import 'package:flutter/material.dart';

import 'package:Note\_App/screens/note\_list.dart';

import 'package:Note\_App/models/note.dart';

import 'package:sqflite/sqflite.dart';

import 'dart:async';

import 'dart:io';

import 'package:path\_provider/path\_provider.dart';

import 'package:Note\_App/models/note.dart';

import 'package:Note\_App/screens/edit\_note.dart';

import 'package:Note\_App/utils/database\_helper.dart';

import 'package:flutter/material.dart';

import 'package:Note\_App/models/note.dart';

import 'package:Note\_App/screens/edit\_note.dart';

import 'package:Note\_App/screens/view\_note.dart';

import 'package:Note\_App/utils/database\_helper.dart';

import 'package:flutter/material.dart';

import 'package:sqflite/sqflite.dart';

import 'package:Note\_App/models/note.dart';

import 'package:Note\_App/utils/database\_helper.dart';

import 'package:flutter/material.dart';

import 'package:intl/intl.dart';

class CreateNote extends StatefulWidget {

final Note note;

CreateNote(this.note);

@override

\_CreateNoteState createState() => \_CreateNoteState(note);

}

class \_CreateNoteState extends State<CreateNote> {

DatabaseHelper \_helper = DatabaseHelper();

Note note;

\_CreateNoteState(this.note);

TextEditingController \_titleController = new TextEditingController();

TextEditingController \_noteController = new TextEditingController();

@override

Widget build(BuildContext context) {

\_titleController.text = this.note.title;

\_noteController.text = this.note.content;

return WillPopScope(

onWillPop: () {

goToLastScreen();

return Future.value(false);

},

child: Scaffold(

appBar: AppBar(

title: Text('Note'),

actions: <Widget>[

FlatButton(

child: Icon(

Icons.done,

color: Colors.black,

),

onPressed: () {

setState(() {

if (note.id == null) {

\_saveToDatabase();

} else {

\_updateNoteToDatabase();

}

});

},

),

],

),

backgroundColor: Colors.white,

body: Padding(

padding: const EdgeInsets.all(8.0),

child: ListView(

children: <Widget>[

TextField(

controller: \_titleController,

maxLength: 256,

decoration: InputDecoration(

labelText: 'Title',

hintText: 'Put the Note Title',

),

onChanged: (value) {

updateTitle(value);

},

),

SizedBox(

height: 10.0,

),

Card(

elevation: 5.0,

child: Padding(

padding:

const EdgeInsets.only(top: 8.0, left: 8.0, right: 8.0),

child: TextField(

controller: \_noteController,

maxLines: null,

decoration: InputDecoration(

hintText: "Write a Note...",

),

onChanged: (value) {

updateContent(value);

},

),

),

),

SizedBox(

height: 10.0,

),

],

),

),

),

);

}

void \_showAlertDialog(String title, String message) {

AlertDialog alertDialog = AlertDialog(

title: Text(title),

content: Text(message),

);

showDialog(context: context, builder: (\_) => alertDialog);

}

void goToLastScreen() {

Navigator.pop(context, true);

}

\_saveToDatabase() async {

goToLastScreen();

note.date = DateFormat.yMMMd().format(DateTime.now());

int response;

if (note.id != null) {

} else {

response = await \_helper.insert(note);

}

if (response != 0) {

// \_showAlertDialog('Status', 'Note Saved');

} else {

\_showAlertDialog('Status', 'Unable to Save');

}

}

\_updateNoteToDatabase() {

goToLastScreen();

\_helper.update(note);

}

updateTitle(String value) {

note.title = value;

}

updateContent(String value) {

note.content = value;

}

}

class NoteList extends StatefulWidget {

@override

\_NoteListState createState() => \_NoteListState();

}

class \_NoteListState extends State<NoteList> {

Note note = Note('', '', '');

List<Note> noteList;

int \_count = 0;

DatabaseHelper \_helper = DatabaseHelper();

@override

Widget build(BuildContext context) {

if (noteList == null) {

noteList = List<Note>();

updateNoteListView();

}

return Scaffold(

appBar: AppBar(

title: Text('Notes'),

),

backgroundColor: Colors.white,

body: showNoteList(),

floatingActionButton: FloatingActionButton(

child: Icon(Icons.create),

tooltip: "Create Note",

onPressed: () {

goToNoteDetails();

},

),

);

}

void goToNoteDetails() async {

bool response = await Navigator.push(context,

MaterialPageRoute(builder: (context) => CreateNote(Note('', '', ''))));

if (response == true) {

updateNoteListView();

}

}

Widget showNoteList() {

var listView = ListView.builder(

itemCount: \_count,

itemBuilder: (context, index) {

return Card(

child: ListTile(

leading: Icon(

Icons.book,

color: Colors.black,

),

title: Text(

this.noteList[index].title,

style: TextStyle(fontFamily: 'amaranth', fontSize: 20.0),

),

subtitle: Text(

this.noteList[index].date,

style: TextStyle(

fontFamily: 'caveat',

fontWeight: FontWeight.bold,

fontSize: 20.0),

),

trailing: GestureDetector(

child: Icon(

Icons.delete,

color: Colors.blueGrey,

),

onTap: () {

\_delete(context, noteList[index]);

},

),

onTap: () {

\_viewNote(noteList[index]);

},

),

color: Colors.white,

elevation: 12.0,

);

},

);

return listView;

}

void updateNoteListView() {

final Future<Database> dbFuture = \_helper.initializeDatabase();

dbFuture.then((database) {

Future<List<Note>> noteListFuture = \_helper.getNoteList();

noteListFuture.then((noteList) {

setState(() {

this.noteList = noteList;

this.\_count = noteList.length;

});

});

});

}

void \_delete(BuildContext context, Note note) async {

int response = await \_helper.delete(note.id);

if (response != 0) {

updateNoteListView();

}

}

void \_viewNote(Note note) async {

bool response = await Navigator.push(

context, MaterialPageRoute(builder: (context) => ViewNote(note)));

if (response) {

updateNoteListView();

}

}

}

class ViewNote extends StatefulWidget {

final Note note;

ViewNote(this.note);

@override

\_ViewNoteState createState() => \_ViewNoteState(this.note);

}

class \_ViewNoteState extends State<ViewNote> {

DatabaseHelper \_helper = DatabaseHelper();

Note note;

\_ViewNoteState(this.note);

@override

Widget build(BuildContext context) {

AppBar appBar = AppBar(

leading: IconButton(

icon: Icon(Icons.arrow\_back),

onPressed: () {

goToLastScreen();

}),

title: Text('Notes'),

actions: <Widget>[

FlatButton(

child: Icon(

Icons.edit,

color: Colors.black,

),

onPressed: () {

setState(() {

\_editNote(context);

});

},

),

],

);

return WillPopScope(

onWillPop: () {

goToLastScreen();

return Future.value(false);

},

child: Scaffold(

appBar: appBar,

backgroundColor: Colors.white,

body: ListView(

children: <Widget>[

Card(

child: Container(

height: 75.0,

child: Padding(

padding: const EdgeInsets.only(top: 20.0, left: 8.0),

child: Text(

this.note.title,

style: TextStyle(fontFamily: 'Amaranth', fontSize: 20.0),

),

),

),

color: Colors.white,

elevation: 3.0,

),

SizedBox(

height: 20.0,

),

Container(

child: Padding(

padding: const EdgeInsets.all(8.0),

child: Text(

this.note.content,

style: TextStyle(

fontSize: 20.0,

fontFamily: 'redHat',

),

),

),

),

],

),

),

);

}

void updateNote() async {

Note note = await \_helper.getNoteById(this.note.id);

setState(() {

this.note = note;

});

}

void goToLastScreen() {

Navigator.pop(context, true);

}

void \_editNote(context) async {

bool response = await Navigator.push(context,

MaterialPageRoute(builder: (context) => CreateNote(this.note)));

if (response) {

print(response.toString());

updateNote();

}

}

}

class DatabaseHelper {

final databaseName = "notes.db";

final databaseVersion = 1;

String tableName = "notes\_table";

String colId = "id";

String colTitle = "title";

String colContent = "content";

String colDate = "date";

static DatabaseHelper \_databaseHelper;

static Database \_database;

DatabaseHelper.\_createInstance();

factory DatabaseHelper() {

if (\_databaseHelper == null) {

\_databaseHelper = DatabaseHelper.\_createInstance();

}

return \_databaseHelper;

}

Future<Database> get database async {

if (\_database == null) {

\_database = await initializeDatabase();

}

return \_database;

}

Future<Database> initializeDatabase() async {

Directory directory = await getApplicationDocumentsDirectory();

String path = directory.path + databaseName;

var notesDatabase =

openDatabase(path, version: databaseVersion, onCreate: \_createDb);

return notesDatabase;

}

void \_createDb(Database db, int version) async {

await db.execute(

("CREATE TABLE $tableName($colId INTEGER PRIMARY KEY AUTOINCREMENT, "

"$colTitle TEXT, $colContent TEXT, $colDate TEXT)"));

}

Future<List<Map<String, dynamic>>> getNotesListMap() async {

Database db = await this.database;

var response = db.query(tableName);

return response;

}

Future<int> insert(Note note) async {

Database db = await this.database;

print(note.objToMap());

int response = await db.insert(tableName, note.objToMap());

return response;

}

Future<int> update(Note note) async {

Database db = await this.database;

int response = await db.update(tableName, note.objToMap(),

where: '$colId = ?', whereArgs: [note.id]);

return response;

}

Future<int> delete(int noteId) async {

Database db = await this.database;

int response = await db.rawDelete('DELETE FROM $tableName WHERE $colId == $noteId');

return response;

}

Future<List<Note>> getNoteList() async {

var noteMapList = await getNotesListMap();

int count = noteMapList.length;

List<Note> noteList = List<Note>();

for(int i = 0 ; i < count ; i++) {

noteList.add(Note.mapToObj(noteMapList[i]));

}

return noteList;

}

Future<Note> getNoteById(int noteId) async {

Database db = await this.database;

List<Map<String, dynamic>> noteMap = await db.query(tableName, where: '$colId = ?', whereArgs: [noteId]);

Note note = Note.mapToObj(noteMap[0]);

return note;

}

}

class Note {

int \_id;

String \_title;

String \_content;

String \_date;

Note(this.\_title, this.\_content, this.\_date);

Note.withId(this.\_id, this.\_title, this.\_content, this.\_date);

int get id => \_id;

String get title => \_title;

String get content => \_content;

String get date => \_date;

set title(String title) {

this.\_title = title;

}

set content(String content) {

this.\_content = content;

}

set date(String date) {

this.\_date = date;

}

Map<String, dynamic> objToMap() {

Map<String, dynamic> mapObj = Map<String, dynamic>();

mapObj["id"] = this.id;

mapObj["title"] = this.title;

mapObj["content"] = this.content;

mapObj["date"] = this.date;

return mapObj;

}

Note.mapToObj(Map<String, dynamic> map) {

this.\_id = map['id'];

this.\_title = map['title'];

this.\_content = map['content'];

this.\_date = map['date'];

}

}

void main() {

runApp(Home());

}

class Home extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

debugShowCheckedModeBanner: false,

title: "Analytica Note App",

theme: ThemeData(

primarySwatch: Colors.teal,

),

home: NoteList(),

);

}

}