**NON FORMAL-NFC MAD USING FLUTTER**

**PROJECT NAME: LOST AND FOUND APP**

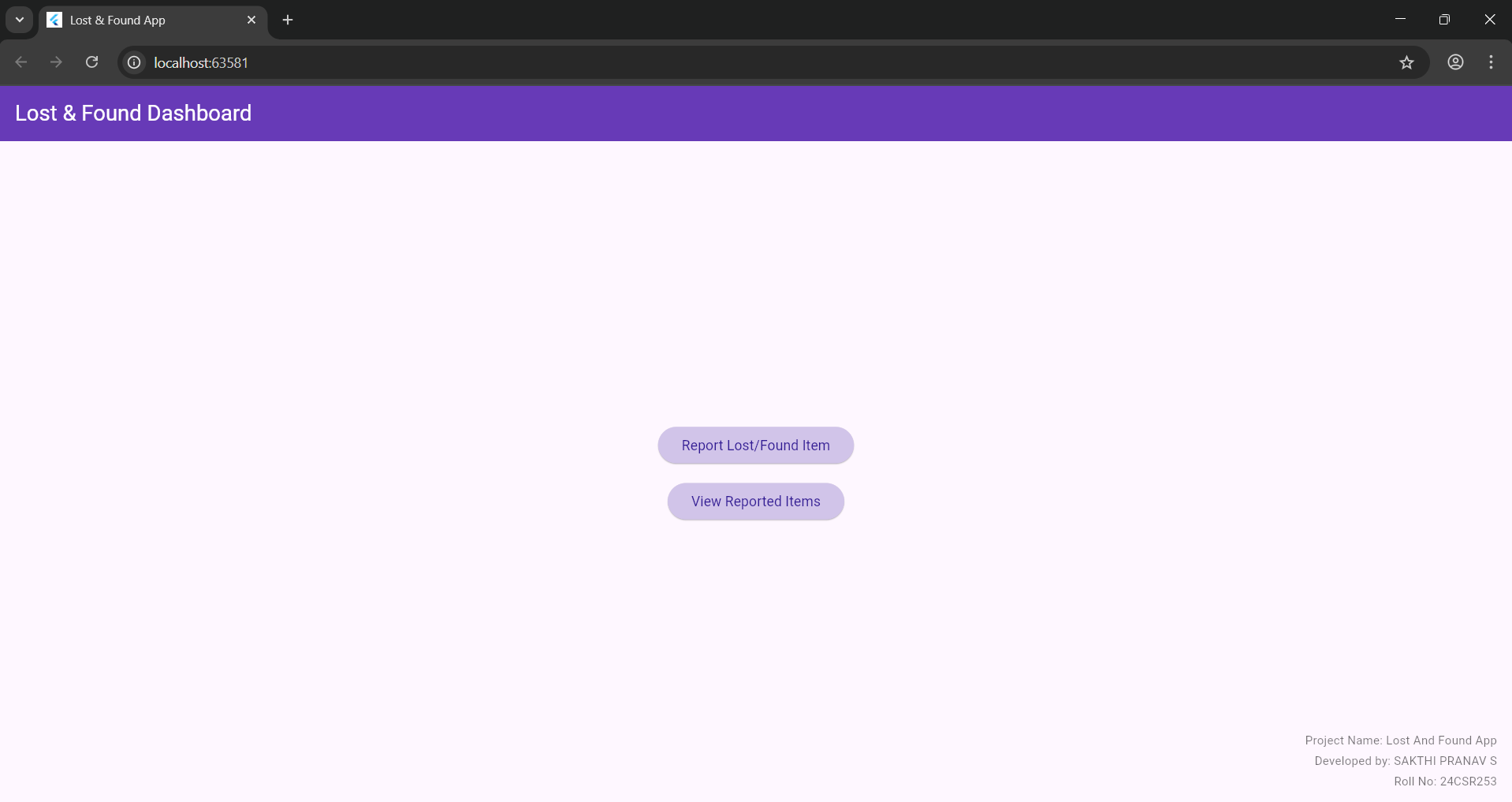
CODE:-

import 'package:flutter/material.dart';  
import 'package:firebase\_core/firebase\_core.dart';  
import 'package:cloud\_firestore/cloud\_firestore.dart';  
import 'firebase\_options.dart';  
  
void main() async {  
 WidgetsFlutterBinding.*ensureInitialized*();  
 await Firebase.*initializeApp*(  
 options: DefaultFirebaseOptions.*currentPlatform*,  
 );  
 runApp(const LostFoundApp());  
}  
  
class LostFoundApp extends StatelessWidget {  
 const LostFoundApp({super.key});  
  
 @override  
 Widget build(BuildContext context) {  
 return MaterialApp(  
 debugShowCheckedModeBanner: false,  
 title: 'Lost & Found App',  
 theme: ThemeData(  
 primarySwatch: Colors.*deepPurple*,  
 appBarTheme: const AppBarTheme(  
 backgroundColor: Colors.*deepPurple*,  
 foregroundColor: Colors.*white*,  
 ),  
 elevatedButtonTheme: ElevatedButtonThemeData(  
 style: ElevatedButton.*styleFrom*(  
 backgroundColor: Colors.*deepPurple*.shade100,  
 foregroundColor: Colors.*deepPurple*.shade900,  
 minimumSize: const Size(150, 45),  
 ),  
 ),  
 ),  
 home: const HomePage(),  
 );  
 }  
}  
  
class HomePage extends StatelessWidget {  
 const HomePage({super.key});  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(title: const Text("Lost & Found Dashboard")),  
 body: Stack(  
 children: [  
 Center(  
 child: Column(  
 mainAxisAlignment: MainAxisAlignment.center,  
 children: [  
 ElevatedButton(  
 child: const Text("Report Lost/Found Item"),  
 onPressed: () {  
 Navigator.*push*(  
 context,  
 MaterialPageRoute(builder: (\_) => const ReportItemPage()),  
 );  
 },  
 ),  
 const SizedBox(height: 20),  
 ElevatedButton(  
 child: const Text("View Reported Items"),  
 onPressed: () {  
 Navigator.*push*(  
 context,  
 MaterialPageRoute(builder: (\_) => const ItemsListPage()),  
 );  
 },  
 ),  
 ],  
 ),  
 ),  
  
  
 Positioned(  
 bottom: 16,  
 right: 16,  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.end,  
 children: const [  
 Text(  
 "Project Name: Lost And Found App",  
 style: TextStyle(fontSize: 12, color: Colors.*black54*),  
 ),  
 SizedBox(height: 4),  
 Text(  
 "Developed by: SAKTHI PRANAV S",  
 style: TextStyle(fontSize: 12, color: Colors.*black54*),  
 ),  
 SizedBox(height: 4),  
 Text(  
 "Roll No: 24CSR253",  
 style: TextStyle(fontSize: 12, color: Colors.*black54*),  
 ),  
 ],  
 ),  
 ),  
 ],  
 ),  
 );  
 }  
}  
  
  
class ReportItemPage extends StatefulWidget {  
 const ReportItemPage({super.key});  
  
 @override  
 State<ReportItemPage> createState() => \_ReportItemPageState();  
}  
  
class \_ReportItemPageState extends State<ReportItemPage> {  
 final \_formKey = GlobalKey<FormState>();  
 final \_nameController = TextEditingController();  
 final \_descriptionController = TextEditingController();  
 final \_locationController = TextEditingController();  
 final \_contactController = TextEditingController();  
  
 String \_type = 'Lost';  
  
 Future<void> \_insert() async {  
 if (!\_formKey.currentState!.validate()) return;  
  
 final name = \_nameController.text.trim();  
 final location = \_locationController.text.trim();  
  
 if (\_type == "Found") {  
  
 final lostItems = await FirebaseFirestore.*instance* .collection('items')  
 .where('type', isEqualTo: 'Lost')  
 .where('name', isEqualTo: name)  
 .where('location', isEqualTo: location)  
 .get();  
  
 for (var doc in lostItems.docs) {  
 await doc.reference.delete();  
 }  
 }  
  
 await FirebaseFirestore.*instance*.collection('items').add({  
 'type': \_type,  
 'name': name,  
 'description': \_descriptionController.text.trim(),  
 'location': location,  
 'contact': \_contactController.text.trim(),  
 'createdAt': FieldValue.*serverTimestamp*(),  
 });  
  
 \_show("Item Reported Successfully");  
  
 \_formKey.currentState?.reset();  
 \_nameController.clear();  
 \_descriptionController.clear();  
 \_locationController.clear();  
 \_contactController.clear();  
 setState(() => \_type = 'Lost');  
 }  
  
 void \_show(String message) {  
 ScaffoldMessenger.*of*(context).showSnackBar(SnackBar(content: Text(message)));  
 }  
  
 @override  
 void dispose() {  
 \_nameController.dispose();  
 \_descriptionController.dispose();  
 \_locationController.dispose();  
 \_contactController.dispose();  
 super.dispose();  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(title: const Text('Report Lost/Found Item')),  
 body: Padding(  
 padding: const EdgeInsets.all(12.0),  
 child: Form(  
 key: \_formKey,  
 child: ListView(  
 children: [  
 DropdownButtonFormField<String>(  
 value: \_type,  
 items: ['Lost', 'Found']  
 .map((t) => DropdownMenuItem(value: t, child: Text(t)))  
 .toList(),  
 onChanged: (val) {  
 if (val != null) setState(() => \_type = val);  
 },  
 decoration: const InputDecoration(labelText: 'Type'),  
 ),  
 const SizedBox(height: 8),  
 TextFormField(  
 controller: \_nameController,  
 decoration: const InputDecoration(labelText: 'Item Name'),  
 validator: (v) =>  
 v == null || v.trim().isEmpty ? 'Enter item name' : null,  
 ),  
 const SizedBox(height: 8),  
 TextFormField(  
 controller: \_descriptionController,  
 decoration: const InputDecoration(labelText: 'Description'),  
 maxLines: 2,  
 validator: (v) =>  
 v == null || v.trim().isEmpty ? 'Enter description' : null,  
 ),  
 const SizedBox(height: 8),  
 TextFormField(  
 controller: \_locationController,  
 decoration: const InputDecoration(labelText: 'Location'),  
 validator: (v) =>  
 v == null || v.trim().isEmpty ? 'Enter location' : null,  
 ),  
 const SizedBox(height: 8),  
 TextFormField(  
 controller: \_contactController,  
 decoration: const InputDecoration(labelText: 'Contact Info'),  
 validator: (v) =>  
 v == null || v.trim().isEmpty ? 'Enter contact info' : null,  
 ),  
 const SizedBox(height: 20),  
 Center(  
 child: ElevatedButton(  
 onPressed: \_insert,  
 child: const Text("Submit"),  
 ),  
 ),  
 ],  
 ),  
 ),  
 ),  
 );  
 }  
}  
  
class ItemsListPage extends StatelessWidget {  
 const ItemsListPage({super.key});  
  
 Future<void> \_deleteItem(String docId, BuildContext context) async {  
 await FirebaseFirestore.*instance*.collection('items').doc(docId).delete();  
 ScaffoldMessenger.*of*(context)  
 .showSnackBar(const SnackBar(content: Text('Item deleted')));  
 }  
  
 void \_editItem(BuildContext context, String docId, Map<String, dynamic> data) {  
 Navigator.*push*(  
 context,  
 MaterialPageRoute(  
 builder: (\_) => EditItemPage(docId: docId, data: data),  
 ),  
 );  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(title: const Text('Reported Items')),  
 body: StreamBuilder<QuerySnapshot>(  
 stream: FirebaseFirestore.*instance* .collection('items')  
 .orderBy('createdAt', descending: true)  
 .snapshots(),  
 builder: (context, snapshot) {  
 if (snapshot.hasError) {  
 return Center(child: Text('Error: ${snapshot.error}'));  
 }  
 if (!snapshot.hasData) {  
 return const Center(child: CircularProgressIndicator());  
 }  
  
 final docs = snapshot.data!.docs;  
 if (docs.isEmpty) {  
 return const Center(child: Text('No items reported yet'));  
 }  
  
 return ListView.builder(  
 itemCount: docs.length,  
 itemBuilder: (context, i) {  
 final doc = docs[i];  
 final data = doc.data() as Map<String, dynamic>? ?? {};  
 final type = (data['type'] ?? '—').toString();  
 final name = (data['name'] ?? '—').toString();  
 final description = (data['description'] ?? '').toString();  
 final location = (data['location'] ?? '').toString();  
 final contact = (data['contact'] ?? '').toString();  
  
 return Card(  
 child: ListTile(  
 leading: Icon(  
 type == 'Lost' ? Icons.*report\_problem* : Icons.*check\_circle*,  
 color: type == 'Lost' ? Colors.*red* : Colors.*green*,  
 ),  
 title: Text(  
 "$type: $name",  
 style: const TextStyle(fontWeight: FontWeight.*bold*),  
 ),  
 subtitle: Text(  
 "Description: $description\nLocation: $location\nContact: $contact"),  
 isThreeLine: true,  
 trailing: Row(  
 mainAxisSize: MainAxisSize.min,  
 children: [  
 IconButton(  
 icon: const Icon(Icons.*edit*, color: Colors.*blue*),  
 onPressed: () => \_editItem(context, doc.id, data),  
 ),  
 IconButton(  
 icon: const Icon(Icons.*delete*, color: Colors.*black54*),  
 onPressed: () => \_deleteItem(doc.id, context),  
 ),  
 ],  
 ),  
 ),  
 );  
 },  
 );  
 },  
 ),  
 );  
 }  
}  
  
class EditItemPage extends StatefulWidget {  
 final String docId;  
 final Map<String, dynamic> data;  
  
 const EditItemPage({super.key, required this.docId, required this.data});  
  
 @override  
 State<EditItemPage> createState() => \_EditItemPageState();  
}  
  
class \_EditItemPageState extends State<EditItemPage> {  
 final \_formKey = GlobalKey<FormState>();  
 late TextEditingController \_nameController;  
 late TextEditingController \_descriptionController;  
 late TextEditingController \_locationController;  
 late TextEditingController \_contactController;  
 late String \_type;  
  
 @override  
 void initState() {  
 super.initState();  
 \_type = widget.data['type'] ?? 'Lost';  
 \_nameController = TextEditingController(text: widget.data['name']);  
 \_descriptionController =  
 TextEditingController(text: widget.data['description']);  
 \_locationController = TextEditingController(text: widget.data['location']);  
 \_contactController = TextEditingController(text: widget.data['contact']);  
 }  
  
 Future<void> \_updateItem() async {  
 if (!\_formKey.currentState!.validate()) return;  
  
 await FirebaseFirestore.*instance*.collection('items').doc(widget.docId).update({  
 'type': \_type,  
 'name': \_nameController.text.trim(),  
 'description': \_descriptionController.text.trim(),  
 'location': \_locationController.text.trim(),  
 'contact': \_contactController.text.trim(),  
 });  
  
 ScaffoldMessenger.*of*(context).showSnackBar(  
 const SnackBar(content: Text('Item updated successfully')));  
 Navigator.*pop*(context);  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(title: const Text("Edit Item")),  
 body: Padding(  
 padding: const EdgeInsets.all(12.0),  
 child: Form(  
 key: \_formKey,  
 child: ListView(  
 children: [  
 DropdownButtonFormField<String>(  
 value: \_type,  
 items: ['Lost', 'Found']  
 .map((t) => DropdownMenuItem(value: t, child: Text(t)))  
 .toList(),  
 onChanged: (val) {  
 if (val != null) setState(() => \_type = val);  
 },  
 decoration: const InputDecoration(labelText: 'Type'),  
 ),  
 const SizedBox(height: 8),  
 TextFormField(  
 controller: \_nameController,  
 decoration: const InputDecoration(labelText: 'Item Name'),  
 validator: (v) =>  
 v == null || v.trim().isEmpty ? 'Enter item name' : null,  
 ),  
 const SizedBox(height: 8),  
 TextFormField(  
 controller: \_descriptionController,  
 decoration: const InputDecoration(labelText: 'Description'),  
 maxLines: 2,  
 validator: (v) =>  
 v == null || v.trim().isEmpty ? 'Enter description' : null,  
 ),  
 const SizedBox(height: 8),  
 TextFormField(  
 controller: \_locationController,  
 decoration: const InputDecoration(labelText: 'Location'),  
 validator: (v) =>  
 v == null || v.trim().isEmpty ? 'Enter location' : null,  
 ),  
 const SizedBox(height: 8),  
 TextFormField(  
 controller: \_contactController,  
 decoration: const InputDecoration(labelText: 'Contact Info'),  
 validator: (v) =>  
 v == null || v.trim().isEmpty ? 'Enter contact info' : null,  
 ),  
 const SizedBox(height: 20),  
 Center(  
 child: ElevatedButton(  
 onPressed: \_updateItem,  
 child: const Text("Update"),  
 ),  
 ),  
 ],  
 ),  
 ),  
 ),  
 );  
 }  
}

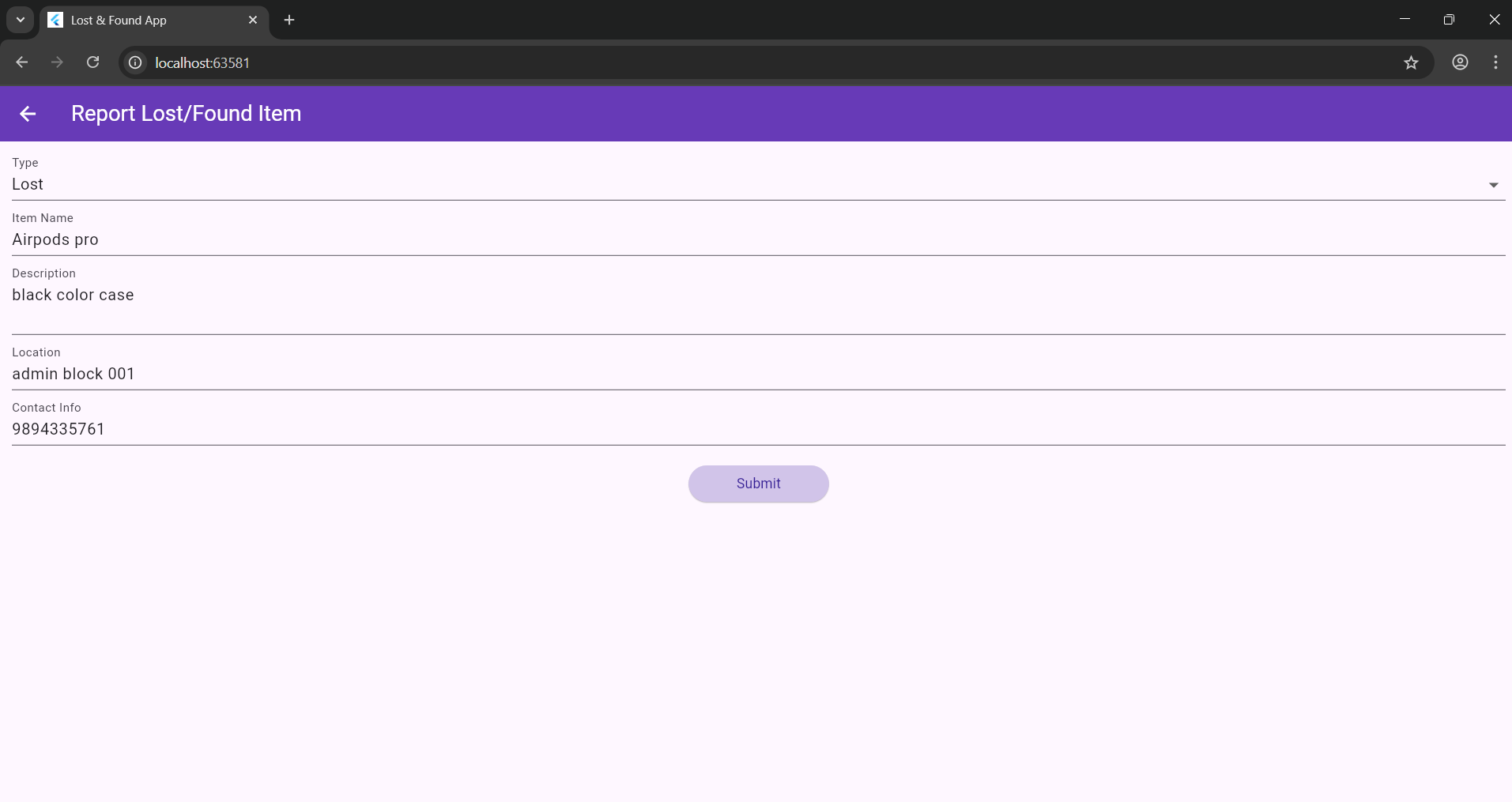
DEPENDENCIES:

flutter:  
 sdk: flutter  
firebase\_core: ^4.1.0  
cloud\_firestore: ^6.0.1

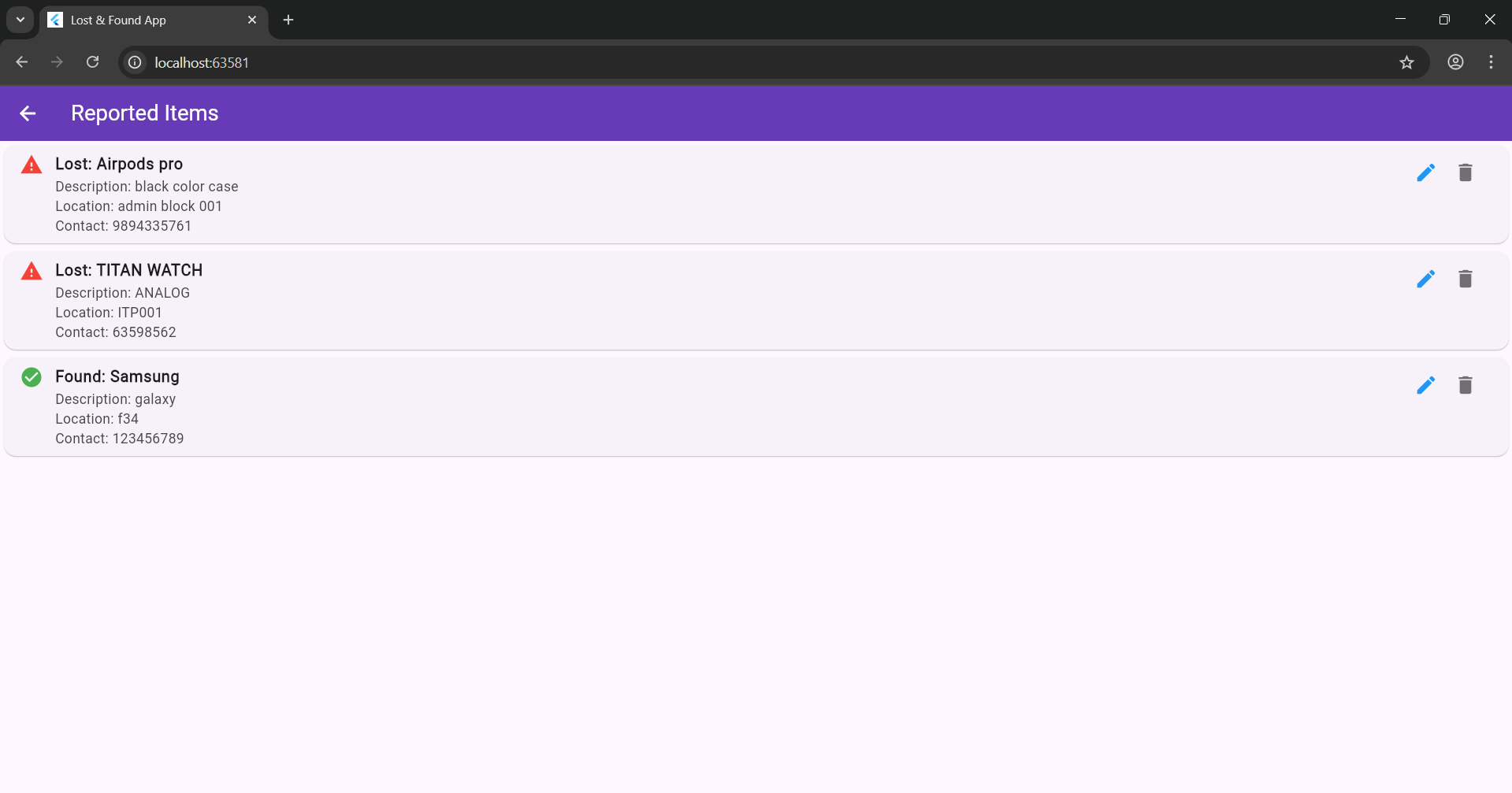
SCREENSHOTS:



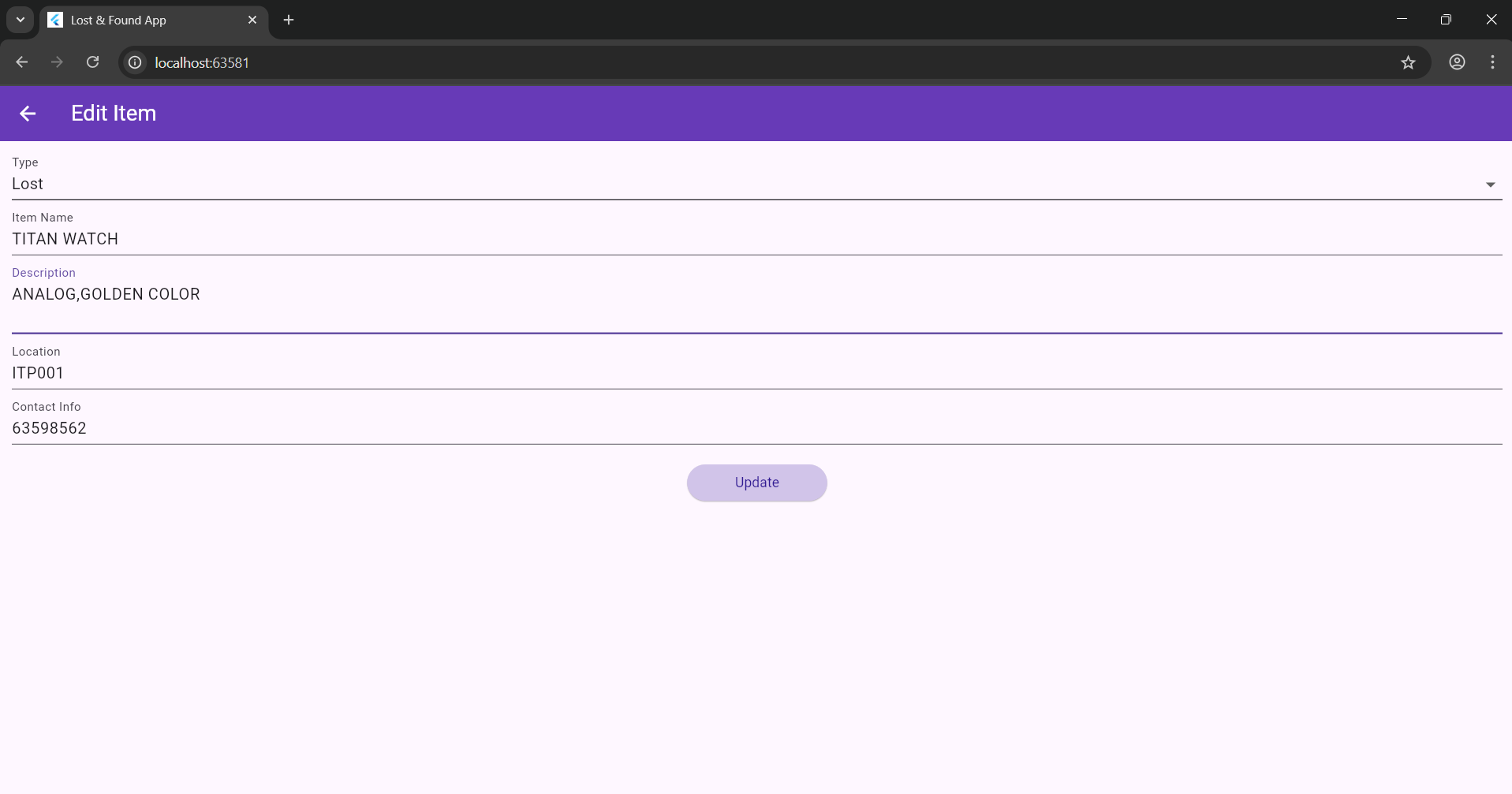
1.HOME PAGE



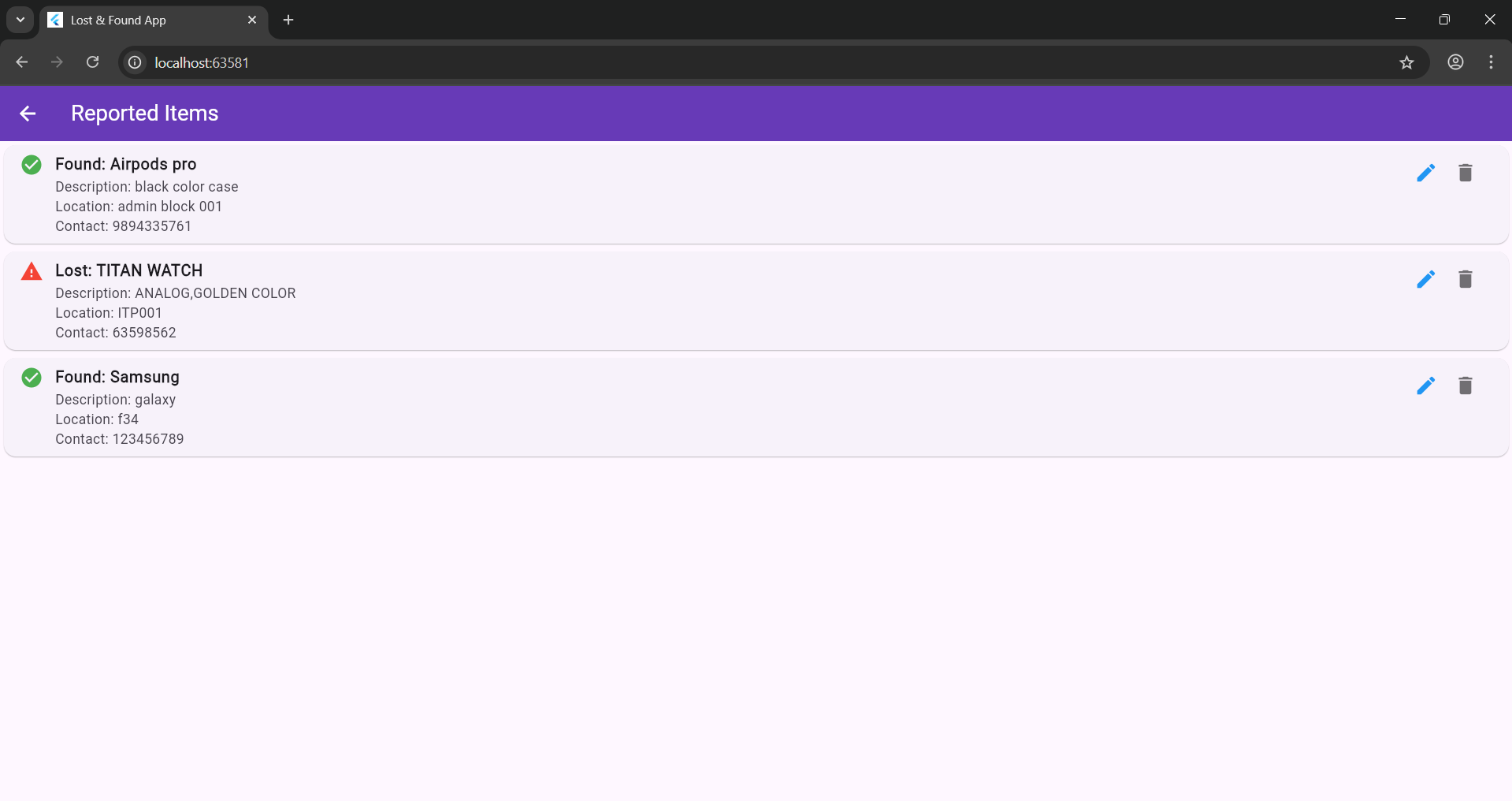
2.INSERT



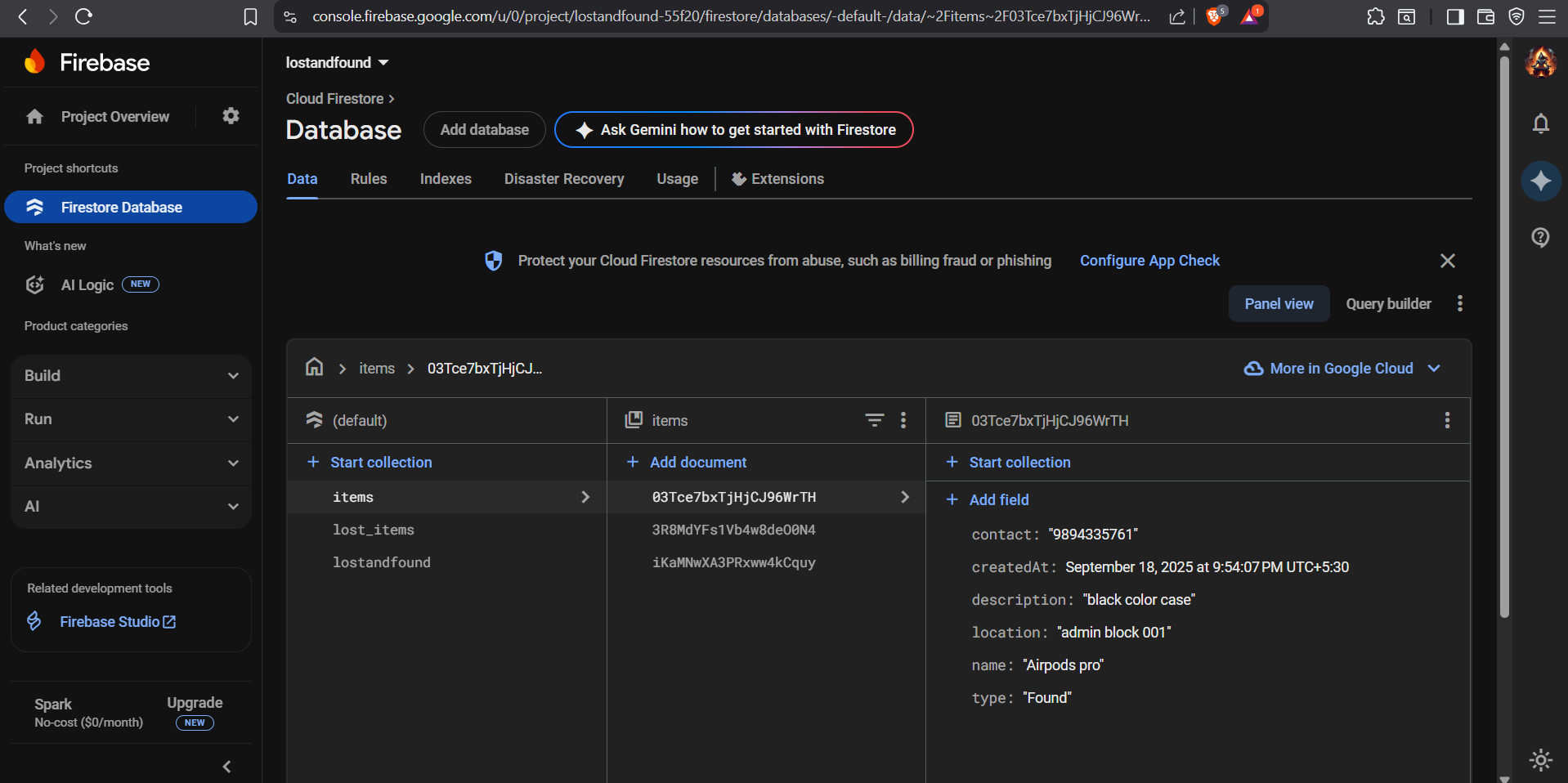
3.READ



4.UPDATE



5.DELETE



6.FIREBASE