import 'package:cloud\_firestore/cloud\_firestore.dart';  
import 'package:flutter/material.dart';  
import 'package:flutter\_screenutil/flutter\_screenutil.dart';  
import 'package:flutter/foundation.dart';  
import 'dart:html' as html;  
  
class Claimpage extends StatefulWidget {  
 final String id;  
 const Claimpage({super.key, required this.id});  
  
 @override  
 State<Claimpage> createState() => \_ClaimpageState();  
}  
  
class \_ClaimpageState extends State<Claimpage> {  
 String? \_selectedValue;  
  
  
 TextEditingController hpnamecontroller = TextEditingController();  
 TextEditingController hpnumbercontroller = TextEditingController();  
 TextEditingController hpipnumbercontroller = TextEditingController();  
 TextEditingController firnumbercontroller = TextEditingController();  
 TextEditingController billnumbercontroller = TextEditingController();  
 TextEditingController amountcontroller = TextEditingController();  
 TextEditingController patientnamecontroller = TextEditingController();  
  
 String? DeviceId;  
 final FirebaseFirestore \_firestore = FirebaseFirestore.*instance*;  
 final GlobalKey<FormState> \_formKey = GlobalKey<FormState>();  
 final devicedetails = FirebaseFirestore.*instance*.collection("DeviceId").snapshots();  
 Uint8List? \_imageBytes; // To store selected image as bytes  
  
  
// For web, you will be using html.File  
  
 Future<void> \_pickImage() async {  
 html.FileUploadInputElement uploadInput = html.FileUploadInputElement();  
 uploadInput.accept = 'image/\*'; // Accept only images  
 uploadInput.click(); // Open file picker  
  
 uploadInput.onChange.listen((e) async {  
 final files = uploadInput.files;  
 if (files == null || files.isEmpty) return;  
  
 final html.File imageFile = files[0]; // Get selected file  
  
 final reader = html.FileReader();  
 reader.readAsArrayBuffer(imageFile); // Read file as bytes  
  
 reader.onLoadEnd.listen((e) {  
 setState(() {  
 \_imageBytes = reader.result as Uint8List; // Store image bytes  
 });  
 });  
 });  
 }  
 // Fetch user's insurance type  
 Future<void> \_fetchInsuranceType() async {  
 try {  
 DocumentSnapshot userDoc = await \_firestore.collection('Users').doc(widget.id).get();  
 if (userDoc.exists) {  
 setState(() {  
 DeviceId = userDoc['deviceId'];  
 if (kDebugMode) {  
 print("Fetched insurance type dddddddddd: $DeviceId");  
 }  
 });  
 }  
 } catch (e) {  
 if (kDebugMode) {  
 print("Error fetching insurance type: $e");  
 }  
 }  
 }  
  
 @override  
 void initState() {  
 super.initState();  
 \_fetchInsuranceType(); // Fetch the insurance type on init  
 }  
  
 @override  
 void didUpdateWidget(Claimpage oldWidget) {  
 super.didUpdateWidget(oldWidget);  
 // Fetch insurance type again if the widget id has changed (user has logged out and logged in with a new id)  
 if (widget.id != oldWidget.id) {  
 \_fetchInsuranceType();  
 }  
 }  
  
  
  
  
 Future<void> \_submitClaim() async {  
 if (\_formKey.currentState!.validate()) {  
 if (\_selectedValue == null) {  
 ScaffoldMessenger.*of*(context).showSnackBar(  
 SnackBar(content: Text("Please select a type of admission")));  
 return;  
 }  
 try {  
 String userId = widget.id;  
 DocumentSnapshot userDoc = await \_firestore.collection('Users').doc(userId).get();  
  
 if (!userDoc.exists) {  
 ScaffoldMessenger.*of*(context).showSnackBar(SnackBar(content: Text("User data not found")));  
 return;  
 }  
  
 var userData = userDoc.data() as Map<String, dynamic>;  
 var deviceQuery = await \_firestore.collection('DeviceId')  
 .where('deviceid', isEqualTo: userData['deviceId'])  
 .get();  
  
 if (deviceQuery.docs.isEmpty) {  
 ScaffoldMessenger.*of*(context).showSnackBar(SnackBar(content: Text("Device data not found")));  
 return;  
 }  
  
 var deviceData = deviceQuery.docs[0].data();  
  
 // Prepare claim data  
 var claimData = {  
 'hospitalName': hpnamecontroller.text,  
 'hospitalNumber': hpnumbercontroller.text,  
 'hospitalIpNumber': hpipnumbercontroller.text,  
 'typeOfAdmission': \_selectedValue,  
 'firNumber': firnumbercontroller.text,  
 'billNumber': billnumbercontroller.text,  
 'amount': amountcontroller.text,  
 'timestamp': FieldValue.*serverTimestamp*(),  
 'aprovelstatus': "pending",  
 };  
  
 // Add data to 'Requests' collection  
 await \_firestore.collection('Requests').doc(userId).set({  
 'Name': userData['name'],  
 'Email': userData['EmailId'],  
 'Phone': userData['phone'],  
 'vehicleNumber': userData['vehicleNumber'],  
 'regId': userData['regId'],  
 'insuranceDate': userData['insuranceDate'],  
 'gender': userData['gender'],  
 'expiryDate': userData['expiryDate'],  
 'deviceId': userData['deviceId'],  
 'address': userData['address'],  
 'InsuranceType': userData['InsuranceType'],  
  
 'hospitalName': hpnamecontroller.text,  
 'hospitalNumber': hpnumbercontroller.text,  
 'hospitalIpNumber': hpipnumbercontroller.text,  
 'typeOfAdmission': \_selectedValue,  
 'firNumber': firnumbercontroller.text,  
 'billNumber': billnumbercontroller.text,  
 'amount': amountcontroller.text,  
 'timestamp': FieldValue.*serverTimestamp*(),  
 'status': false,  
 'aprovelstatus': "pending",  
 'id': userId,  
 'reason': '',  
  
 // Device details  
 'engineNumber': deviceData['EngineNo'] ?? 'N/A',  
 'chassisNumber': deviceData['chassis\_no'] ?? 'N/A',  
 'speed': deviceData['Speed'] ?? 'N/A',  
 'accidentLocation':{deviceData['location']?['latitude'] ?? 'N/A',deviceData['location']?['longitude'] ?? 'N/A'} , // Store human-readable location  
 'engineTemperature': deviceData['EngineTemp'] ?? 'N/A',  
 'date': deviceData['date'] ?? 'N/A',  
 'accidentTime': deviceData['time'] ?? 'N/A'  
 });  
  
 await \_firestore.collection('Users').doc(userId).collection('Claim').add(claimData);  
  
 setState(() {  
 patientnamecontroller.clear();  
 hpnamecontroller.clear();  
 hpnumbercontroller.clear();  
 hpipnumbercontroller.clear();  
 firnumbercontroller.clear();  
 billnumbercontroller.clear();  
 amountcontroller.clear();  
 \_selectedValue = null;  
 });  
  
 Navigator.*of*(context).pop();  
 ScaffoldMessenger.*of*(context).showSnackBar(SnackBar(content: Text("Claim Submitted Successfully")));  
 } catch (e) {  
 ScaffoldMessenger.*of*(context).showSnackBar(SnackBar(content: Text("Error submitting claim")));  
 if (kDebugMode) {  
 print("Error submitting claim: $e");  
 }  
 }  
 }  
 }  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 backgroundColor: Colors.*white*,  
 ),  
 body: Container(  
 width: 1440.w,  
 height: 833.h,  
 decoration: BoxDecoration(image: DecorationImage(image: AssetImage("assets/claimbg.jpg"), fit: BoxFit.fill)),  
 child: Center(  
 child: Container(  
 width: 700.w,  
 height: 833.h,  
 color: Colors.*white*.withOpacity(.9),  
 child: Padding(  
 padding: EdgeInsets.only(left: 20.w),  
 child: Center(  
 child: SingleChildScrollView(  
 child: Form(  
 key: \_formKey,  
 child: Column(crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 StreamBuilder<QuerySnapshot>(  
 stream: devicedetails,  
 builder: (context, snapshot) {  
 if (snapshot.connectionState == ConnectionState.waiting) {  
 return Center(child: CircularProgressIndicator());  
 }  
  
 if (snapshot.hasError) {  
 return Center(child: Text('Error: ${snapshot.error}'));  
 }  
  
 if (!snapshot.hasData || snapshot.data!.docs.isEmpty) {  
 return Center(child: Text('No data available'));  
 }  
  
 for (var deviceData in snapshot.data!.docs) {  
 if (DeviceId == (deviceData['deviceid'] ?? '')) {  
  
 // Build UI with raw coordinate values (without asynchronous placemark conversion)  
 return Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 SizedBox(height: 20.h),  
 Center(  
 child: Text(  
 "Device Details",  
 style: TextStyle(  
 color: Colors.*black*,  
 fontSize: 22.sp,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ),  
 Text(  
 "Engine No : ${deviceData['EngineNo'] ?? 'N/A'}",  
 style: TextStyle(  
 color: Colors.*black*,  
 fontSize: 20.sp,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 Text(  
 "Chassis Number : ${deviceData['chassis\_no'] ?? 'N/A'}",  
 style: TextStyle(  
 color: Colors.*black*,  
 fontSize: 20.sp,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 Text(  
 "Speed : ${deviceData['Speed'] ?? 'N/A'}",  
 style: TextStyle(  
 color: Colors.*black*,  
 fontSize: 20.sp,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 Text(  
 "accident Location :${deviceData['location']?['latitude'] ?? 'N/A'}, ${deviceData['location']?['longitude'] ?? 'N/A'}",  
 style: TextStyle(  
 color: Colors.*black*,  
 fontSize: 20.sp,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 Text(  
 "Engine Temp : ${deviceData['EngineTemp'] ?? 'N/A'}",  
 style: TextStyle(  
 color: Colors.*black*,  
 fontSize: 20.sp,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 Text(  
 "Date : ${deviceData['date'] ?? 'N/A'}",  
 style: TextStyle(  
 color: Colors.*black*,  
 fontSize: 20.sp,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 Text(  
 "Accident Time : ${deviceData['time'] ?? 'N/A'}",  
 style: TextStyle(  
 color: Colors.*black*,  
 fontSize: 20.sp,  
 fontWeight: FontWeight.*bold*,  
 ),  
 ),  
 ],  
 );  
 }  
 }  
  
 return SizedBox(); // No matching device found  
 },  
 ),  
  
  
  
  
  
  
  
  
 Center(  
 child: Text(  
 "Medical Details",  
 style: TextStyle(color: Colors.*black*, fontSize: 20.sp, fontWeight: FontWeight.*bold*),  
 ),  
 ),  
 SizedBox(height: 20.h),  
 // Hospital Name input  
 Row(  
 children: [  
 Text(  
 "Hospital Name : ",  
 style: TextStyle(color: Colors.*black*, fontSize: 20.sp, fontWeight: FontWeight.*bold*),  
 ),  
 SizedBox(  
 width: 400.w,  
 child: TextFormField(  
 controller: hpnamecontroller,  
 decoration: InputDecoration(  
 hintText: "Hospital name ",  
 border: OutlineInputBorder(),  
 filled: true,  
 fillColor: Colors.*white*,  
 ),  
 validator: (value) {  
 if (value == null || value.isEmpty) {  
 return 'Hospital name is required';  
 }  
 return null;  
 },  
 ),  
 ),  
 ],  
 ),  
 SizedBox(height: 20.h),  
  
 // Hospital Number input  
 Row(  
 children: [  
 Text(  
 "Hospital Number : ",  
 style: TextStyle(color: Colors.*black*, fontSize: 20.sp, fontWeight: FontWeight.*bold*),  
 ),  
 SizedBox(  
 width: 400.w,  
 child: TextFormField(  
 controller: hpnumbercontroller,  
 decoration: InputDecoration(  
 hintText: "Hospital Number ",  
 border: OutlineInputBorder(),  
 filled: true,  
 fillColor: Colors.*white*,  
 ),  
 validator: (value) {  
 if (value == null || value.isEmpty) {  
 return 'Hospital number is required';  
 }  
 return null;  
 },  
 ),  
 ),  
 ],  
 ),  
 SizedBox(height: 20.h),  
  
 // Hospital IP Number input  
 Row(  
 children: [  
 Text(  
 "Hospital Ip Number : ",  
 style: TextStyle(color: Colors.*black*, fontSize: 20.sp, fontWeight: FontWeight.*bold*),  
 ),  
 SizedBox(  
 width: 400.w,  
 child: TextFormField(  
 controller: hpipnumbercontroller,  
 decoration: InputDecoration(  
 hintText: "Hospital Ip Number ",  
 border: OutlineInputBorder(),  
 filled: true,  
 fillColor: Colors.*white*,  
 ),  
 validator: (value) {  
 if (value == null || value.isEmpty) {  
 return 'Hospital IP number is required';  
 }  
 return null;  
 },  
 ),  
 ),  
 ],  
 ),  
 SizedBox(height: 20.h),  
  
 // Type of Admission dropdown  
 Row(  
 children: [  
 Text(  
 "Type of Admission : ",  
 style: TextStyle(color: Colors.*black*, fontSize: 20.sp, fontWeight: FontWeight.*bold*),  
 ),  
 DropdownButton<String>(  
 value: \_selectedValue,  
 hint: Text("Select"), // Display "Select" when empty  
 onChanged: (String? newValue) {  
 setState(() {  
 \_selectedValue = newValue;  
 });  
 },  
 items: <String>['Casualty', 'Emergency', 'OP']  
 .map<DropdownMenuItem<String>>((String value) {  
 return DropdownMenuItem<String>(  
 value: value,  
 child: SizedBox(  
 width: 350.w,  
 height: 50.h,  
 child: Text(value),  
 ),  
 );  
 }).toList(),  
 ),  
 ],  
 ),  
 SizedBox(height: 20.h),  
  
 // FIR Number input  
 Row(  
 children: [  
 Text(  
 "FIR Number : ",  
 style: TextStyle(color: Colors.*black*, fontSize: 20.sp, fontWeight: FontWeight.*bold*),  
 ),  
 SizedBox(  
 width: 400.w,  
 child: TextFormField(  
 controller: firnumbercontroller,  
 decoration: InputDecoration(  
 hintText: "FIR Number ",  
 border: OutlineInputBorder(),  
 filled: true,  
 fillColor: Colors.*white*,  
 ),  
 validator: (value) {  
 if (value == null || value.isEmpty) {  
 return 'FIR number is required';  
 }  
 return null;  
 },  
 ),  
 ),  
 ],  
 ),  
 SizedBox(height: 20.h),  
  
 // Bill Number input  
 Row(  
 children: [  
 Text(  
 "Bill Number : ",  
 style: TextStyle(color: Colors.*black*, fontSize: 20.sp, fontWeight: FontWeight.*bold*),  
 ),  
 SizedBox(  
 width: 400.w,  
 child: TextFormField(  
 controller: billnumbercontroller,  
 decoration: InputDecoration(  
 hintText: "Bill Number ",  
 border: OutlineInputBorder(),  
 filled: true,  
 fillColor: Colors.*white*,  
 ),  
 validator: (value) {  
 if (value == null || value.isEmpty) {  
 return 'Bill number is required';  
 }  
 return null;  
 },  
 ),  
 ),  
 ],  
 ),  
 SizedBox(height: 20.h),  
  
 // Amount input  
 Row(  
 children: [  
 Text(  
 "Amount : ",  
 style: TextStyle(color: Colors.*black*, fontSize: 20.sp, fontWeight: FontWeight.*bold*),  
 ),  
 SizedBox(  
 width: 400.w,  
 child: TextFormField(  
 controller: amountcontroller,  
 decoration: InputDecoration(  
 hintText: "Amount ",  
 border: OutlineInputBorder(),  
 filled: true,  
 fillColor: Colors.*white*,  
 ),  
 validator: (value) {  
 if (value == null || value.isEmpty) {  
 return 'Amount is required';  
 }  
 return null;  
 },  
 ),  
 ),  
 ],  
 ),  
 SizedBox(height: 20.h),  
  
 // Upload Image row  
 Row(  
 children: [  
 Text(  
 "Upload Image : ",  
 style: TextStyle(color: Colors.*black*, fontSize: 20.sp, fontWeight: FontWeight.*bold*),  
 ),  
  
 GestureDetector(  
 onTap: \_pickImage, // Open the gallery when tapped  
 child: Container(  
 width: 150.w,  
 height: 40.h,  
 decoration: ShapeDecoration(  
 color: Colors.*blue*,  
 shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(20.r)),  
 ),  
 child: Center(  
 child: Text(  
 "Choose Photo",  
 style: TextStyle(color: Colors.*black*, fontSize: 15.sp, fontWeight: FontWeight.*bold*),  
 ),  
 ),  
 ),  
 ),  
  
 SizedBox(width: 50.w),  
 Container(  
 width: 200.w,  
 height: 250.h,  
 color: Colors.*white10*,  
 child: \_imageBytes != null  
 ? Image.memory(\_imageBytes!, fit: BoxFit.contain) // Show selected image  
 : Center(child: Text('No Image Selected')), // Placeholder text  
 ),  
 ],  
 ),  
 SizedBox(height: 30.h),  
  
 Center(  
 child: InkWell(  
 onTap: \_submitClaim,  
 child: Container(  
 width: 150.w,  
 height: 50.h,  
 decoration: ShapeDecoration(  
 color: Colors.*blue*,  
 shape: RoundedRectangleBorder(  
 borderRadius: BorderRadius.circular(20.r),  
 ),  
 ),  
 child: Center(  
 child: Text(  
 "Submit",  
 style: TextStyle(color: Colors.*black*, fontSize: 15.sp, fontWeight: FontWeight.*bold*),  
 ),  
 ),  
 ),  
 ),  
 ),  
 SizedBox(height: 20.h),  
 ],  
 ),  
 ),  
 ),  
 ),  
 ),  
 ),  
 ),  
 ),  
 );  
 }  
}