**Assignment 21**

Aim: Create a student database with Student table and perfrom CURD operations

Code:-

**main.dart**

import 'package:flutter/material.dart';

import 'package:dbpract/database\_helper.dart';

import 'package:flutter/material.dart';

void main() {

runApp(const MyApp());

}

class MyApp extends StatelessWidget {

const MyApp({Key? key}) : super(key: key);

// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Flutter Demo',

theme: ThemeData(

primarySwatch: Colors.blue,

),

home: const MyHomePage(title: 'Flutter Demo Home Page'),

);

}

}

class MyHomePage extends StatefulWidget {

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

final String title;

@override

State<MyHomePage> createState() => \_MyHomePageState();

}

class \_MyHomePageState extends State<MyHomePage> {

void \_incrementCounter() {

setState(() {

});

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text(widget.title),

),

body: Center(

child: Column(

children: [

ElevatedButton(onPressed: () async {

int i = await DatabaseHelper.instance.insert({

DatabaseHelper.columnname:'abc'

});

print('Inserted id is : $i');

}, child: Text('Add Records')),

ElevatedButton(onPressed: () async{

List<Map<String,dynamic>> rows = await DatabaseHelper.instance.showAll();

print(rows);

}, child: Text('Show Records')),

ElevatedButton(onPressed: () async{

int updatedId = await DatabaseHelper.instance.update({

DatabaseHelper.columnid: 2,

DatabaseHelper.columnname: 'harsha'

});

print('Updated $updatedId rows');

}, child: Text('Update')),

ElevatedButton(onPressed: () async{

int rowsaffected = await DatabaseHelper.instance.delete(2);

print(rowsaffected);

}, child: Text('Delete'))

],

),

)

);

}

}

**database\_helper.dart**

import "dart:io";

import "package:sqflite/sqflite.dart";

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

import 'package:path/path.dart';

class DatabaseHelper{

static final \_dbName = 'student.db';

static final \_dbVersion = 1;

static final \_tableName = 'student';

static final columnid = '\_id';

static final columnname ='name';

DatabaseHelper.\_privateConstructor();

static final DatabaseHelper instance = DatabaseHelper.\_privateConstructor();

static Database? \_database;

Future<Database?> get database async{

if (\_database!=null) return \_database;

\_database = await \_initiateDatabase();

return \_database;

/\* \_database ??=await \_initiateDatabase();

return \_database;\*/

}

\_initiateDatabase() async{

Directory directory = await getApplicationDocumentsDirectory();

String path = join(directory.path, \_dbName);

return await openDatabase(path, version: \_dbVersion, onCreate: \_onCreate);

}

Future \_onCreate(Database db, int version) async {

db.execute('''

CREATE TABLE $\_tableName(

$columnid INTEGER PRIMARY KEY,

$columnname TEXT NOT NULL

)

''');

}

Future<int> insert(Map<String, dynamic> row) async {

Database? db = await instance.database;

return await db!.insert(\_tableName, row );

}

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

Database? db = await instance.database;

return await db!.query(\_tableName);

}

Future<int> update(Map<String,dynamic> row) async{

Database? db = await instance.database;

int id = row[columnid];

return await db!.update(\_tableName, row, where:'$columnid = ?', whereArgs: [id]);

}

Future<int> delete(int id) async{

Database? db = await instance.database;

return await db!.delete(\_tableName, where: '$columnid=?', whereArgs: [id]);

}

}

Output:-



