**Name: Satish Kumar Sharma**

**BITS ID: 2022MT93327**

Creating a Database experience in a Flutter app for both iOS and Android. sqflite plugin used to insert, read, update, and remove data about various Persons.

Plugin**: sqflite plugin, Flutter plugin**

Language: **Dart**

Tool: **Emulator, Android studio, Android SDK**,

I have uploaded code on my GitHub account as well. Please find link for your reference.

Video Demo link**:** [**https://www.youtube.com/watch?v=xDZd-Iwhm\_Y**](https://www.youtube.com/watch?v=xDZd-Iwhm_Y)

Code uploaded, My GitHub link: <https://github.com/iamsatishsharma/flutter_curd>

Please find the code as below, It includes mainly four classes.

**1. SQLHelper class**

**2. Person Model class**

**3. Main.dart class**

**4. Home Screen UI design class**

#1

**lib/sql\_helper.dart –** This class create to initiate DB and perform CURD operation.

**//===============START SQLHelper Class====================**

import 'dart:ffi';

import 'package:flutter/foundation.dart';

import 'package:sqflite/sqflite.dart';

import 'package:path/path.dart';

import './model/Persons.dart';

//This class created by Satish Sharma

//Date: 07-05-2023

//Flutter Sqlite CRUD Operation works with Example code

**//It’s just a SQLHelper Class making which we do in every OOP language.**

class SQLHelper {

**/\***

**getDataBase() function takes 3 arguments and gives us a database which we again return to our future function. String path of the Database Version onCreate Function**

**\*/**

**static String \_tableName = "persondetails"; //Table name**

**static Future<Database> getDataBase() async {**

**return openDatabase(**

**join(await getDatabasesPath(), "personsDatabase.db"),**

**onCreate: (db, version) async {**

**await db.execute(**

**"CREATE TABLE $\_tableName (id TEXT PRIMARY KEY, name TEXT, age TEXT, address TEXT)",**

**);**

**},**

**version: 1,**

**);**

**}**

**//=============== START CURD OPERATION============**

//====================================================

**//Insert/Add Method**

**// First Insert Function Which will take values and add them inside the Database**

static Future<int> **insert**(Person person) async {

int userId = 0;

Database db = await getDataBase();

String path = await getDatabasesPath();

print('''getDatabasesPath: ${path}''');

await db.insert( \_tableName, person.toMap(), conflictAlgorithm: ConflictAlgorithm.replace).then((value) {

userId = value;

});

return userId;

}

//====================================================

**//Get All Method: Fetch list of persons from the table.**

//After adding data then comes the part where we need data and show it into the UI.

//First there is a need for a database and then db.query method provides all the rows in the provided table inside the query() method.

static Future<List<Person>> **getAllPersons() async** {

Database db = await getDataBase();

List<Map<String, dynamic>> personsMap = await db.query(\_tableName);

print('''SQlite: ${ personsMap}''');

return List.generate(personsMap.length, (index) {

return Person(

id: personsMap[index]["id"],

name: personsMap[index]["name"],

age: personsMap[index]["age"],

address: personsMap[index]["address"]);

}); }

//================================================

**//Get method: Get one-person details**

//This will return a Single user by using Where in the SQL query. For this, we need to pass the userID while calling this function.

static Future<Person> **getPerson(String userId)async**{

Database db = await getDataBase();

List<Map<String, dynamic>> person = await db.rawQuery("SELECT \* FROM $\_tableName WHERE id = $userId");

if(person.length == 1){

return Person(

id: person[0]["id"],

name: person[0]["name"],

age: person[0]["age"],

address: person[0]["address"]);

} else {

return Person();

}

}

//======================================================

**//Update Method: Method to update a single person detail**

//Updating a particular person needs UserId and new values in the function when calling it. Here is the function for that below.

//Need to add an Update query and give it the table and updated values where id = userId.

static Future<void> **update(String userId, String name, String age, String address) async** {

Database db = await getDataBase();

db.rawUpdate("UPDATE $\_tableName SET name = '$name', age = '$age', address = '$address' WHERE id = '$userId'");

}

//====================================================

**//Delete method: Method to delete a person detailes from Table**

static Future<void**> deletePerson(String userId) async** {

Database db = await getDataBase();

await db.rawDelete("DELETE FROM $\_tableName WHERE id = '$userId'");

}

//==================== END =========================

}

**//==================END SQLHelper Class================**

#2

**//================START MAIN DART CLASS================**

import 'package:flutter/material.dart';

import 'package:flutter\_curd/view/home\_screen.dart';

import 'sql\_helper.dart';

import './model/Persons.dart';

/\*

//This class created by Satish Sharma

//Date: 07-05-2023

//Flutter Sqlite CRUD Operation works with Example code

\*/

void main() {

runApp(MyApp());

}

const darkBlueColor = Color(0xff486579); //App theme

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Enter Person details - CURD',

debugShowCheckedModeBanner: false,

theme: ThemeData(

// primaryColor: Colors.orange,

primarySwatch: Colors.blue,

),

home: const MyHomePage(title: 'Enter Person details - CURD'),

);

}

}

//Add home page

class MyHomePage extends StatefulWidget {

const MyHomePage({Key? key, required String title}) : super(key: key);

String get title => "Enter Person details - CURD";

@override

State<MyHomePage> createState() => MyHomePageState();

}

**//================END MAIN DART CLASS================**

#3

**//=============START PERSON MODEL CLASS==============**

**///Create the Modal Class of Person and their respective functions as shown below**

class Person {

final String? id;

final String? name;

final String? age;

final String? address;

const Person({ this.id, this.name, this.age, this.address});

//Alternate way to use model by const.

//const Person({ this.id, this.name, this.age, this.address });

factory Person.fromMap(Map<String, dynamic> json) =>

Person(id: json["id"], name: json["name"], age: json["age"], address: json["address"]);

Map<String, dynamic> toMap() {

return {

'id': id,

'name': name,

'age': age,

'address': address,

};

}

}

**//================END PERSON MODEL================**

#4

**//==============START HOME SCREEN DART=============**

Class Name:MyHomePageState.dart

import 'package:flutter/material.dart';

import '../main.dart';

import '../sql\_helper.dart';

import '../model/Persons.dart';

/\*

//This class created by Satish Sharma

//Date: 07-05-2023

//Flutter Sqlite CRUD Operation works with Example code

\*/

class MyHomePageState extends State<MyHomePage> {

//Initialize the text fields for the operations

var id = "";

final TextEditingController \_nameController = TextEditingController();

final TextEditingController \_ageController = TextEditingController();

final TextEditingController \_addressController = TextEditingController();

List<Person> \_journals = []; // Save the persons data

bool \_isLoading = true;

**// This function is used to fetch all data from the database**

void \_refreshPersonLists() async {

final data = await SQLHelper.getAllPersons();

setState(() {

\_journals = data;

\_isLoading = false;

});

}

// Loading the data when the app starts

@override

void initState() {

super.initState();

\_refreshPersonLists(); // Loading the data when the app starts

}

//Build

@override

Widget build(BuildContext context) {

return Scaffold(

backgroundColor: Colors.grey[200],

appBar: AppBar(

backgroundColor: Colors.blue,

title: Center(

child: Text(

widget.title,

style: const TextStyle(color: Colors.white),

),

),

),

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.start,

children: <Widget>[\_form(), \_listTitle(), \_personList()],

),

),

);

}

//other state properties and Start Form filling

final \_formKey = GlobalKey<FormState>();

\_form() => Container(

color: Colors.white,

padding: const EdgeInsets.symmetric(vertical: 10, horizontal: 30),

child: Form(

key: \_formKey,

child: Column(

children: <Widget>[

TextFormField(

controller: \_nameController,

decoration: const InputDecoration(labelText: 'Name'),

validator: (value) {

if (value == null || value.isEmpty) {

return 'Please enter name';

}}

),

TextFormField(

controller: \_ageController,

decoration: const InputDecoration(labelText: 'Age'),

validator: (value) {

if (value == null || value.isEmpty) {

return 'Please enter age';

}}

),

TextFormField(

controller: \_addressController,

decoration: const InputDecoration(labelText: 'Address'),

validator: (value) {

if (value == null || value.isEmpty) {

return 'Please enter address';

}}

),

Container(

margin: EdgeInsets.all(10.0),

child: ElevatedButton (

onPressed:() async {

FocusManager.instance.primaryFocus?.unfocus();

// Save new journal

if (\_formKey.currentState!.validate()) {

if( id == ""){

\_addItem();

}else {

\_updateItem(id);

id = "";

}

}

// Clear the text fields

\_nameController.text = '';

\_ageController.text = '';

\_addressController.text = '';

}, child:Text('Submit'),

),

),

],

),

),

);

//Person list title display here

\_listTitle() => Container( //apply margin and padding using Container Widget.

padding: const EdgeInsets.all(10), //You can use EdgeInsets like above

margin: const EdgeInsets.all(0),

child: const Text('Person list display here', style: TextStyle(

color: darkBlueColor, fontWeight: FontWeight.bold, fontSize: 20))

);

//Person list goes here

\_personList() => Expanded(

child: Card(

margin: const EdgeInsets.fromLTRB(10, 10, 10, 5),

shape: const RoundedRectangleBorder( //<-- SEE HERE

side: BorderSide(

color: Colors.grey,

),

),

child: Scrollbar(

child: ListView.builder(

padding: const EdgeInsets.all(5),

itemBuilder: (context, index) {

//List view title

if(index == 0) {

return Column(

// The header

children: <Widget> [ListTile(

leading: const Text('ID', style: TextStyle(fontWeight: FontWeight.bold),),

title: const Text('Name-(Age)', style: TextStyle(fontWeight: FontWeight.bold),),

trailing: SizedBox(

width: 100,

child: Row(

children: const [

Text(' Edit ', style: TextStyle(fontWeight: FontWeight.bold),),

Text('Delete', style: TextStyle(fontWeight: FontWeight.bold),) ],),

),

),

const Divider(height: 10.0, color: Colors.grey,),

\_listItemDisplay(index)

],

);

}

return Column(

children: <Widget>[

\_listItemDisplay(index),

Divider(height: 10.0,),

],

);

},

itemCount: \_journals.length,

),

),

),

);

// Item of the ListView

Widget \_listItemDisplay(index) {

return Container(

padding: const EdgeInsets.all(0),

decoration: const BoxDecoration(

border: Border(bottom: BorderSide(width: 1, color: Colors.black26))),

child: ListTile(

leading: Text(

\_journals[index].id.toString(),

textAlign: TextAlign.left,

style: const TextStyle(

color: darkBlueColor, fontWeight: FontWeight.bold),

),

title: Text(

"${\_journals[index].name} - (${\_journals[index].age})",

style: const TextStyle(

color: darkBlueColor, fontWeight: FontWeight.bold),

),

subtitle: Text(\_journals[index].address.toString()),

trailing: SizedBox(

width: 100,

child: Row(

children: [

IconButton(

icon: const Icon(Icons.edit),

onPressed: () async {

//update the details

\_nameController.text = \_journals[index].name.toString();

\_ageController.text = \_journals[index].age.toString();

\_addressController.text = \_journals[index].address.toString();

id = \_journals[index].id.toString();

}

),

IconButton(

icon: const Icon(Icons.delete),

onPressed: () =>

\_deleteItem(\_journals[index].id.toString()),

),

],

),

),

),

);

}

// Insert a new Person to the database

Future<void> \_addItem() async {

Person person = Person(id: (\_journals.length + 1).toString(), name: \_nameController.text, age: \_ageController.text, address: \_addressController.text);

await SQLHelper.insert(person);

ScaffoldMessenger.of(context).showSnackBar(const SnackBar(

content: Text('Successfully added a record!'),

));

\_refreshPersonLists();

}

// Update an existing journal

Future<void> \_updateItem(String id) async {

await SQLHelper.update(

id, \_nameController.text, \_ageController.text, \_addressController.text);

ScaffoldMessenger.of(context).showSnackBar(const SnackBar(

content: Text('Successfully updated a record!'),

));

//Refresh the person list

\_refreshPersonLists();

}

// Delete an item

void \_deleteItem(String id) async {

await SQLHelper.deletePerson(id);

ScaffoldMessenger.of(context).showSnackBar(const SnackBar(

content: Text('Successfully deleted a journal!'),

));

//Refresh the person list

\_refreshPersonLists();

}

//======================END==========================

}

**//==============END HOME SCREEN DART=============**

**I have uploaded code on my GitHub account. Please find link.**

**My GitHub link:** [**https://github.com/iamsatishsharma/flutter\_curd**](https://github.com/iamsatishsharma/flutter_curd)

**HAPPY CODING. THANK YOU.**