Experiment 10

AIM:- Create and application Crud Operation with SQLite in Flutter.

Source :- Main.dart

Code :-

```
import 'package:flutter/material.dart'; import 'package:resetapi/sqlHelper.dart';

void main() { runApp(const MyApp());

}

class MyApp extends StatelessWidget {

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

@override

Widget build(BuildContext context) { return MaterialApp(

// Remove the debug banner debugShowCheckedModeBanner: false, title: 'SQLITE',

theme: ThemeData( primarySwatch: Colors.orange,

),

home: const HomePage());

}

}

class HomePage extends StatefulWidget {

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

@override

_HomePageState createState() => _HomePageState();

}

class _HomePageState extends State<HomePage> {
```

```
// All journals
List<Map<String, dynamic>> _journals = [];
bool _isLoading = true;
// This function is used to fetch all data from the database void _refreshJournals() async {
final data = await SQLHelper.getItems(); setState(() {
_journals = data;
_isLoading = false;
});
}
@override
void initState() {
super.initState();
_refreshJournals(); // Loading the diary when the app starts
}
final TextEditingController _titleController = TextEditingController();
final TextEditingController _descriptionController = TextEditingController();
// This function will be triggered when the floating button is pressed
// It will also be triggered when you want to update an item void _showForm(int? id) async {
if (id != null) {
// id == null -> create new item
// id != null -> update an existing item final existingJournal =
_journals.firstWhere((element) => element['id'] == id);
_titleController.text = existingJournal['title'];
_descriptionController.text = existingJournal['description'];
}
```

```
showModalBottomSheet( context: context, elevation: 5, isScrollControlled: true, builder: (_) => Container(
padding: EdgeInsets.only( top: 15,
left: 15,
right: 15,
// this will prevent the soft keyboard from covering the text fields bottom:
MediaQuery.of(context).viewInsets.bottom + 120,
),
child: Column(
mainAxisSize: MainAxisSize.min, crossAxisAlignment: CrossAxisAlignment.end, children: [
TextField(
controller: _titleController,
decoration: const InputDecoration(hintText: 'Title'),
),
const SizedBox( height: 10,
),
TextField(
controller: _descriptionController,
decoration: const InputDecoration(hintText: 'Description'),
),
const SizedBox( height: 20,
),
ElevatedButton( onPressed: () async {
// Save new journal
if (id == null) { await _addItem();
}
```

```
if (id != null) {
await _updateItem(id);
}
// Clear the text fields
_titleController.text = '';
_descriptionController.text = '';
)
],
),
));
}// Close the bottom sheet Navigator.of(context).pop();
},
child: Text(id == null ? 'Create New' : 'Update'),
// Insert a new journal to the database Future<void> _addItem() async { await SQLHelper.createItem(
_titleController.text, _descriptionController.text);
_refreshJournals();
}
// Update an existing journal Future<void> _updateItem(int id) async { await SQLHelper.updateItem(
id, _titleController.text, _descriptionController.text);
_refreshJournals();
}
// Delete an item
void _deleteItem(int id) async { await SQLHelper.deleteItem(id);
ScaffoldMessenger.of(context).showSnackBar(const SnackBar( content: Text('Successfully deleted a
journal!'),
```

```
));
_refreshJournals();
}
@override
Widget build(BuildContext context) { return Scaffold(
appBar: AppBar(
title: const Text('SQL'),
),
body: _isLoading
? const Center(
child: CircularProgressIndicator(),
)
: ListView.builder( itemCount: _journals.length,
itemBuilder: (context, index) => Card( color: Colors.orange[200],
margin: const EdgeInsets.all(15), child: ListTile(
title: Text(_journals[index]['title']),
subtitle: Text(_journals[index]['description']), trailing: SizedBox(
width: 100, child: Row( children: [
IconButton(
icon: const Icon(Icons.edit),
onPressed: () => _showForm(_journals[index]['id']),
),
IconButton(
icon: const Icon(Icons.delete), onPressed: () =>
_deleteItem(_journals[index]['id']),
```

),],),)),),

), floatingActionButton: FloatingActionButton(child: const Icon(Icons.add),

onPressed: () => _showForm(null),

),

);

}

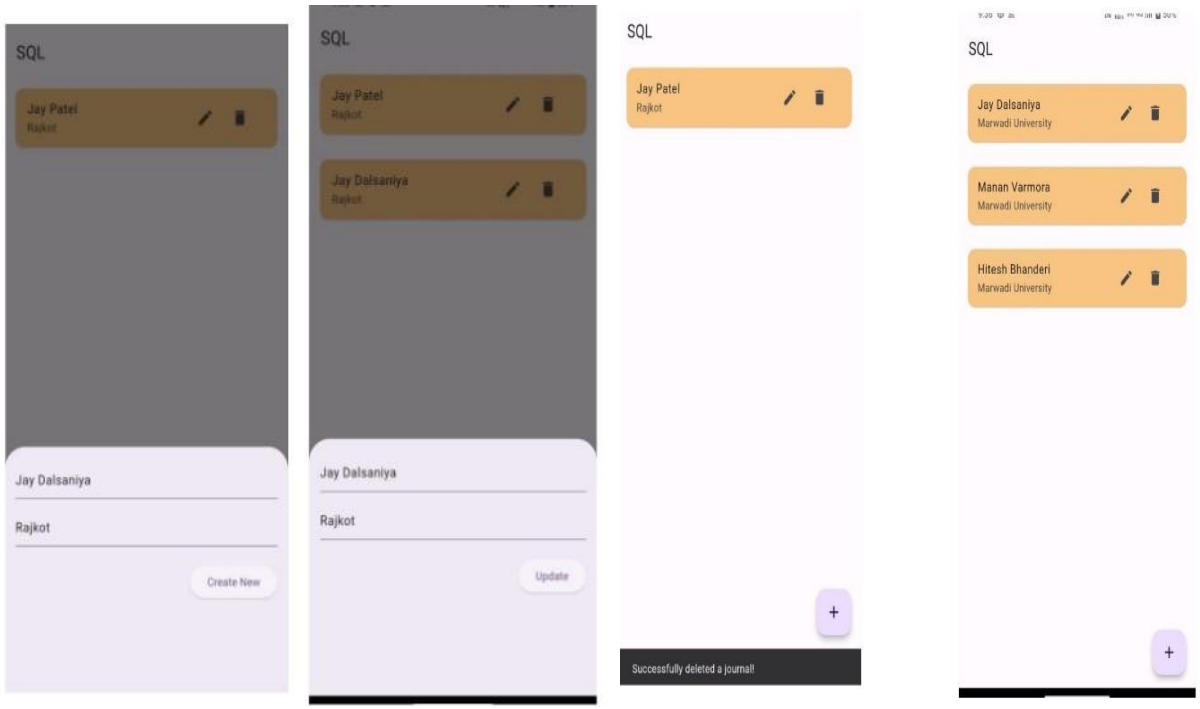
}

Source :- sqlHelper

```
import 'package:flutter/foundation.dart'; import 'package:sqflite/sqflite.dart' as sql;
class SQLHelper {
static Future<void> createTables(sql.Database database) async { await database.execute("""CREATE
TABLE items(
id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
title TEXT, description TEXT,
createdAt TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
)
""");
}
// id: the id of a item
// title, description: name and description of your activity
// created_at: the time that the item was created. It will be automatically handled by SQLite
static Future<sql.Database> db() async { return sql.openDatabase(
'dbtech.db', version: 1,
onCreate: (sql.Database database, int version) async { await createTables(database);
},
);
}
// Create new item (journal)
static Future<int> createItem(String title, String? descrption) async { final db = await SQLHelper.db();
final data = {'title': title, 'description': descrption}; final id = await db.insert('items', data,
conflictAlgorithm: sql.ConflictAlgorithm.replace); return id;
}
// Read all items (journals)
static Future<List<Map<String, dynamic>>> getItems() async { final db = await SQLHelper.db();
return db.query('items', orderBy: "id");
}
// Read a single item by id
// The app doesn't use this method but I put here in case you want to see it static Future<List<Map<String,
dynamic>>> getItem(int id) async {
final db = await SQLHelper.db()
```

```
return db.query('items', where: "id = ?", whereArgs: [id], limit: 1);
}
// Update an item by id
static Future<int> updateItem(
int id, String title, String? descrption) async { final db = await SQLHelper.db();
final data = { 'title': title,
'description': descrption,
'createdAt': DateTime.now().toString()
};
final result =
await db.update('items', data, where: "id = ?", whereArgs: [id]); return result;
}
// Delete
static Future<void> deleteItem(int id) async { final db = await SQLHelper.db();
try {
await db.delete("items", where: "id = ?", whereArgs: [id]);
} catch (err) {
debugPrint("Something went wrong when deleting an item: $err");
}
}
}
```

Output :-



Experiment 11

AIM:- Create an application Connecting to REST API in Flutter.

Source :- Main.dart

Code :-

```
import 'package:flutter/material.dart';

import 'package:resetapi/data_screen.dart';

void main() { runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

debugShowCheckedModeBanner: false,

title: 'Flutter REST API Demo',

theme: ThemeData(

primarySwatch: Colors.blue,

),

home: DataScreen(),

);

}

}
```