import 'dart:io';

import 'package:flutter/material.dart';

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

import 'package:geolocator/geolocator.dart';

import 'package:sqflite/sqflite.dart';

import 'package:path/path.dart';

import 'package:intl/intl.dart';

import 'package:qr\_flutter/qr\_flutter.dart';

import 'package:uuid/uuid.dart';

void main() async {

WidgetsFlutterBinding.ensureInitialized();

final db = await AppDatabase.open();

runApp(MyApp(database: db));

}

class MyApp extends StatelessWidget {

final AppDatabase database;

MyApp({required this.database});

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'RSW Prototype',

theme: ThemeData(primarySwatch: Colors.green),

home: HomePage(database: database),

);

}

}

class HomePage extends StatefulWidget {

final AppDatabase database;

HomePage({required this.database});

@override

\_HomePageState createState() => \_HomePageState();

}

class \_HomePageState extends State<HomePage> {

List<TreeRecord> trees = [];

@override

void initState() {

super.initState();

\_loadTrees();

}

Future<void> \_loadTrees() async {

final all = await widget.database.getAllTrees();

setState(() => trees = all);

}

void \_openAdd() async {

final res = await Navigator.push(context,

MaterialPageRoute(builder: (c) => AddTreePage(database: widget.database)));

if (res == true) \_loadTrees();

}

void \_openDetail(TreeRecord t) {

Navigator.push(context, MaterialPageRoute(builder: (c) => DetailPage(tree: t)));

}

void \_openScan() async {

// For prototype, simple dialog to input QR data (UUID)

final code = await showDialog<String>(

context: context,

builder: (c) => ScanSimDialog(),

);

if (code != null && code.isNotEmpty) {

final t = await widget.database.getTreeById(code);

if (t != null) \_openDetail(t);

else ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text('Pokok tak jumpa')));

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text('RSW - Prototype')),

body: ListView.builder(

itemCount: trees.length,

itemBuilder: (c, i) {

final t = trees[i];

return ListTile(

leading: t.imagePath != null ? Image.file(File(t.imagePath!), width: 50, height: 50, fit: BoxFit.cover) : Icon(Icons.nature),

title: Text(t.name ?? 'Unnamed'),

subtitle: Text('Semai: ${DateFormat('yyyy-MM-dd').format(t.seedDate)} | Rootstock: ${t.rootstock}'),

onTap: () => \_openDetail(t),

);

},

),

floatingActionButton: Column(

mainAxisSize: MainAxisSize.min,

children: [

FloatingActionButton.extended(onPressed: \_openScan, icon: Icon(Icons.qr\_code\_scanner), label: Text('Scan QR')),

SizedBox(height: 8),

FloatingActionButton.extended(onPressed: \_openAdd, icon: Icon(Icons.add), label: Text('Daftar Pokok')),

],

),

);

}

}

class AddTreePage extends StatefulWidget {

final AppDatabase database;

AddTreePage({required this.database});

@override

\_AddTreePageState createState() => \_AddTreePageState();

}

class \_AddTreePageState extends State<AddTreePage> {

final \_formKey = GlobalKey<FormState>();

String name = '';

DateTime seedDate = DateTime.now();

String? rootstock;

File? imageFile;

double? lat;

double? lng;

final picker = ImagePicker();

final uuid = Uuid();

final List<String> rootstockList = ['Angsana', 'Acacia', 'Albizia', 'Macrocarpus'];

Future<void> \_getLocation() async {

bool serviceEnabled = await Geolocator.isLocationServiceEnabled();

if (!serviceEnabled) {

ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text('Sila aktifkan location services')));

return;

}

LocationPermission permission = await Geolocator.checkPermission();

if (permission == LocationPermission.denied) permission = await Geolocator.requestPermission();

if (permission == LocationPermission.deniedForever || permission == LocationPermission.denied) {

ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text('Kebenaran lokasi diperlukan')));

return;

}

final pos = await Geolocator.getCurrentPosition(desiredAccuracy: LocationAccuracy.high);

setState(() {

lat = pos.latitude;

lng = pos.longitude;

});

}

Future<void> \_pickImage() async {

final picked = await picker.pickImage(source: ImageSource.camera, maxWidth: 1600);

if (picked != null) setState(() => imageFile = File(picked.path));

}

Future<void> \_save() async {

if (!\_formKey.currentState!.validate() || rootstock == null) {

ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text('Lengkapkan maklumat')));

return;

}

final id = uuid.v4();

final record = TreeRecord(

id: id,

name: name,

seedDate: seedDate,

rootstock: rootstock!,

imagePath: imageFile?.path,

lat: lat,

lng: lng,

createdAt: DateTime.now(),

);

await widget.database.insertTree(record);

Navigator.pop(context, true);

}

@override

void initState() {

super.initState();

\_getLocation();

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text('Daftar Pokok Baru')),

body: Padding(

padding: EdgeInsets.all(12),

child: Form(

key: \_formKey,

child: ListView(

children: [

TextFormField(

decoration: InputDecoration(labelText: 'Nama / ID'),

onChanged: (v) => name = v,

validator: (v) => v == null || v.isEmpty ? 'Sila isi' : null,

),

SizedBox(height: 8),

ListTile(

title: Text('Tarikh Semai: ${DateFormat('yyyy-MM-dd').format(seedDate)}'),

trailing: Icon(Icons.calendar\_today),

onTap: () async {

final picked = await showDatePicker(context: context, initialDate: seedDate, firstDate: DateTime(2000), lastDate: DateTime.now());

if (picked != null) setState(() => seedDate = picked);

},

),

DropdownButtonFormField<String>(

hint: Text('Pilih Rootstock'),

items: rootstockList.map((r) => DropdownMenuItem(child: Text(r), value: r)).toList(),

onChanged: (v) => setState(() => rootstock = v),

validator: (v) => v == null ? 'Pilih rootstock' : null,

),

SizedBox(height: 8),

imageFile == null

? TextButton.icon(onPressed: \_pickImage, icon: Icon(Icons.camera\_alt), label: Text('Ambil Gambar'))

: Image.file(imageFile!, height: 220),

SizedBox(height: 8),

ListTile(

title: Text(lat == null ? 'Mendapatkan lokasi...' : 'Lokasi: ${lat!.toStringAsFixed(5)}, ${lng!.toStringAsFixed(5)}'),

trailing: IconButton(icon: Icon(Icons.my\_location), onPressed: \_getLocation),

),

SizedBox(height: 12),

ElevatedButton(onPressed: \_save, child: Text('Simpan')),

],

),

),

),

);

}

}

class DetailPage extends StatelessWidget {

final TreeRecord tree;

DetailPage({required this.tree});

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text('Detail Pokok')),

body: Padding(

padding: EdgeInsets.all(12),

child: ListView(

children: [

if (tree.imagePath != null) Image.file(File(tree.imagePath!), height: 240),

SizedBox(height: 8),

Text('Nama: ${tree.name}', style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold)),

SizedBox(height: 6),

Text('Tarikh Semai: ${DateFormat('yyyy-MM-dd').format(tree.seedDate)}'),

SizedBox(height: 6),

Text('Rootstock: ${tree.rootstock}'),

SizedBox(height: 6),

Text('Lokasi: ${tree.lat != null ? tree.lat!.toStringAsFixed(5) : '-'}, ${tree.lng != null ? tree.lng!.toStringAsFixed(5) : '-'}'),

SizedBox(height: 12),

Text('QR untuk pokok (scan guna app lain):', style: TextStyle(fontWeight: FontWeight.w600)),

SizedBox(height: 8),

Center(

child: QrImage(

data: tree.id,

version: QrVersions.auto,

size: 200.0,

),

),

],

),

),

);

}

}

class ScanSimDialog extends StatefulWidget {

@override

\_ScanSimDialogState createState() => \_ScanSimDialogState();

}

class \_ScanSimDialogState extends State<ScanSimDialog> {

String code = '';

@override

Widget build(BuildContext context) {

return AlertDialog(

title: Text('Simulate Scan QR'),

content: TextField(

decoration: InputDecoration(hintText: 'Paste UUID QR di sini'),

onChanged: (v) => code = v,

),

actions: [

TextButton(onPressed: () => Navigator.pop(context), child: Text('Batal')),

ElevatedButton(onPressed: () => Navigator.pop(context, code), child: Text('OK')),

],

);

}

}

// Simple DB layer using sqflite

class AppDatabase {

final Database db;

AppDatabase.\_(this.db);

static Future<AppDatabase> open() async {

final databasesPath = await getDatabasesPath();

final path = join(databasesPath, 'rsw\_proto.db');

final db = await openDatabase(path, version: 1, onCreate: (d, v) async {

await d.execute('''

CREATE TABLE trees(

id TEXT PRIMARY KEY,

name TEXT,

seedDate INTEGER,

rootstock TEXT,

imagePath TEXT,

lat REAL,

lng REAL,

createdAt INTEGER

)

''');

});

return AppDatabase.\_(db);

}

Future<void> insertTree(TreeRecord t) async {

await db.insert('trees', t.toMap());

}

Future<List<TreeRecord>> getAllTrees() async {

final rows = await db.query('trees', orderBy: 'createdAt DESC');

return rows.map((r) => TreeRecord.fromMap(r)).toList();

}

Future<TreeRecord?> getTreeById(String id) async {

final rows = await db.query('trees', where: 'id = ?', whereArgs: [id]);

if (rows.isEmpty) return null;

return TreeRecord.fromMap(rows.first);

}

}

class TreeRecord {

final String id;

final String? name;

final DateTime seedDate;

final String rootstock;

final String? imagePath;

final double? lat;

final double? lng;

final DateTime createdAt;

TreeRecord({required this.id, this.name, required this.seedDate, required this.rootstock, this.imagePath, this.lat, this.lng, required this.createdAt});

Map<String, dynamic> toMap() => {

'id': id,

'name': name,

'seedDate': seedDate.millisecondsSinceEpoch,

'rootstock': rootstock,

'imagePath': imagePath,

'lat': lat,

'lng': lng,

'createdAt': createdAt.millisecondsSinceEpoch,

};

static TreeRecord fromMap(Map<String, dynamic> m) => TreeRecord(

id: m['id'] as String,

name: m['name'] as String?,

seedDate: DateTime.fromMillisecondsSinceEpoch(m['seedDate'] as int),

rootstock: m['rootstock'] as String,

imagePath: m['imagePath'] as String?,

lat: m['lat'] as double?,

lng: m['lng'] as double?,

createdAt: DateTime.fromMillisecondsSinceEpoch(m['createdAt'] as int),

);

}

/\*