import 'dart:convert';  
import 'dart:html' as html;  
import 'package:cloud\_firestore/cloud\_firestore.dart';  
import 'package:flutter/foundation.dart';  
import 'package:flutter/material.dart';  
import 'package:flutter\_bloc/flutter\_bloc.dart';  
import 'package:flutter\_screenutil/flutter\_screenutil.dart';  
import 'package:image\_picker/image\_picker.dart';  
import 'package:insuranceadminside/Bloc/ImageUpload%20Bloc/upload\_image\_bloc.dart';  
import 'package:insuranceadminside/Succesfull.dart';  
import 'package:intl/intl.dart';  
import 'Bloc/ImageUpload Bloc/upload\_image\_event.dart';  
import 'Repository/ModelClass/ProfileUploadClass.dart';  
  
class Addusers extends StatefulWidget {  
 const Addusers({super.key});  
  
 @override  
 State<Addusers> createState() => \_AddusersState();  
}  
  
class \_AddusersState extends State<Addusers> {  
 String? \_selectedParty;  
 DateTime? selectedDate;  
 DateTime? expireDate;  
 bool isMaleSelected = false;  
 bool isFemaleSelected = false;  
 int \_counter = 1; // Keep track of the increment counter  
  
 final FirebaseFirestore firestore =  
 FirebaseFirestore.*instance*; // Correct Firestore initialization  
 final TextEditingController nameController = TextEditingController();  
 final TextEditingController phoneController = TextEditingController();  
 final TextEditingController regIdController = TextEditingController();  
 final TextEditingController deviceIdController = TextEditingController();  
 final TextEditingController addressController = TextEditingController();  
 final TextEditingController emailIdController = TextEditingController();  
 final TextEditingController vehiclenumberController = TextEditingController();  
  
 final \_formKey = GlobalKey<FormState>();  
 final ImagePicker \_picker = ImagePicker();  
  
  
 // Fetch counter from Firestore  
 Future<int> \_getCounter() async {  
 try {  
 DocumentSnapshot snapshot =  
 await firestore.collection('Counters').doc('reg\_counter').get();  
  
 if (snapshot.exists) {  
 \_counter = snapshot['counter']; // Get the current counter value  
 return \_counter;  
 } else {  
 await firestore  
 .collection('Counters')  
 .doc('reg\_counter')  
 .set({'counter': \_counter});  
 return \_counter;  
 }  
 } catch (e) {  
 print("Error fetching counter: $e");  
 return \_counter;  
 }  
 }  
 // Update counter in Firestore  
 Future<void> \_updateCounter() async {  
 try {  
 await firestore  
 .collection('Counters')  
 .doc('reg\_counter')  
 .update({'counter': \_counter});  
 } catch (e) {  
 print("Error updating counter: $e");  
 }  
 }  
 // Custom function to generate Reg ID  
 void \_generateRegId() async {  
 // Fetch the current counter from Firestore  
 \_counter = await \_getCounter();  
  
 DateTime now = DateTime.now();  
 String datePart = DateFormat('ddyyyy')  
 .format(now); // Current date in ddyyyy format (day + year)  
  
 // Generate an incremental binary-like number (e.g., 0001, 0002, 0003...)  
 String binaryPart = \_generateBinaryLikeNumber();  
  
 setState(() {  
 regIdController.text =  
 'INS$datePart$binaryPart'; // Create the ID in the format "INSddyyyy0001"  
 });  
  
 // After generating the ID, increment and save the counter back to Firestore  
 \_counter++;  
 \_updateCounter();  
 }  
  
 String \_generateBinaryLikeNumber() {  
 // Format the counter as a binary-like number with leading zeros (e.g., 0001, 0002, etc.)  
 return \_counter.toString().padLeft(4, '0'); // 4 digits with leading zeros  
 }  
  
 // Date selection methods (same as in the original code)  
 Future<void> \_selectDate(BuildContext context) async {  
 final DateTime? picked = await showDatePicker(  
 context: context,  
 initialDate: selectedDate ?? DateTime.now(),  
 firstDate: DateTime(2000),  
 lastDate: DateTime(2100),  
 );  
  
 if (picked != null) {  
 setState(() {  
 selectedDate = picked;  
 });  
 }  
 }  
  
 Future<void> \_selectExpireDate(BuildContext context) async {  
 final DateTime? picked = await showDatePicker(  
 context: context,  
 initialDate: expireDate ?? DateTime.now(),  
 firstDate: DateTime(2000),  
 lastDate: DateTime(2100),  
 );  
  
 if (picked != null) {  
 setState(() {  
 expireDate = picked;  
 });  
 }  
 }  
  
  
  
 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  
 });  
 });  
 });  
 }  
  
 void \_submitData() async {  
 if (\_formKey.currentState?.validate() ?? false) {  
 if (selectedDate == null) {  
 ScaffoldMessenger.*of*(context).showSnackBar(  
 const SnackBar(content: Text('Insurance Date is required')));  
 return;  
 }  
 if (expireDate == null) {  
 ScaffoldMessenger.*of*(context).showSnackBar(  
 const SnackBar(content: Text('Expiry Date is required')));  
 return;  
 }  
 if (\_selectedParty == null) {  
 ScaffoldMessenger.*of*(context).showSnackBar(  
 const SnackBar(content: Text('Insurance Type is required')));  
 return;  
 }  
 if (!isMaleSelected && !isFemaleSelected) {  
 ScaffoldMessenger.*of*(context)  
 .showSnackBar(const SnackBar(content: Text('Gender is required')));  
 return;  
 }  
  
  
  
 try {  
 var docRef = await firestore.collection('Users').add({  
 'name': nameController.text,  
 'phone': phoneController.text,  
 'regId': regIdController.text,  
 'deviceId': deviceIdController.text,  
 'address': addressController.text,  
 'InsuranceType': \_selectedParty,  
 'EmailId': emailIdController.text,  
 'vehicleNumber': vehiclenumberController.text,  
 'insuranceDate': DateFormat('dd-MM-yyyy').format(selectedDate!),  
 'expiryDate': DateFormat('dd-MM-yyyy').format(expireDate!),  
 'gender': isMaleSelected ? 'Male' : 'Female',  
 'password': "",  
 });  
  
 await docRef.update({  
 'docId': docRef.id,  
 });  
  
  
 Navigator.*of*(context).push(MaterialPageRoute(  
 builder: (\_) => Succesfull(  
 id:docRef.id,  
 )));  
 // Clear the fields after submission  
 nameController.clear();  
 phoneController.clear();  
 regIdController.clear();  
 deviceIdController.clear();  
 addressController.clear();  
 emailIdController.clear();  
 vehiclenumberController.clear();  
  
 setState(() {  
 \_selectedParty = null;  
 selectedDate = null;  
 expireDate = null;  
 isMaleSelected = false;  
 isFemaleSelected = false;  
 });  
  
 ScaffoldMessenger.*of*(context).showSnackBar(  
 const SnackBar(content: Text('User added successfully')));  
 } catch (e) {  
 ScaffoldMessenger.*of*(context)  
 .showSnackBar(SnackBar(content: Text('Error: $e')));  
 }  
 }  
 }  
  
  
  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 backgroundColor: Colors.*grey*.withOpacity(0.4),  
 ),  
 body: Container(  
 width: 1440.w, // Full container width  
 height: 800.h, // Full container height  
 decoration: BoxDecoration(  
 image: DecorationImage(  
 image: AssetImage("assets/img.jpg"), // Background image  
 fit: BoxFit.cover, // Full-screen image  
 ),  
 ),  
 child:  
  
  
  
 Container(  
 color: Colors.*grey*.withOpacity(0.4),  
 child: Padding(  
 padding: EdgeInsets.only(left: 10.w, top: 10.h),  
 child: Form(  
 key: \_formKey,  
 child: SingleChildScrollView(  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 Row(  
 children: [  
 Column(  
 children: [  
  
  
 // Display selected image or default image  
 Container(  
 width: 200.w,  
 height: 250.h,  
 color: Colors.*white10*,  
 child: \_imageBytes != null  
 ? Image.memory(\_imageBytes!, fit: BoxFit.contain) // Shows full image without cropping  
 : Center(child: Text('No Image Selected')),  
 ),  
  
 SizedBox(height: 20),  
 InkWell(  
 onTap: () async {  
 \_pickImage();  
 },  
 child: Container(  
 width: 150.w,  
 height: 50.h,  
 decoration: ShapeDecoration(  
 color: Colors.*white*,  
 shape: RoundedRectangleBorder(  
 borderRadius: BorderRadius.circular(20),  
 ),  
 ),  
 child: Center(  
 child: Text(  
 "Upload photo",  
 style: TextStyle(fontSize: 20),  
 ),  
 ),  
 ),  
 ),  
 ],  
 ),  
 SizedBox(width: 10.w),  
 Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 Row(  
 children: [  
 \_buildField("Name ", nameController,  
 "Name is required"),  
 SizedBox(width: 20.w),  
 \_buildField("Phone ", phoneController,  
 "Phone number is required"),  
 SizedBox(width: 20.w),  
 ],  
 ),  
 SizedBox(height: 20.h),  
 Row(  
 children: [  
 \_buildFieldWithSuffix(  
 "Reg ID ",  
 regIdController,  
 "Registration ID is required"),  
 SizedBox(width: 20.w),  
 \_buildField("Email ID ",  
 emailIdController, "Email ID is required"),  
 ],  
 ),  
 SizedBox(height: 20.h),  
 Row(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 mainAxisAlignment: MainAxisAlignment.start,  
 children: [  
 \_buildField("Device ID ", deviceIdController,  
 "Device ID is required"),  
 SizedBox(width: 20.w),  
 \_buildField(  
 "Vechicle No ",  
 vehiclenumberController,  
 " Vechicle Number is required"),  
 ],  
 ),  
 ],  
 ),  
 ],  
 ),  
 SizedBox(height: 20.h),  
 Row(  
 children: [  
 SizedBox(width: 200.w),  
 \_buildField("Address ", addressController,  
 "Address is required",  
 maxLines: 5),  
 SizedBox(width: 50.w),  
 Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 SizedBox(height: 30.h),  
 Row(  
 children: [  
 Text("Gender : ",  
 style: TextStyle(  
 color: Colors.*black*, fontSize: 20.sp)),  
 SizedBox(width: 10.w),  
 GestureDetector(  
 onTap: () {  
 setState(() {  
 isMaleSelected = !isMaleSelected;  
 if (isMaleSelected) {  
 isFemaleSelected = false;  
 }  
 });  
 },  
 child: Container(  
 decoration: BoxDecoration(  
 shape: BoxShape.circle,  
 border: Border.all(  
 color: Colors.*black*, width: 2),  
 ),  
 child: Padding(  
 padding: EdgeInsets.all(2.0),  
 child: isMaleSelected  
 ? Container(  
 width: 20.sp,  
 height: 20.sp,  
 decoration: BoxDecoration(  
 shape: BoxShape.circle,  
 color: Colors.*white*,  
 ),  
 )  
 : SizedBox(  
 width: 20.sp, height: 20.sp),  
 ),  
 ),  
 ),  
 Text("Male",  
 style: TextStyle(  
 color: Colors.*black*, fontSize: 20.sp)),  
 SizedBox(width: 10.w),  
 GestureDetector(  
 onTap: () {  
 setState(() {  
 isFemaleSelected = !isFemaleSelected;  
 if (isFemaleSelected) {  
 isMaleSelected = false;  
 }  
 });  
 },  
 child: Container(  
 decoration: BoxDecoration(  
 shape: BoxShape.circle,  
 border: Border.all(  
 color: Colors.*black*, width: 2),  
 ),  
 child: Padding(  
 padding: EdgeInsets.all(2.0),  
 child: isFemaleSelected  
 ? Container(  
 width: 20.sp,  
 height: 20.sp,  
 decoration: BoxDecoration(  
 shape: BoxShape.circle,  
 color: Colors.*white*,  
 ),  
 )  
 : SizedBox(  
 width: 20.sp, height: 20.sp),  
 ),  
 ),  
 ),  
 Text("Female",  
 style: TextStyle(  
 color: Colors.*black*, fontSize: 20.sp)),  
 ],  
 ),  
 SizedBox(  
 height: 10.h,  
 ),  
 Row(  
 children: [  
 Text("Type :",  
 style: TextStyle(  
 color: Colors.*black*, fontSize: 20.sp)),  
 SizedBox(width: 20.w),  
 Container(  
 color: Colors.*white*,  
 child: DropdownButton<String>(  
 dropdownColor: Colors.*white*,  
 focusColor: Colors.*white*,  
 value: \_selectedParty,  
 hint: Text("Select Party Type"),  
 items: [  
 DropdownMenuItem(  
 value: "Full Cover",  
 child: Text("Full Cover")),  
 DropdownMenuItem(  
 value: "Third Party",  
 child: Text("Third Party")),  
 ],  
 onChanged: (String? newValue) {  
 setState(() {  
 \_selectedParty = newValue;  
 });  
 },  
 underline: SizedBox.shrink(),  
 ),  
 ),  
 ],  
 ),  
 SizedBox(height: 10.h),  
 \_buildDateField(  
 "Insurance Date :",  
 selectedDate,  
 () => \_selectDate(context),  
 "Select Insurance Date"),  
 SizedBox(height: 10.h),  
 \_buildDateField(  
 "Expiry Date :",  
 expireDate,  
 () => \_selectExpireDate(context),  
 "Select Expiry Date"),  
 SizedBox(height: 30.h),  
 ],  
 ),  
 ],  
 ),  
 InkWell(  
 onTap: \_submitData,  
 child: Center(  
 child: Container(  
 width: 350.w,  
 height: 60.h,  
 decoration: ShapeDecoration(  
 color: Colors.*blueAccent*,  
 shape: RoundedRectangleBorder(  
 borderRadius: BorderRadius.circular(20.r),  
 ),  
 ),  
 child: Center(  
 child: Text(  
 "Submit",  
 style: TextStyle(  
 color: Colors.*white*,  
 fontSize: 20.sp,  
 fontWeight: FontWeight.*bold*),  
 ),  
 ),  
 ),  
 ),  
 ),  
 SizedBox(  
 height: 40.h,  
 )  
 ],  
 ),  
 ),  
 ),  
 ),  
 ),  
  
 ),  
 );  
 }  
  
 Widget \_buildField(String label, TextEditingController controller,  
 String errorText,  
 {int maxLines = 1}) {  
 return Row(  
 children: [  
 Text(label, style: TextStyle(color: Colors.*black*, fontSize: 20.sp)),  
 SizedBox(  
 width: 400.w,  
 child: TextFormField(  
 controller: controller,  
 maxLines: maxLines,  
 decoration: InputDecoration(  
 hintText: label,  
 border: OutlineInputBorder(),  
 filled: true,  
 fillColor: Colors.*white*,  
 ),  
 validator: (value) {  
 if (value == null || value.isEmpty) {  
 return errorText;  
 }  
 return null;  
 },  
 ),  
 ),  
 ],  
 );  
 }  
  
 Widget \_buildFieldWithSuffix(String label, TextEditingController controller,  
 String errorText,  
 {int maxLines = 1}) {  
 return Row(  
 children: [  
 Text(label, style: TextStyle(color: Colors.*black*, fontSize: 20.sp)),  
 SizedBox(  
 width: 400.w,  
 child: TextFormField(  
 controller: controller,  
 maxLines: maxLines,  
 decoration: InputDecoration(  
 hintText: label,  
 border: OutlineInputBorder(),  
 filled: true,  
 fillColor: Colors.*white*,  
 suffixText: 'auto',  
 suffixIcon: IconButton(  
 icon: Icon(Icons.*refresh*),  
 onPressed: \_generateRegId, // Generates new Reg ID on tap  
 ),  
 ),  
 validator: (value) {  
 if (value == null || value.isEmpty) {  
 return errorText;  
 }  
 return null;  
 },  
 ),  
 ),  
 ],  
 );  
 }  
  
 Widget \_buildDateField(String label, DateTime? date, VoidCallback onTap,  
 String hintText) {  
 return Row(  
 children: [  
 Text(label, style: TextStyle(color: Colors.*black*, fontSize: 20.sp)),  
 SizedBox(width: 20.w),  
 GestureDetector(  
 onTap: onTap,  
 child: Container(  
 width: 200.w,  
 height: 50.h,  
 decoration: BoxDecoration(  
 borderRadius: BorderRadius.circular(10.r),  
 border: Border.all(color: Colors.*black*, width: 2),  
 ),  
 child: Center(  
 child: Text(  
 date == null ? hintText : DateFormat('dd-MM-yyyy').format(date),  
 style: TextStyle(fontSize: 18.sp),  
 ),  
 ),  
 ),  
 ),  
 ],  
 );  
 }  
}