

      Uri.parse('$baseUrl/update\_complaint\_status.php'),

      body: jsonEncode({"id": id, "status": status}),

      headers: {"Content-Type": "application/json"},

    );

    return jsonDecode(response.body)['success'];

  }

}

1. **Screens/add\_complaint\_list.dart**

import 'package:flutter/material.dart';

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

import 'package:latlong2/latlong.dart';

import 'package:url\_launcher/url\_launcher.dart';

import 'package:geolocator/geolocator.dart';

import '../models/complaint.dart';

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

import '../helpers/location\_helper.dart';

class AdminComplaintList extends StatefulWidget {

  const AdminComplaintList({super.key});

  @override

  State<AdminComplaintList> createState() => \_AdminComplaintListState();

}

class \_AdminComplaintListState extends State<AdminComplaintList> {

  late Future<List<Complaint>> \_futureComplaints;

  Position? \_adminPosition;

  final Color primaryColor = const Color(0xFFFF8774);

  @override

  void initState() {

    super.initState();

    \_loadComplaints();

    \_getAdminLocation();

  }

  void \_loadComplaints() {

    \_futureComplaints = ApiService.getAllComplaints();

  }

  Future<void> \_getAdminLocation() async {

    try {

      \_adminPosition = await Geolocator.getCurrentPosition(

          desiredAccuracy: LocationAccuracy.high);

      setState(() {});

    } catch (e) {

      print("Location error: $e");

    }

  }

  Future<void> \_refreshComplaints() async {

    setState(() {

      \_loadComplaints();

    });

  }

  Future<void> \_markAsResolved(int complaintId) async {

    final success =

        await ApiService.updateComplaintStatus(complaintId, 'resolved');

    ScaffoldMessenger.of(context).showSnackBar(

      SnackBar(

        content: Text(success

            ? "Marked as resolved"

            : "Failed to update complaint status"),

      ),

    );

    if (success) \_refreshComplaints();

  }

  Widget \_buildComplaintCard(Complaint c) {

    return FutureBuilder<String>(

      future: LocationHelper.getAddress(c.latitude, c.longitude, c.id),

      builder: (context, snapshot) {

        final address = snapshot.data ?? 'Resolving location...';

        double? distanceInKm;

        if (\_adminPosition != null) {

          distanceInKm = Geolocator.distanceBetween(\_adminPosition!.latitude,

                  \_adminPosition!.longitude, c.latitude, c.longitude) /

              1000;

        }

        return Card(

          margin: const EdgeInsets.only(bottom: 16),

          elevation: 3,

          shape:

              RoundedRectangleBorder(borderRadius: BorderRadius.circular(12)),

          child: Padding(

            padding: const EdgeInsets.all(16),

            child: Column(

              crossAxisAlignment: CrossAxisAlignment.start,

              children: [

                Text(

                  c.title,

                  style: const TextStyle(

                    fontSize: 20,

                    fontWeight: FontWeight.bold,

                  ),

                ),

                const SizedBox(height: 6),

                Text(

                  c.description,

                  style: const TextStyle(fontSize: 16),

                ),

                const SizedBox(height: 8),

                Text(" $address",

                    style: const TextStyle(color: Colors.black87)),

                if (distanceInKm != null)

                  Padding(

                    padding: const EdgeInsets.only(top: 4),

                    child: Text(

                      " Distance: ${distanceInKm.toStringAsFixed(2)} km",

                      style:

                          const TextStyle(color: Colors.grey, fontSize: 14.5),

                    ),

                  ),

                const SizedBox(height: 10),

                const SizedBox(height: 12),

                ClipRRect(

                  borderRadius: BorderRadius.circular(8),

                  child: SizedBox(

                    height: 160,

                    child: FlutterMap(

                      options: MapOptions(

                        center: LatLng(c.latitude, c.longitude),

                        zoom: 15.0,

                        interactiveFlags: InteractiveFlag.none,

                      ),

                      children: [

                        TileLayer(

                          urlTemplate:

                              'https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png',

                          subdomains: ['a', 'b', 'c'],

                        ),

                        MarkerLayer(

                          markers: [

                            Marker(

                              point: LatLng(c.latitude, c.longitude),

                              width: 40,

                              height: 40,

                              child: const Icon(Icons.location\_pin,

                                  size: 40, color: Colors.red),

                            ),

                          ],

                        ),

                      ],

                    ),

                  ),

                ),

                const SizedBox(height: 12),

                Row(

                  mainAxisAlignment: MainAxisAlignment.spaceBetween,

                  children: [

                    Text(

                      "Status: ${c.status}",

                      style: TextStyle(

                        fontWeight: FontWeight.w600,

                        color: c.status == 'resolved'

                            ? Colors.green

                            : Colors.deepOrange,

                      ),

                    ),

                    Row(

                      children: [

                        IconButton(

                          icon: const Icon(Icons.directions),

                          tooltip: "Open in Google Maps",

                          onPressed: () async {

                            final uri = Uri.parse(

                                "https://www.google.com/maps/search/?api=1&query=${c.latitude},${c.longitude}");

                            if (await canLaunchUrl(uri)) {

                              await launchUrl(uri);

                            } else {

                              ScaffoldMessenger.of(context).showSnackBar(

                                const SnackBar(

                                    content: Text("Could not open map.")),

                              );

                            }

                          },

                        ),

                        if (c.status != 'resolved')

                          ElevatedButton.icon(

                            style: ElevatedButton.styleFrom(

                              backgroundColor: primaryColor,

                              foregroundColor: Colors.white,

                              padding: const EdgeInsets.symmetric(

                                  horizontal: 12, vertical: 8),

                              shape: RoundedRectangleBorder(

                                  borderRadius: BorderRadius.circular(8)),

                            ),

                            icon: const Icon(Icons.check\_circle\_outline),

                            label: const Text("Resolve"),

                            onPressed: () => \_markAsResolved(c.id),

                          ),

                      ],

                    ),

                  ],

                ),

              ],

            ),

          ),

        );

      },

    );

  }

  Widget \_buildComplaintList(List<Complaint> complaints) {

    return RefreshIndicator(

      onRefresh: \_refreshComplaints,

      child: ListView.builder(

        padding: const EdgeInsets.all(16),

        itemCount: complaints.length,

        itemBuilder: (context, index) => \_buildComplaintCard(complaints[index]),

      ),

    );

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      backgroundColor: const Color(0xFFFDFDFD),

      appBar: AppBar(

        title: const Text("Admin Complaints List"),

        backgroundColor: primaryColor,

        elevation: 0,

      ),

      body: FutureBuilder<List<Complaint>>(

        future: \_futureComplaints,

        builder: (context, snapshot) {

          if (snapshot.connectionState == ConnectionState.waiting) {

            return const Center(child: CircularProgressIndicator());

          } else if (snapshot.hasError) {

            return Center(child: Text("Error: ${snapshot.error}"));

          } else if (snapshot.data?.isEmpty ?? true) {

            return const Center(child: Text(" No complaints found."));

          }

          return \_buildComplaintList(snapshot.data!);

        },

      ),

    );

  }

}

1. Screens/complaint\_form\_screen.dart

import 'dart:convert';

import 'dart:io';

import 'package:flutter/material.dart';

import 'package:geolocator/geolocator.dart';

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

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

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

class ComplaintFormScreen extends StatefulWidget {

  final User user;

  const ComplaintFormScreen({super.key, required this.user});

  @override

  State<ComplaintFormScreen> createState() => \_ComplaintFormScreenState();

}

class \_ComplaintFormScreenState extends State<ComplaintFormScreen> {

  final titleController = TextEditingController();

  final descriptionController = TextEditingController();

  double? lat;

  double? lng;

  File? \_imageFile;

  bool isSubmitting = false;

  @override

  void initState() {

    super.initState();

    fetchLocation();

  }

  Future<void> fetchLocation() async {

    bool serviceEnabled = await Geolocator.isLocationServiceEnabled();

    if (!serviceEnabled) {

      showError("Please enable location services.");

      return;

    }

    LocationPermission permission = await Geolocator.checkPermission();

    if (permission == LocationPermission.denied) {

      permission = await Geolocator.requestPermission();

      if (permission == LocationPermission.denied) {

        showError("Location permission denied.");

        return;

      }

    }

    if (permission == LocationPermission.deniedForever) {

      showError(

        "Location permission permanently denied. Please enable in settings.",

      );

      return;

    }

    try {

      Position position = await Geolocator.getCurrentPosition(

        desiredAccuracy: LocationAccuracy.high,

      );

      setState(() {

        lat = position.latitude;

        lng = position.longitude;

      });

    } catch (e) {

      showError("Error fetching location.");

    }

  }

  void showError(String message) {

    ScaffoldMessenger.of(context)

        .showSnackBar(SnackBar(content: Text(message)));

  }

  Future<void> pickImage() async {

    final picked = await ImagePicker().pickImage(source: ImageSource.camera);

    if (picked != null) {

      setState(() => \_imageFile = File(picked.path));

    }

  }

  Future<void> submitComplaint() async {

    if (titleController.text.isEmpty ||

        descriptionController.text.isEmpty ||

        lat == null ||

        lng == null) {

      showError("All fields and location are required.");

      return;

    }

    setState(() => isSubmitting = true);

    String photoBase64 = '';

    if (\_imageFile != null) {

      List<int> imageBytes = await \_imageFile!.readAsBytes();

      photoBase64 = base64Encode(imageBytes);

    }

    final data = {

      'user\_id': widget.user.id,

      'title': titleController.text,

      'description': descriptionController.text,

      'photo': photoBase64,

      'latitude': lat,

      'longitude': lng,

    };

    bool success = await ApiService.createComplaint(data);

    setState(() => isSubmitting = false);

    ScaffoldMessenger.of(context).showSnackBar(SnackBar(

      content:

          Text(success ? ' Complaint submitted!' : 'Submission failed.'),

    ));

    if (success) Navigator.pop(context);

  }

  @override

  Widget build(BuildContext context) {

    final themeColor = const Color(0xFFFF8774);

    return Scaffold(

      backgroundColor: const Color(0xFFF8F9FA),

      appBar: AppBar(

        title: const Text("Submit Complaint"),

        backgroundColor: themeColor,

        elevation: 2,

      ),

      body: SingleChildScrollView(

        padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 25),

        child: Column(

          crossAxisAlignment: CrossAxisAlignment.start,

          children: [

            \_buildLabel("Title"),

            \_styledInputField(

                controller: titleController, hint: "Complaint title..."),

            const SizedBox(height: 15),

            \_buildLabel("Description"),

            \_styledInputField(

              controller: descriptionController,

              hint: "Describe the complaint...",

              maxLines: 4,

            ),

            const SizedBox(height: 15),

            \_buildLabel("Photo"),

            if (\_imageFile != null) ...[

              const SizedBox(height: 10),

              ClipRRect(

                borderRadius: BorderRadius.circular(8),

                child: Image.file(\_imageFile!, height: 160, fit: BoxFit.cover),

              ),

            ],

            const SizedBox(height: 20),

            Row(

              children: [

                const Icon(Icons.location\_pin, color: Colors.redAccent),

                const SizedBox(width: 8),

                Text(

                  lat != null ? '($lat, $lng)' : 'Fetching location...',

                  style: TextStyle(color: Colors.grey[700]),

                ),

              ],

            ),

            const SizedBox(height: 30),

            Center(

              child: SizedBox(

                width: double.infinity,

                height: 48,

                child: ElevatedButton.icon(

                  icon: const Icon(Icons.send),

                  onPressed: isSubmitting ? null : submitComplaint,

                  label: isSubmitting

                      ? const SizedBox(

                          width: 24,

                          height: 24,

                          child: CircularProgressIndicator(

                              color: Colors.white, strokeWidth: 2),

                        )

                      : const Text("Submit Complaint"),

                  style: ElevatedButton.styleFrom(

                    backgroundColor: themeColor,

                    textStyle: const TextStyle(

                        fontSize: 16, fontWeight: FontWeight.bold),

                    shape: RoundedRectangleBorder(

                        borderRadius: BorderRadius.circular(14)),

                  ),

                ),

              ),

            ),

          ],

        ),

      ),

    );

  }

  Widget \_buildLabel(String text) {

    return Text(text,

        style: TextStyle(

            fontWeight: FontWeight.bold,

            fontSize: 14,

            color: Colors.grey[800]));

  }

  Widget \_styledInputField({

    required TextEditingController controller,

    required String hint,

    int maxLines = 1,

  }) {

    return TextField(

      controller: controller,

      maxLines: maxLines,

      decoration: InputDecoration(

        hintText: hint,

        filled: true,

        fillColor: Colors.white,

        contentPadding:

            const EdgeInsets.symmetric(horizontal: 14, vertical: 12),

        border: OutlineInputBorder(

          borderRadius: BorderRadius.circular(12),

          borderSide: const BorderSide(color: Colors.transparent),

        ),

        enabledBorder: OutlineInputBorder(

          borderRadius: BorderRadius.circular(12),

          borderSide: const BorderSide(color: Colors.transparent),

        ),

      ),

    );

  }

}

1. **Complaint\_location\_screen.dart**

// complaint\_location\_screen.dart

import 'package:flutter/material.dart';

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

import 'package:latlong2/latlong.dart';

class ComplaintLocationScreen extends StatelessWidget {

  final double latitude;

  final double longitude;

  final String title;

  const ComplaintLocationScreen({

    super.key,

    required this.latitude,

    required this.longitude,

    required this.title,

  });

  @override

  Widget build(BuildContext context) {

    final LatLng position = LatLng(latitude, longitude);

    return Scaffold(

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

      body: FlutterMap(

        options: MapOptions(center: position, zoom: 15.0),

        children: [

          TileLayer(

            urlTemplate: "https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",

            subdomains: ['a', 'b', 'c'],

          ),

          MarkerLayer(

            markers: [

              Marker(

                point: position,

                width: 40,

                height: 40,

                child:

                    const Icon(Icons.location\_pin, size: 40, color: Colors.red),

              ),

            ],

          ),

        ],

      ),

    );

  }

}

1. **Complaint\_map\_screen.dart**

import 'package:flutter/material.dart';

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

import 'package:latlong2/latlong.dart';

class MapScreen extends StatelessWidget {

  final double latitude;

  final double longitude;

  final String title;

  const MapScreen({

    super.key,

    required this.latitude,

    required this.longitude,

    required this.title,

  });

  @override

  Widget build(BuildContext context) {

    final LatLng position = LatLng(latitude, longitude);

    return Scaffold(

      appBar: AppBar(

        title: Text(title),

        backgroundColor: Colors.blueAccent,

      ),

      body: FlutterMap(

        options: MapOptions(

          center: position,

          zoom: 16,

        ),

        children: [

          TileLayer(

            urlTemplate: 'https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png',

            subdomains: const ['a', 'b', 'c'],

            userAgentPackageName: 'com.example.yourapp',

          ),

          MarkerLayer(

            markers: [

              Marker(

                width: 50,

                height: 50,

                point: position,

                child: const Icon(

                  Icons.location\_on,

                  size: 40,

                  color: Colors.red,

                ),

              ),

            ],

          )

        ],

      ),

    );

  }

}

1. **Screens/Dashboard\_screen.dart**

import 'package:flutter/material.dart';

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

import '../models/complaint.dart';

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

import 'complaint\_location\_screen.dart';

import 'complaint\_form\_screen.dart'; // ✅ Add this import

const Color primaryColor = Color(0xFFFF8774); // Your color template

class DashboardScreen extends StatefulWidget {

  final User user;

  const DashboardScreen({super.key, required this.user});

  @override

  State<DashboardScreen> createState() => \_DashboardScreenState();

}

class \_DashboardScreenState extends State<DashboardScreen> {

  late Future<List<Complaint>> futureComplaints;

  @override

  void initState() {

    super.initState();

    futureComplaints = ApiService.getAllComplaints();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      backgroundColor: const Color(0xFFFDFDFD),

      appBar: AppBar(

        title: Text('Dashboard - ${widget.user.name}'),

        backgroundColor: primaryColor,

      ),

      body: FutureBuilder<List<Complaint>>(

        future: futureComplaints,

        builder: (context, snapshot) {

          if (snapshot.connectionState == ConnectionState.waiting) {

            return const Center(child: CircularProgressIndicator());

          } else if (snapshot.hasError) {

            return Center(child: Text("Error: ${snapshot.error}"));

          } else if (!snapshot.hasData || snapshot.data!.isEmpty) {

            return const Center(child: Text("No complaints found."));

          }

          final complaints = snapshot.data!;

          return ListView.builder(

            itemCount: complaints.length,

            itemBuilder: (context, index) =>

                \_buildComplaintCard(context, complaints[index]),

          );

        },

      ),

      floatingActionButton: FloatingActionButton.extended(

        backgroundColor: primaryColor,

        icon: const Icon(Icons.add\_location\_alt),

        label: const Text("File Complaint"),

        onPressed: () {

          Navigator.push(

            context,

            MaterialPageRoute(

              builder: (\_) => ComplaintFormScreen(user: widget.user),

            ),

          );

        },

      ),

    );

  }

  Widget \_buildComplaintCard(BuildContext context, Complaint c) {

    return GestureDetector(

      onTap: () {

        Navigator.push(

          context,

          MaterialPageRoute(

            builder: (\_) => ComplaintLocationScreen(

              latitude: c.latitude,

              longitude: c.longitude,

              title: c.title,

            ),

          ),

        );

      },

      child: Card(

        margin: const EdgeInsets.all(10),

        elevation: 4,

        child: Padding(

          padding: const EdgeInsets.all(12),

          child: Column(

            crossAxisAlignment: CrossAxisAlignment.start,

            children: [

              Text(

                c.title,

                style: const TextStyle(

                  fontSize: 18,

                  fontWeight: FontWeight.bold,

                ),

              ),

              const SizedBox(height: 8),

              Text(c.description),

              const SizedBox(height: 6),

              Text(

                "Status: ${c.status}",

                style: TextStyle(

                  color: c.status == 'resolved' ? Colors.green : Colors.orange,

                ),

              ),

              const SizedBox(height: 10),

              const Text("📍 Tap card to view location on map"),

            ],

          ),

        ),

      ),

    );

  }

}

1. **Screens/login\_screen.dart**

import 'package:flutter/material.dart';

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

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

import 'dashboard\_screen.dart';

import 'register\_screen.dart';

import 'admin\_complaint\_list.dart';

class LoginScreen extends StatefulWidget {

  const LoginScreen({super.key});

  @override

  State<LoginScreen> createState() => \_LoginScreenState();

}

class \_LoginScreenState extends State<LoginScreen> {

  final emailController = TextEditingController();

  final passwordController = TextEditingController();

  bool isLoading = false;

  final Color primaryColor = const Color(0xFFFF8774);

  void login() async {

    setState(() => isLoading = true);

    User? user = await ApiService.login(

      emailController.text.trim(),

      passwordController.text.trim(),

    );

    setState(() => isLoading = false);

    if (user != null) {

      if (user.role == 'admin') {

        Navigator.pushReplacement(

          context,

          MaterialPageRoute(builder: (\_) => const AdminComplaintList()),

        );

      } else {

        Navigator.pushReplacement(

          context,

          MaterialPageRoute(builder: (\_) => DashboardScreen(user: user)),

        );

      }

    } else {

      ScaffoldMessenger.of(context).showSnackBar(

        const SnackBar(content: Text("Invalid credentials")),

      );

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      backgroundColor: Colors.white,

      body: Center(

        child: SingleChildScrollView(

          padding: const EdgeInsets.symmetric(horizontal: 28, vertical: 12),

          child: Column(

            mainAxisAlignment: MainAxisAlignment.center,

            children: [

              Image.asset(

                'assets/images/complaint\_vector.png',

                height: 200,

              ),

              const SizedBox(height: 12),

              Text(

                "Quick Response App",

                style: TextStyle(

                  fontSize: 26,

                  fontWeight: FontWeight.bold,

                  color: primaryColor,

                ),

              ),

              const SizedBox(height: 30),

              // Email Field

              TextField(

                controller: emailController,

                decoration: InputDecoration(

                  labelText: 'Email',

                  prefixIcon: const Icon(Icons.email),

                  border: OutlineInputBorder(

                      borderRadius: BorderRadius.circular(10)),

                ),

              ),

              const SizedBox(height: 16),

              // Password Field

              TextField(

                controller: passwordController,

                obscureText: true,

                decoration: InputDecoration(

                  labelText: 'Password',

                  prefixIcon: const Icon(Icons.lock),

                  border: OutlineInputBorder(

                      borderRadius: BorderRadius.circular(10)),

                ),

              ),

              const SizedBox(height: 24),

              // Login Button

              SizedBox(

                width: double.infinity,

                child: ElevatedButton(

                  onPressed: login,

                  style: ElevatedButton.styleFrom(

                    backgroundColor: primaryColor,

                    padding: const EdgeInsets.symmetric(vertical: 14),

                    shape: RoundedRectangleBorder(

                      borderRadius: BorderRadius.circular(10),

                    ),

                  ),

                  child: isLoading

                      ? const CircularProgressIndicator(

                          valueColor:

                              AlwaysStoppedAnimation<Color>(Colors.white),

                        )

                      : const Text(

                          "Login",

                          style: TextStyle(fontSize: 16, color: Colors.white),

                        ),

                ),

              ),

              // Register Button

              TextButton(

                onPressed: () => Navigator.push(

                  context,

                  MaterialPageRoute(builder: (\_) => const RegisterScreen()),

                ),

                child: Text(

                  "No account? Register here",

                  style: TextStyle(color: primaryColor),

                ),

              ),

            ],

          ),

        ),

      ),

    );

  }

}

1. **Screens/register\_screen.dart**

import 'package:flutter/material.dart';

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

import 'login\_screen.dart';

class RegisterScreen extends StatefulWidget {

  const RegisterScreen({super.key});

  @override

  State<RegisterScreen> createState() => \_RegisterScreenState();

}

class \_RegisterScreenState extends State<RegisterScreen> {

  final nameController = TextEditingController();

  final emailController = TextEditingController();

  final passwordController = TextEditingController();

  bool isLoading = false;

  final Color primaryColor = const Color(0xFFFF8774);

  void register() async {

    setState(() => isLoading = true);

    bool success = await ApiService.register(

      nameController.text.trim(),

      emailController.text.trim(),

      passwordController.text.trim(),

    );

    setState(() => isLoading = false);

    if (success) {

      ScaffoldMessenger.of(context).showSnackBar(

        const SnackBar(content: Text("Registration successful")),

      );

      Navigator.pushReplacement(

        context,

        MaterialPageRoute(builder: (\_) => const LoginScreen()),

      );

    } else {

      ScaffoldMessenger.of(context).showSnackBar(

        const SnackBar(content: Text("Registration failed")),

      );

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      backgroundColor: Colors.white,

      appBar: AppBar(

        title: const Text("Create Account"),

        backgroundColor: primaryColor,

        elevation: 0,

      ),

      body: Center(

        child: SingleChildScrollView(

          padding: const EdgeInsets.all(24),

          child: Column(

            crossAxisAlignment: CrossAxisAlignment.stretch,

            children: [

              Image.asset(

                'assets/images/complaint\_vector.png',

                height: 200,

              ),

              const SizedBox(height: 10),

              TextField(

                controller: nameController,

                decoration: InputDecoration(

                  labelText: 'Full Name',

                  prefixIcon: const Icon(Icons.person),

                  border: OutlineInputBorder(

                      borderRadius: BorderRadius.circular(10)),

                ),

              ),

              const SizedBox(height: 16),

              TextField(

                controller: emailController,

                decoration: InputDecoration(

                  labelText: 'Email',

                  prefixIcon: const Icon(Icons.email),

                  border: OutlineInputBorder(

                      borderRadius: BorderRadius.circular(10)),

                ),

              ),

              const SizedBox(height: 16),

              TextField(

                controller: passwordController,

                obscureText: true,

                decoration: InputDecoration(

                  labelText: 'Password',

                  prefixIcon: const Icon(Icons.lock),

                  border: OutlineInputBorder(

                      borderRadius: BorderRadius.circular(10)),

                ),

              ),

              const SizedBox(height: 24),

              ElevatedButton(

                onPressed: register,

                style: ElevatedButton.styleFrom(

                  backgroundColor: primaryColor,

                  padding: const EdgeInsets.symmetric(vertical: 14),

                  shape: RoundedRectangleBorder(

                    borderRadius: BorderRadius.circular(10),

                  ),

                ),

                child: isLoading

                    ? const CircularProgressIndicator(

                        valueColor: AlwaysStoppedAnimation<Color>(Colors.white),

                      )

                    : const Text(

                        "Register",

                        style: TextStyle(fontSize: 16, color: Colors.white),

                      ),

              ),

              TextButton(

                onPressed: () => Navigator.pushReplacement(

                  context,

                  MaterialPageRoute(builder: (\_) => const LoginScreen()),

                ),

                child: Text(

                  "Already have an account? Login",

                  style: TextStyle(color: primaryColor),

                ),

              )

            ],

          ),

        ),

      ),

    );

  }

}

1. **Screens/user\_complaint\_screen.dart**

import 'package:flutter/material.dart';

import '../models/complaint.dart';

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

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

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

class UserComplaintsScreen extends StatefulWidget {

  final User user;

  const UserComplaintsScreen({super.key, required this.user});

  @override

  State<UserComplaintsScreen> createState() => \_UserComplaintsScreenState();

}

class \_UserComplaintsScreenState extends State<UserComplaintsScreen> {

  List<Complaint> complaints = [];

  bool isLoading = true;

  bool hasError = false;

  final Color primaryColor = const Color(0xFFFF8774);

  Future<void> loadUserComplaints() async {

    try {

      final result = await ApiService.fetchUserComplaints(widget.user.id);

      setState(() {

        complaints = result;

        isLoading = false;

        hasError = false;

      });

    } catch (e) {

      setState(() {

        isLoading = false;

        hasError = true;

      });

    }

  }

  @override

  void initState() {

    super.initState();

    loadUserComplaints();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      backgroundColor: const Color(0xFFFDFDFD),

      appBar: AppBar(

        title: const Text("📋 My Complaints"),

        backgroundColor: primaryColor,

        elevation: 0,

      ),

      body: isLoading

          ? const Center(child: CircularProgressIndicator())

          : hasError

              ? const Center(child: Text("❌ Failed to load complaints"))

              : complaints.isEmpty

                  ? const Center(child: Text("🎉 You have no complaints yet."))

                  : RefreshIndicator(

                      onRefresh: loadUserComplaints,

                      child: ListView.builder(

                        padding: const EdgeInsets.all(16),

                        itemCount: complaints.length,

                        itemBuilder: (context, index) =>

                            ComplaintCard(complaint: complaints[index]),

                      ),

                    ),

    );

  }

}

**15. Models/complaint.dart**

class Complaint {

  final int id;

  final int userId;

  final String title;

  final String description;

  final String photo;

  final double latitude;

  final double longitude;

  final String status;

  final DateTime createdAt;

  Complaint({

    required this.id,

    required this.userId,

    required this.title,

    required this.description,

    required this.photo,

    required this.latitude,

    required this.longitude,

    required this.status,

    required this.createdAt,

  });

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

    final latitudeStr = json['latitude']?.toString() ?? '0.0';

    final longitudeStr = json['longitude']?.toString() ?? '0.0';

    final double latitude = double.tryParse(latitudeStr) ?? 0.0;

    final double longitude = double.tryParse(longitudeStr) ?? 0.0;

    print('[Complaint] ID: ${json['id']} → lat: $latitude, lng: $longitude');

    return Complaint(

      id: int.tryParse(json['id'].toString()) ?? 0,

      userId: int.tryParse(json['user\_id'].toString()) ?? 0,

      title: json['title'] ?? '',

      description: json['description'] ?? '',

      photo: json['photo'] ?? '',

      latitude: latitude,

      longitude: longitude,

      status: json['status'] ?? 'pending',

      createdAt:

          DateTime.tryParse(json['created\_at'].toString()) ?? DateTime.now(),

    );

  }

  Map<String, dynamic> toJson() {

    return {

      'id': id,

      'user\_id': userId,

      'title': title,

      'description': description,

      'photo': photo,

      'latitude': latitude,

      'longitude': longitude,

      'status': status,

      'created\_at': createdAt.toIso8601String(),

    };

  }

}

**16. Models/user.dart**

class User {

  final int id;

  final String name;

  final String email;

  final String role;

  User(

      {required this.id,

      required this.name,

      required this.email,

      required this.role});

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

    return User(

      id: int.parse(json['id'].toString()),

      name: json['name'],

      email: json['email'],

      role: json['role'],

    );

  }

}

**17. Helpers/location\_helpers.dart**

import 'package:geocoding/geocoding.dart';

class LocationHelper {

  static final Map<int, String> \_addressCache = {};

  static Future<String> getAddress(double lat, double lng, int id) async {

    if (\_addressCache.containsKey(id)) return \_addressCache[id]!;

    if (lat == 0.0 && lng == 0.0) {

      print("Invalid coordinates for complaint ID $id");

      return 'Unknown location';

    }

    try {

      List<Placemark> placemarks = await placemarkFromCoordinates(lat, lng);

      if (placemarks.isNotEmpty) {

        final place = placemarks.first;

        final address =

            '${place.street ?? ''}, ${place.locality ?? ''}, ${place.administrativeArea ?? ''}';

        \_addressCache[id] = address;

        return address;

      }

    } catch (e) {

      print("Geocoding error for ID $id: $e");

    }

    return 'Unknown location';

  }

}

**18 . Widgets/addmin\_complaint\_list.dart**

import 'dart:typed\_data';

import 'package:flutter/material.dart';

import 'package:url\_launcher/url\_launcher.dart';

import '../models/complaint.dart';

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

import '../helpers/image\_helper.dart';

import '../helpers/location\_helper.dart';

class AdminComplaintList extends StatefulWidget {

  const AdminComplaintList({super.key});

  @override

  State<AdminComplaintList> createState() => \_AdminComplaintListState();

}

class \_AdminComplaintListState extends State<AdminComplaintList> {

  late Future<List<Complaint>> futureComplaints;

  @override

  void initState() {

    super.initState();

    futureComplaints = ApiService.getAllComplaints();

  }

  Future<void> \_refreshComplaints() async {

    setState(() {

      futureComplaints = ApiService.getAllComplaints();

    });

  }

  Future<void> \_markAsResolved(int complaintId) async {

    final success =

        await ApiService.updateComplaintStatus(complaintId, 'resolved');

    final snackBar = SnackBar(

      content: Text(success ? "Marked as resolved" : "Failed to update status"),

    );

    ScaffoldMessenger.of(context).showSnackBar(snackBar);

    if (success) \_refreshComplaints();

  }

  void \_openMap(double lat, double lng) async {

    final googleMapsUrl =

        'https://www.google.com/maps/search/?api=1&query=$lat,$lng';

    final uri = Uri.parse(googleMapsUrl);

    if (await canLaunchUrl(uri)) {

      await launchUrl(uri, mode: LaunchMode.externalApplication);

    } else {

      ScaffoldMessenger.of(context).showSnackBar(

        const SnackBar(content: Text("Could not open map")),

      );

    }

  }

  Widget \_buildComplaintCard(Complaint c) {

    return FutureBuilder<String>(

      future: LocationHelper.getAddress(c.latitude, c.longitude, c.id),

      builder: (context, locSnap) {

        final address = locSnap.data ?? 'Resolving location...';

        final Uint8List? photoBytes = ImageHelper.tryDecodeImage(c.photo);

        return InkWell(

          onTap: () => \_openMap(c.latitude, c.longitude),

          child: Card(

            margin: const EdgeInsets.only(bottom: 16),

            child: Padding(

              padding: const EdgeInsets.all(14),

              child: Column(

                crossAxisAlignment: CrossAxisAlignment.start,

                children: [

                  Text(c.title,

                      style: const TextStyle(

                          fontSize: 18, fontWeight: FontWeight.bold)),

                  const SizedBox(height: 6),

                  Text(c.description),

                  const SizedBox(height: 10),

                  Text("📍 $address"),

                  const SizedBox(height: 10),

                  ClipRRect(

                    borderRadius: BorderRadius.circular(8),

                    child: photoBytes != null

                        ? Image.memory(

                            photoBytes,

                            height: 200,

                            width: double.infinity,

                            fit: BoxFit.cover,

                          )

                        : Container(

                            height: 200,

                            color: Colors.grey[300],

                            child: const Center(

                              child: Icon(Icons.broken\_image,

                                  size: 50, color: Colors.grey),

                            ),

                          ),

                  ),

                  const SizedBox(height: 12),

                  Row(

                    mainAxisAlignment: MainAxisAlignment.spaceBetween,

                    children: [

                      Text("Status: ${c.status}",

                          style: TextStyle(

                            fontWeight: FontWeight.w500,

                            color: c.status == 'resolved'

                                ? Colors.green

                                : Colors.orange,

                          )),

                      if (c.status != 'resolved')

                        ElevatedButton.icon(

                          icon: const Icon(Icons.check),

                          label: const Text("Resolve"),

                          onPressed: () => \_markAsResolved(c.id),

                        ),

                    ],

                  )

                ],

              ),

            ),

          ),

        );

      },

    );

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: const Text("Admin Complaints")),

      body: FutureBuilder<List<Complaint>>(

        future: futureComplaints,

        builder: (context, snapshot) {

          if (snapshot.connectionState == ConnectionState.waiting) {

            return const Center(child: CircularProgressIndicator());

          } else if (snapshot.hasError) {

            return Center(child: Text("Error: ${snapshot.error}"));

          } else if (!snapshot.hasData || snapshot.data!.isEmpty) {

            return const Center(child: Text("No complaints found."));

          }

          final complaints = snapshot.data!;

          return RefreshIndicator(

            onRefresh: \_refreshComplaints,

            child: ListView.builder(

              padding: const EdgeInsets.all(10),

              itemCount: complaints.length,

              itemBuilder: (context, index) =>

                  \_buildComplaintCard(complaints[index]),

            ),

          );

        },

      ),

    );

  }

}

**19 . Widgets/complaint\_card.dart**

import 'package:flutter/material.dart';

import '../models/complaint.dart';

class ComplaintCard extends StatelessWidget {

  final Complaint complaint;

  final VoidCallback? onUpdate;

  final bool isAdmin;

  const ComplaintCard({

    super.key,

    required this.complaint,

    this.onUpdate,

    this.isAdmin = false,

  });

  @override

  Widget build(BuildContext context) {

    return Card(

      margin: const EdgeInsets.symmetric(vertical: 8, horizontal: 16),

      elevation: 3,

      child: Padding(

        padding: const EdgeInsets.all(12),

        child: Column(

          crossAxisAlignment: CrossAxisAlignment.start,

          children: [

            Text(complaint.title,

                style:

                    const TextStyle(fontSize: 18, fontWeight: FontWeight.bold)),

            Text(complaint.description),

            Text("Status: ${complaint.status}"),

            const SizedBox(height: 6),

            Text("Location: (${complaint.latitude}, ${complaint.longitude})"),

            if (isAdmin) ...[

              const SizedBox(height: 10),

              Row(

                children: [

                  ElevatedButton(

                    onPressed: () => onUpdate?.call(),

                    child: const Text("Mark Resolved"),

                  )

                ],

              )

            ]

          ],

        ),

      ),

    );

  }

}

**BACKEND PHP, MYSQL CODE**

1. **Config/database.php**

<?php

class Database {

    private $host = 'localhost';

    private $db\_name = 'cra\_db';

    private $username = 'root';

    private $password = '';

    public $conn;

    public function connect() {

        $this->conn = null;

        try {

            $this->conn = new PDO("mysql:host={$this->host};dbname={$this->db\_name}", $this->username, $this->password);

            $this->conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

        } catch (PDOException $e) {

            echo "Connection error: " . $e->getMessage();

        }

        return $this->conn;

    }

}

1. **Api/login.php**

<?php

require\_once '../config/Database.php';

require\_once '../controllers/AuthController.php';

$db = (new Database())->connect();

$controller = new AuthController($db);

$data = json\_decode(file\_get\_contents("php://input"), true);

$user = $controller->login($data);

if ($user) {

    echo json\_encode(['success' => true, 'user' => $user]);

} else {

    echo json\_encode(['success' => false]);

}

1. **Api/register.php**

<?php

require\_once '../config/Database.php';

require\_once '../controllers/AuthController.php';

$db = (new Database())->connect();

$controller = new AuthController($db);

$data = json\_decode(file\_get\_contents("php://input"), true);

if ($controller->register($data)) {

    echo json\_encode(['success' => true]);

} else {

    echo json\_encode(['success' => false]);

}

1. **Api/create\_complaint.php**
2. <?php
3. require\_once '../config/Database.php';
4. require\_once '../controllers/ComplaintController.php';
5. $db = (new Database())->connect();
6. $controller = new ComplaintController($db);
7. $data = json\_decode(file\_get\_contents("php://input"), true);
8. if ($controller->create($data)) {
9. echo json\_encode(['success' => true]);
10. } else {
11. echo json\_encode(['success' => false]);
12. }

**5. api/get\_all\_complaints.php**

<?php

require\_once '../config/Database.php';

require\_once '../controllers/ComplaintController.php';

$db = (new Database())->connect();

$controller = new ComplaintController($db);

$complaints = $controller->getAll();

if ($complaints) {

    echo json\_encode(['success' => true, 'data' => $complaints]);

} else {

    echo json\_encode(['success' => false, 'message' => 'No complaints found']);

}

**6. api/get\_user\_complaint.php**

<?php

require\_once '../config/Database.php';

require\_once '../controllers/ComplaintController.php';

$db = (new Database())->connect();

$controller = new ComplaintController($db);

$user\_id = $\_GET['user\_id'];

$data = $controller->getUserComplaints($user\_id);

echo json\_encode(['success' => true, 'complaints' => $data]);

// File: api/get\_all\_complaints.php

require\_once '../config/Database.php';

require\_once '../controllers/ComplaintController.php';

$db = (new Database())->connect();

$controller = new ComplaintController($db);

$data = $controller->getAll();

echo json\_encode(['success' => true, 'complaints' => $data]);

**7. api/update\_complaint.php**

<?php

require\_once '../config/Database.php';

require\_once '../controllers/ComplaintController.php';

$db = (new Database())->connect();

$controller = new ComplaintController($db);

$data = json\_decode(file\_get\_contents("php://input"), true);

if ($controller->updateStatus($data['id'], $data['status'])) {

    echo json\_encode(['success' => true]);

} else {

    echo json\_encode(['success' => false]);

}

**8. controllers/AuthControllers.php**

<?php

require\_once '../models/User.php';

class AuthController {

    private $user;

    public function \_\_construct($db) {

        $this->user = new User($db);

    }

    public function register($data) {

        return $this->user->register($data['name'], $data['email'], $data['password']);

    }

    public function login($data) {

        $user = $this->user->login($data['email']);

        if ($user && password\_verify($data['password'], $user['password'])) {

            return $user;

        }

        return false;

    }

}

**9. controllers/complaint\_controllers.php**

<?php

require\_once '../models/Complaint.php';

class ComplaintController {

    private $complaint;

    public function \_\_construct($db) {

        $this->complaint = new Complaint($db);

    }

    public function create($data) {

        return $this->complaint->create($data['user\_id'], $data['title'], $data['description'], $data['photo'], $data['latitude'], $data['longitude']);

    }

    public function getUserComplaints($user\_id) {

        return $this->complaint->getUserComplaints($user\_id);

    }

    public function getAll() {

        return $this->complaint->getAll();

    }

    public function updateStatus($id, $status) {

        return $this->complaint->updateStatus($id, $status);

    }

}

1. **Models/complaint.php**

<?php

class Complaint {

    private $conn;

    private $table = 'complaints';

    public function \_\_construct($db) {

        $this->conn = $db;

    }

    public function create($user\_id, $title, $description, $photo, $lat, $lng) {

        $stmt = $this->conn->prepare("INSERT INTO {$this->table} (user\_id, title, description, photo, latitude, longitude, status) VALUES (?, ?, ?, ?, ?, ?, 'pending')");

        return $stmt->execute([$user\_id, $title, $description, $photo, $lat, $lng]);

    }

    public function getUserComplaints($user\_id) {

        $stmt = $this->conn->prepare("SELECT \* FROM {$this->table} WHERE user\_id = ?");

        $stmt->execute([$user\_id]);

        return $stmt->fetchAll(PDO::FETCH\_ASSOC);

    }

    public function getAll() {

        $stmt = $this->conn->query("SELECT \* FROM {$this->table} ORDER BY created\_at DESC");

        return $stmt->fetchAll(PDO::FETCH\_ASSOC);

    }

    public function updateStatus($id, $status) {

        $stmt = $this->conn->prepare("UPDATE {$this->table} SET status = ? WHERE id = ?");

        return $stmt->execute([$status, $id]);

    }

}

1. **Models/user.php**

<?php

class User {

    private $conn;

    private $table = 'users';

    public function \_\_construct($db) {

        $this->conn = $db;

    }

    public function register($name, $email, $password) {

        $stmt = $this->conn->prepare("INSERT INTO {$this->table} (name, email, password, role) VALUES (?, ?, ?, 'user')");

        return $stmt->execute([$name, $email, password\_hash($password, PASSWORD\_BCRYPT)]);

    }

    public function login($email) {

        $stmt = $this->conn->prepare("SELECT \* FROM {$this->table} WHERE email = ?");

        $stmt->execute([$email]);

        return $stmt->fetch(PDO::FETCH\_ASSOC);

    }

}

**SQL DUMP FILE**

-- phpMyAdmin SQL Dump

-- version 5.2.1

-- https://www.phpmyadmin.net/

--

-- Host: 127.0.0.1

-- Generation Time: Jul 18, 2025 at 10:28 AM

-- Server version: 10.4.32-MariaDB

-- PHP Version: 8.0.30

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

START TRANSACTION;

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- Database: `cra\_db`

--

-- --------------------------------------------------------

--

-- Table structure for table `complaints`

--

CREATE TABLE `complaints` (

  `id` int(11) NOT NULL,

  `user\_id` int(11) NOT NULL,

  `title` varchar(255) NOT NULL,

  `description` text NOT NULL,

  `photo` varchar(255) DEFAULT NULL,

  `latitude` decimal(10,8) DEFAULT NULL,

  `longitude` decimal(11,8) DEFAULT NULL,

  `status` enum('pending','in\_progress','resolved') DEFAULT 'pending',

  `created\_at` timestamp NOT NULL DEFAULT current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1\_swedish\_ci;

--

-- Dumping data for table `complaints`

--

INSERT INTO `complaints` (`id`, `user\_id`, `title`, `description`, `photo`, `latitude`, `longitude`, `status`, `created\_at`) VALUES

(1, 2, 'Leak', 'Water leak', '/9j/4QGvRXhpZgAATU0AKgAAAAgABwEQAAIAAAAUAAAAYgEAAAQAAAABAAAFoAEBAAQAAAABAAAHgAEyAAIAAAAUAAAAdgESAAMAAAABAAEAAIdpAAQAAAABAAAAkQEPAAIAAAAHAAAAigAAAABzZGtfZ3Bob25lNjRfeDg2XzY0ADIwMjU6MDc6MTggMTA6NDE6MjkAR29vZ2xlAAAQgp0ABQAAAAEAAAFXgpoABQAAAAEAAAFfkpIAAgAAAAQ', 6.69429830, 124.67449500, 'resolved', '2025-07-18 02:41:33'),

(2, 3, 'Lock meter', 'lolckee', '', 6.69443530, 124.67499230, 'pending', '2025-07-18 05:28:09');

-- --------------------------------------------------------

--

-- Table structure for table `users`

--

CREATE TABLE `users` (

  `id` int(11) NOT NULL,

  `name` varchar(100) NOT NULL,

  `email` varchar(100) NOT NULL,

  `password` varchar(255) NOT NULL,

  `role` enum('user','admin') DEFAULT 'user',

  `created\_at` timestamp NOT NULL DEFAULT current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1\_swedish\_ci;

--

-- Dumping data for table `users`

--

INSERT INTO `users` (`id`, `name`, `email`, `password`, `role`, `created\_at`) VALUES

(1, 'Marchel Decena', 'admin@gmail.com', '$2y$10$jcx.jkYoFlPKhBChV0gC1.ps4ILgt7FtUVOqewXy1teVmzPdxeOtS', 'admin', '2025-07-18 01:38:49'),

(2, 'Juan Dela Cruz', 'user@gmail.com', '$2y$10$NzLqO6hA3THPqqFhOMun2.vmteqcJwzul5ygKjJWwvE5sRxAFKLYe', 'user', '2025-07-18 01:54:59'),

(3, 'user1', 'user1@gmail.com', '$2y$10$VVbRcW5l9LBLKAat3ZINMO1d1674sPS4QluOjsT8wc0FfH9daqYay', 'user', '2025-07-18 05:17:50');

--

-- Indexes for dumped tables

--

--

-- Indexes for table `complaints`

--

ALTER TABLE `complaints`

  ADD PRIMARY KEY (`id`),

  ADD KEY `user\_id` (`user\_id`);

--

-- Indexes for table `users`

--

ALTER TABLE `users`

  ADD PRIMARY KEY (`id`),

  ADD UNIQUE KEY `email` (`email`);

--

-- AUTO\_INCREMENT for dumped tables

--

--

-- AUTO\_INCREMENT for table `complaints`

--

ALTER TABLE `complaints`

  MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

--

-- AUTO\_INCREMENT for table `users`

--

ALTER TABLE `users`

  MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

--

-- Constraints for dumped tables

--

--

-- Constraints for table `complaints`

--

ALTER TABLE `complaints`

  ADD CONSTRAINT `complaints\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`) ON DELETE CASCADE;

COMMIT;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;