// lib/main.dart  
import 'package:flutter/material.dart';  
import 'price\_editor.dart';  
import 'roll\_calc.dart';  
import 'package:shared\_preferences/shared\_preferences.dart';  
import 'dart:convert';  
import 'models.dart';

import 'settings.dart';

import 'history.dart';

import 'invoice\_view.dart';

import 'utils.dart';

import 'package:uuid/uuid.dart';

import 'package:intl/intl.dart';  
void main() async {  
 WidgetsFlutterBinding.ensureInitialized();  
 runApp(const ShutterApp());  
}  
  
class ShutterApp extends StatelessWidget {  
 const ShutterApp({super.key});  
 @override  
 Widget build(BuildContext context) {  
 return MaterialApp(  
 title: 'کرکره برقی - نوازش',  
 theme: ThemeData.dark().copyWith(  
 colorScheme: ColorScheme.dark(  
 primary: Colors.tealAccent,  
 secondary: Colors.teal,  
 ),  
 scaffoldBackgroundColor: const Color(0xFF0F1722),  
 inputDecorationTheme: const InputDecorationTheme(  
 filled: true,  
 fillColor: Color(0xFF111827),  
 border: OutlineInputBorder(),  
 ),  
 ),  
 home: const ShutterHome(),  
 debugShowCheckedModeBanner: false,  
 );  
 }  
}  
  
class PriceStore {  
 static const \_key = 'shutter\_prices\_v2';  
 Map<String, dynamic> data = {  
 "blades": {  
 "اوپال 80": {"price": 2400000, "thickness": 2.0}  
 },  
 "motors": {"ADC 300": 7900000},  
 "shafts": {"11": 350000},  
 "boxes": {"4x4": 168000},  
 "install": 200000,  
 "welding": 2000000,  
 "transport": 200000,  
 "lock\_simple": 300000,  
 "lock\_electric": 300000,  
 "motor\_cover": 300000  
 };  
  
 Future<void> load() async {  
 final prefs = await SharedPreferences.getInstance();  
 final s = prefs.getString(\_key);  
 if (s != null) {  
 try {  
 data = json.decode(s);  
 } catch (\_) {}  
 }  
 }  
  
 Future<void> save() async {  
 final prefs = await SharedPreferences.getInstance();  
 await prefs.setString(\_key, json.encode(data));  
 }  
}  
  
class ShutterHome extends StatefulWidget {  
 const ShutterHome({super.key});  
 @override  
 State<ShutterHome> createState() => \_ShutterHomeState();  
}  
  
class \_ShutterHomeState extends State<ShutterHome> {  
 final PriceStore store = PriceStore();  
  
 final TextEditingController widthCtrl = TextEditingController();  
 final TextEditingController heightCtrl = TextEditingController();  
 final TextEditingController shaftLenCtrl = TextEditingController();  
 final TextEditingController boxLenCtrl = TextEditingController();  
  
 String? selectedBlade;  
 String? selectedMotor;  
 String? selectedShaft;  
 String? selectedBox;  
  
 bool lockSimple = false;  
 bool lockElectric = false;  
 bool motorCover = false;  
  
 double area = 0.0;  
 double rollDiameter = 0.0;  
 Map<String, dynamic> prices = {};  
  
 bool shaftManualEdited = false;  
 bool boxManualEdited = false;  
  
 @override  
 void initState() {  
 super.initState();  
 \_init();  
 }  
  
 Future<void> \_init() async {  
 await store.load();  
 setState(() {  
 prices = store.data;  
 if ((prices['blades'] as Map).isNotEmpty) selectedBlade = (prices['blades'] as Map).keys.first;  
 if ((prices['motors'] as Map).isNotEmpty) selectedMotor = (prices['motors'] as Map).keys.first;  
 if ((prices['shafts'] as Map).isNotEmpty) selectedShaft = (prices['shafts'] as Map).keys.first;  
 if ((prices['boxes'] as Map).isNotEmpty) selectedBox = (prices['boxes'] as Map).keys.first;  
 });  
 }  
  
 void \_calcArea() {  
 final w = double.tryParse(widthCtrl.text.replaceAll(',', '.')) ?? 0.0;  
 final h = double.tryParse(heightCtrl.text.replaceAll(',', '.')) ?? 0.0;  
 setState(() {  
 area = w \* h;  
 });  
 \_syncDependentFields(w, h);  
 \_calcRollAuto();  
 }  
  
 void \_syncDependentFields(double w, double h) {  
 if (!shaftManualEdited) {  
 shaftLenCtrl.text = w.toStringAsFixed(2);  
 }  
 if (!boxManualEdited) {  
 final boxLen = ((h - 0.30) > 0 ? (h - 0.30) \* 2 : 0.0);  
 boxLenCtrl.text = boxLen.toStringAsFixed(2);  
 }  
 }  
  
 void \_onShaftEdited(String v) {  
 shaftManualEdited = true;  
 }  
 void \_onBoxEdited(String v) {  
 boxManualEdited = true;  
 }  
  
 void \_calcRollAuto() {  
 try {  
 final shaftDiameter = double.tryParse(selectedShaft ?? '') ?? 11.0;  
 final thickness = (prices['blades']?[selectedBlade]?['thickness'] as num?)?.toDouble() ?? 2.0;  
 final heightCm = (double.tryParse(heightCtrl.text.replaceAll(',', '.')) ?? 0.0) \* 100.0;  
 final roll = shaftDiameter + (heightCm \* (thickness / 100.0));  
 setState(() {  
 rollDiameter = roll;  
 });  
 } catch (\_) {  
 setState(() { rollDiameter = 0.0; });  
 }  
 }  
  
 Map<String, double> \_computeTotals() {  
 final bladePricePerM2 = (prices['blades']?[selectedBlade]?['price'] as num?)?.toDouble() ?? 0.0;  
 final motorPrice = (prices['motors']?[selectedMotor] as num?)?.toDouble() ?? 0.0;  
 final shaftPricePerM = (prices['shafts']?[selectedShaft] as num?)?.toDouble() ?? 0.0;  
 final boxPricePerM = (prices['boxes']?[selectedBox] as num?)?.toDouble() ?? 0.0;  
 final shaftLen = double.tryParse(shaftLenCtrl.text.replaceAll(',', '.')) ?? 0.0;  
 final boxLen = double.tryParse(boxLenCtrl.text.replaceAll(',', '.')) ?? 0.0;  
 final installBase = (prices['install'] as num?)?.toDouble() ?? 0.0;  
 final welding = (prices['welding'] as num?)?.toDouble() ?? 0.0;  
 final transport = (prices['transport'] as num?)?.toDouble() ?? 0.0;  
 final lockSimplePrice = (prices['lock\_simple'] as num?)?.toDouble() ?? 0.0;  
 final lockElectricPrice = (prices['lock\_electric'] as num?)?.toDouble() ?? 0.0;  
 final motorCoverPrice = (prices['motor\_cover'] as num?)?.toDouble() ?? 0.0;  
  
 final bladeTotal = area \* bladePricePerM2;  
 final shaftTotal = shaftLen \* shaftPricePerM;  
 final boxTotal = boxLen \* boxPricePerM;  
 // install with min-10 rule  
 double installTotal;  
 if (area <= 10.0) {  
 installTotal = 10.0 \* installBase;  
 } else {  
 installTotal = area \* installBase;  
 }  
  
 double total = bladeTotal + motorPrice + shaftTotal + boxTotal + installTotal + welding + transport;  
 if (lockSimple) total += lockSimplePrice;  
 if (lockElectric) total += lockElectricPrice;  
 if (motorCover) total += motorCoverPrice;  
  
 return {  
 'area': area,  
 'bladeTotal': bladeTotal,  
 'shaftTotal': shaftTotal,  
 'boxTotal': boxTotal,  
 'installTotal': installTotal,  
 'welding': welding,  
 'transport': transport,  
 'total': total  
 };  
 }  
  
 void \_showResult() {  
 \_calcArea();  
 final t = \_computeTotals();  
 showDialog(context: context, builder: (\_) {  
 return AlertDialog(  
 backgroundColor: const Color(0xFF0B1220),  
 title: const Text('نتیجه محاسبه', style: TextStyle(color: Colors.white)),  
 content: Text(  
 'مساحت: ${t['area']!.toStringAsFixed(2)} متر مربع\n'  
 'قیمت تیغه: ${t['bladeTotal']!.toStringAsFixed(0)} تومان\n'  
 'شفت: ${t['shaftTotal']!.toStringAsFixed(0)} تومان\n'  
 'قوطی: ${t['boxTotal']!.toStringAsFixed(0)} تومان\n'  
 'نصب: ${t['installTotal']!.toStringAsFixed(0)} تومان\n'  
 'جوشکاری: ${t['welding']!.toStringAsFixed(0)} تومان\n'  
 'حمل: ${t['transport']!.toStringAsFixed(0)} تومان\n'  
 'آیتم‌های انتخابی: ${(lockSimple? (prices['lock\_simple'] as num).toDouble():0) + (lockElectric? (prices['lock\_electric'] as num).toDouble():0) + (motorCover? (prices['motor\_cover'] as num).toDouble():0)} تومان\n\n'  
 'قیمت نهایی: ${t['total']!.toStringAsFixed(0)} تومان\n\n'  
 'قطر رول: ${rollDiameter.toStringAsFixed(1)} cm\n\n'  
 'فروشنده: نوازش\nتلفن: 09168413916',  
 style: const TextStyle(color: Colors.white70),  
 ),  
 actions: [  
 TextButton(onPressed: () => Navigator.pop(context), child: const Text('بستن'))  
 ],  
 );  
 });  
 }  
  
 Future<void> \_openPriceEditor() async {  
 await Navigator.push(context, MaterialPageRoute(builder: (\_) => PriceEditorScreen(store: store)));  
 await store.load();  
 setState(() { prices = store.data; });  
 }  
  
 Future<void> \_openRollCalc() async {  
 await Navigator.push(context, MaterialPageRoute(builder: (\_) => const RollCalcScreen()));  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 final bladeKeys = (prices['blades'] as Map?)?.keys?.toList() ?? <String>[];  
 final motorKeys = (prices['motors'] as Map?)?.keys?.toList() ?? <String>[];  
 final shaftKeys = (prices['shafts'] as Map?)?.keys?.toList() ?? <String>[];  
 final boxKeys = (prices['boxes'] as Map?)?.keys?.toList() ?? <String>[];  
  
 return Scaffold(  
 appBar: AppBar(  
 title: const Text('کرکره برقی - نوازش'),  
 actions: [  
 IconButton(icon: const Icon(Icons.settings), onPressed: \_openPriceEditor),  
 IconButton(icon: const Icon(Icons.straighten), onPressed: \_openRollCalc),  
 ],  
 ),  
 body: Padding(  
 padding: const EdgeInsets.all(12.0),  
 child: SingleChildScrollView(  
 child: Column(  
 children: [  
 Row(children: [  
 Expanded(child: TextField(controller: widthCtrl, keyboardType: const TextInputType.numberWithOptions(decimal: true), decoration: const InputDecoration(labelText: 'عرض (متر)'), onChanged: (\_) => \_calcArea())),  
 const SizedBox(width: 8),  
 Expanded(child: TextField(controller: heightCtrl, keyboardType: const TextInputType.numberWithOptions(decimal: true), decoration: const InputDecoration(labelText: 'ارتفاع (متر)'), onChanged: (\_) => \_calcArea())),  
 IconButton(onPressed: \_calcArea, icon: const Icon(Icons.refresh))  
 ]),  
 const SizedBox(height: 8),  
 Align(alignment: Alignment.centerLeft, child: Text('مساحت: ${area.toStringAsFixed(2)} متر مربع', style: const TextStyle(fontSize: 16))),  
 const SizedBox(height: 12),  
 DropdownButtonFormField<String>(  
 value: bladeKeys.contains(selectedBlade) ? selectedBlade : (bladeKeys.isNotEmpty ? bladeKeys.first : null),  
 decoration: const InputDecoration(labelText: 'تیغه'),  
 items: bladeKeys.map((e) => DropdownMenuItem(value: e, child: Text(e))).toList(),  
 onChanged: (v) { setState(() { selectedBlade = v; \_calcRollAuto(); }); },  
 ),  
 const SizedBox(height: 8),  
 Align(alignment: Alignment.centerLeft, child: Text('قیمت تیغه: ${prices['blades']?[selectedBlade]?['price']?.toString() ?? '-'} تومان', style: const TextStyle(color: Colors.white70))),  
 const SizedBox(height: 12),  
 DropdownButtonFormField<String>(  
 value: motorKeys.contains(selectedMotor) ? selectedMotor : (motorKeys.isNotEmpty ? motorKeys.first : null),  
 decoration: const InputDecoration(labelText: 'موتور'),  
 items: motorKeys.map((e) => DropdownMenuItem(value: e, child: Text(e))).toList(),  
 onChanged: (v) { setState(() { selectedMotor = v; }); },  
 ),  
 const SizedBox(height: 12),  
 DropdownButtonFormField<String>(  
 value: shaftKeys.contains(selectedShaft) ? selectedShaft : (shaftKeys.isNotEmpty ? shaftKeys.first : null),  
 decoration: const InputDecoration(labelText: 'شفت (قطر یا نام)'),  
 items: shaftKeys.map((e) => DropdownMenuItem(value: e, child: Text(e))).toList(),  
 onChanged: (v) { setState(() { selectedShaft = v; \_calcRollAuto(); }); },  
 ),  
 TextField(controller: shaftLenCtrl, keyboardType: const TextInputType.numberWithOptions(decimal: true), decoration: const InputDecoration(labelText: 'متراژ شفت (متر)'), onChanged: \_onShaftEdited),  
 const SizedBox(height: 12),  
 DropdownButtonFormField<String>(  
 value: boxKeys.contains(selectedBox) ? selectedBox : (boxKeys.isNotEmpty ? boxKeys.first : null),  
 decoration: const InputDecoration(labelText: 'قوطی'),  
 items: boxKeys.map((e) => DropdownMenuItem(value: e, child: Text(e))).toList(),  
 onChanged: (v) { setState(() { selectedBox = v; }); },  
 ),  
 TextField(controller: boxLenCtrl, keyboardType: const TextInputType.numberWithOptions(decimal: true), decoration: const InputDecoration(labelText: 'متراژ قوطی (متر)'), onChanged: \_onBoxEdited),  
 const SizedBox(height: 12),  
 CheckboxListTile(title: const Text('جاقفلی ساده'), value: lockSimple, onChanged: (v){ setState(()=> lockSimple = v ?? false); }),  
 CheckboxListTile(title: const Text('جاقفلی برقی'), value: lockElectric, onChanged: (v){ setState(()=> lockElectric = v ?? false); }),  
 CheckboxListTile(title: const Text('کاور موتور'), value: motorCover, onChanged: (v){ setState(()=> motorCover = v ?? false); }),  
 const SizedBox(height: 12),  
 Align(alignment: Alignment.centerLeft, child: Text('قطر رول: ${rollDiameter.toStringAsFixed(1)} cm', style: const TextStyle(fontSize: 16))),  
 const SizedBox(height: 12),  
 ElevatedButton.icon(onPressed: \_showResult, icon: const Icon(Icons.calculate), label: const Text('محاسبه نهایی 💰')),  
 const SizedBox(height: 20),  
 const Divider(),  
 const SizedBox(height: 8),  
 const Text('فروشنده: نوازش', style: TextStyle(fontSize: 14)),  
 const Text('تلفن: 09168413916', style: TextStyle(fontSize: 14)),  
 ],  
 ),  
 ),  
 ),  
 );  
 }  
}