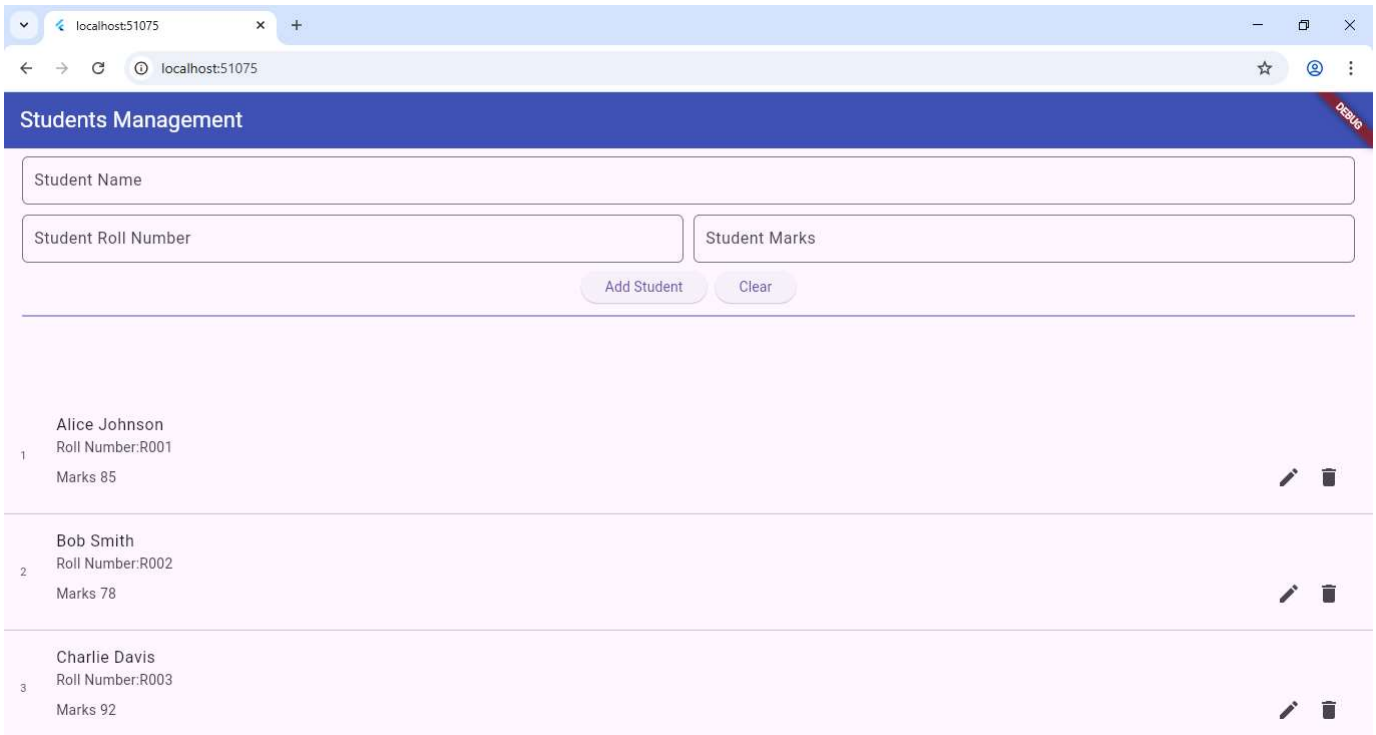
This Documentation is based on Student Management App which consist a student model with following parameters and UI

```dart
class StudentModel {
  final String id;
  String studentName;
  String studentRollNumber;
  String studentMarks;

  StudentModel({
    required this.id,
    required this.studentName,
    required this.studentRollNumber,
    required this.studentMarks,
  });
}
```

Step 1: Design the UI and Model Class



After completing the UI

Step 2: Installing dependencies

https://pub.dev/

search and install the dependencies

hive_flutter
hive_generator
build_runner

Step 3: Close and restart the IDE(VS Code)

Step4: Create a folder and dart file in lib => Core/core.dart

This is where we write commonly accessible data. In this project students data is being held in a List<StudentModel>. So the list is declared and initialized here. Since we are using stateless widget, the list should be of type ValueNotifier.

Step 5: Create a folder and dart file in lib => Infrastructure/db_functions.dart

This is where we write the code to create, read and write tables in db.

Step 6: Change in the StudentModel to create an adapter

Adapter is created to exchange data between the List and Hive Database. It's a middleware code, which converts list data to hive compactible and vice versa.

5.1 Annotate the Student Model Class as below

```
@HiveType(typeId: 1)
class StudentModel {
  @HiveField(0)
  final String id;
  @HiveField(1)
  String studentName;
  @HiveField(2)
  String studentRollNumber;
  @HiveField(3)
  String studentMarks;

  StudentModel({
    required this.id,
    required this.studentName,
    required this.studentRollNumber,
    required this.studentMarks,
  });
}
```

Where

1. @HiveType(typeId: 1) indicates the identifier for the table in the Hive DB
2. @HiveField(0) indicates the field name

5.2 Run build runner code in the terminal to generate the adapter

flutter packages pub run build_runner watch --use-polling-watcher --delete-conflicting-outputs

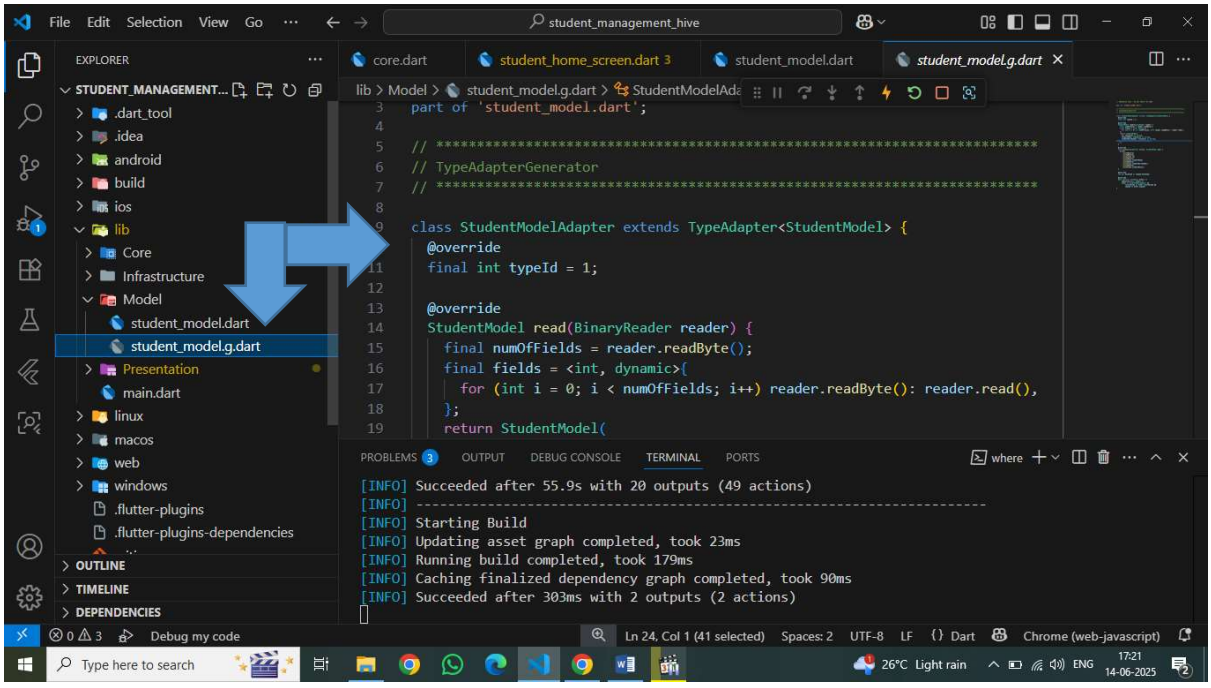
at the end of this script execution you will be asked add part 'student_model.g.dart'; in your student_model.dart file as give below and then save.

```
import 'package:hive_flutter/adapters.dart';
part 'student_model.g.dart';

@HiveType(typeId: 1)
class StudentModel {
  @HiveField(0)
  final String id;
```

Once saved, the adapter will be generated in the Model folder as student_model.g.dart

Which contains the code to transfer data between flutter list and hive database



Step 6: Register the hive data base at the beginning of the project

6.1 The registration process is done in the main, and since this is a time consuming process, make main an asynchronous function

```
Future<void> main() async{
  runApp(const MainApp());
}
```

6.2 use the code

```
  WidgetsFlutterBinding.ensureInitialized();
```

As the first line in the main. This will ensure that all input output streams of flutter is initialized before starting the project

6.2 Initialize the Hive(use await keyword to ensure that this step must be completed before going to the below codes)

```
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Hive.initFlutter();
```

Check whether hive with typeId is registered. If not register else do nothing.

```
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Hive.initFlutter();
  if (!Hive.isAdapterRegistered(StudentModelAdapter().typeId)) {
    Hive.registerAdapter(StudentModelAdapter());
  }
  runApp(const MainApp());
}
```

Step 7: Now write functions to read, write and update hive database in Infrastructure/db_functions.dart

7.1 Write a function to getAll students when the UI is loading

```
void getAllStudent() async {
  final result = await Hive.openBox<StudentModel>('student_db');
  studentsListNotifier.value.clear();
  studentsListNotifier.value.addAll(result.values);
  studentsListNotifier.notifyListeners();
}
```

And call this function in the corresponding screens Widget build(BuildContext context) function

```
Widget build(BuildContext context) {
    getAllStudent();
    return Scaffold(
```

7.2 Add Student Function

```
addStudent(StudentModel s) async {
  final result = await Hive.openBox<StudentModel>('student_db');
  final id = await result.add(s);
  s.id = id.toString();
  await result.put(id, s);

  getAllStudent();
}
```

After add operation, it will return an int value which is the primary key of the current added row.

getAllStudents is called every time when a change occurs to the DB to refresh the UI

```
void deleteAllStudents() async {
  final result = await Hive.openBox<StudentModel>('student_db');
  await result.clear();
  getAllStudent();
}

void deleteStudent(int id) async {
  final result = await Hive.openBox<StudentModel>('student_db');
  await result.delete(id);
  getAllStudent();
}
```

```
void editStudent(StudentModel s) async {
  int id = int.parse(s.id);
  final result = await Hive.openBox<StudentModel>('student_db');
  await result.put(id, s);
  getAllStudent();
}
```