### 1. ****Setting Up Firebase****

**Create a Firebase Project:**

* 1. Go to the Firebase Console.
  2. Click on "Add project" and follow the setup steps.

**Add Your App to Firebase:**

For Android:

* + 1. Click on the Android icon.
    2. Register your app with the package name (e.g., com.example.studentfiredata).
    3. Download the google-services.json file and place it in your Android app directory under android/app/.

For iOS:

* + 1. Click on the iOS icon.
    2. Register your app with the iOS bundle ID (e.g., com.example.studentfiredata).
    3. Download the GoogleService-Info.plist file and place it in your iOS app directory under ios/Runner/.

**Enable Firestore in Firebase:**

* 1. Go to **Firestore Database** in the Firebase Console.
  2. Click on "Create database" and select the mode (start in test mode or locked mode). For development, you might start in test mode and later switch to locked mode with proper security rules.

### 2. ****Add Firebase Dependencies to Flutter****

In your pubspec.yaml file, add the following dependencies:

yaml

Copy code

dependencies:

flutter:

sdk: flutter

firebase\_core: ^2.7.0

cloud\_firestore: ^5.3.0

Run flutter pub get to install these packages.

### 3. ****Configure Firebase in Flutter****

**Initialize Firebase:** Ensure that Firebase is initialized in your main.dart file:

dart

Copy code

import 'package:firebase\_core/firebase\_core.dart';

import 'firebase\_options.dart'; // Your Firebase options file

void main() async {

WidgetsFlutterBinding.ensureInitialized();

await Firebase.initializeApp(

options: DefaultFirebaseOptions.currentPlatform,

);

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Student Management System',

theme: ThemeData(

primarySwatch: Colors.blue,

),

home: StudentManagementScreen(),

);

}

}

**Create Firestore Service:** Create a firestore\_service.dart file to handle Firestore operations:

dart

Copy code

import 'package:cloud\_firestore/cloud\_firestore.dart';

class FirestoreService {

final FirebaseFirestore \_db = FirebaseFirestore.instance;

final String collection = 'students';

Future<void> addStudent(String name, int age) async {

await \_db.collection(collection).add({

'name': name,

'age': age,

});

}

Future<void> updateStudent(String id, String name, int age) async {

await \_db.collection(collection).doc(id).update({

'name': name,

'age': age,

});

}

Future<void> deleteStudent(String id) async {

await \_db.collection(collection).doc(id).delete();

}

Stream<List<Student>> getStudents() {

return \_db.collection(collection).snapshots().map((snapshot) =>

snapshot.docs.map((doc) => Student.fromFirestore(doc.data(), doc.id)).toList());

}

}

class Student {

final String id;

final String name;

final int age;

Student({required this.id, required this.name, required this.age});

factory Student.fromFirestore(Map<String, dynamic> firestore, String id) {

return Student(

id: id,

name: firestore['name'],

age: firestore['age'],

);

}

}

### 4. ****Create the UI for CRUD Operations****

Create a student\_management\_screen.dart file to implement CRUD operations and UI:

dart

Copy code

import 'package:flutter/material.dart';

import 'firestore\_service.dart';

class StudentManagementScreen extends StatefulWidget {

@override

\_StudentManagementScreenState createState() => \_StudentManagementScreenState();

}

class \_StudentManagementScreenState extends State<StudentManagementScreen> {

final \_nameController = TextEditingController();

final \_ageController = TextEditingController();

final \_searchController = TextEditingController();

final FirestoreService \_firestoreService = FirestoreService();

String? \_selectedStudentId;

void \_addStudent() {

final name = \_nameController.text;

final age = int.tryParse(\_ageController.text) ?? 0;

if (name.isNotEmpty && age > 0) {

\_firestoreService.addStudent(name, age);

\_nameController.clear();

\_ageController.clear();

}

}

void \_updateStudent() {

final name = \_nameController.text;

final age = int.tryParse(\_ageController.text) ?? 0;

if (\_selectedStudentId != null && name.isNotEmpty && age > 0) {

\_firestoreService.updateStudent(\_selectedStudentId!, name, age);

\_nameController.clear();

\_ageController.clear();

setState(() {

\_selectedStudentId = null;

});

}

}

void \_deleteStudent(String id) {

\_firestoreService.deleteStudent(id);

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Student Management System'),

),

body: Padding(

padding: const EdgeInsets.all(16.0),

child: Column(

children: <Widget>[

Row(

children: <Widget>[

Expanded(

child: TextField(

controller: \_nameController,

decoration: InputDecoration(labelText: 'Name'),

),

),

SizedBox(width: 16),

Expanded(

child: TextField(

controller: \_ageController,

keyboardType: TextInputType.number,

decoration: InputDecoration(labelText: 'Age'),

),

),

SizedBox(width: 16),

ElevatedButton(

onPressed: \_selectedStudentId == null ? \_addStudent : \_updateStudent,

child: Text(\_selectedStudentId == null ? 'Add Student' : 'Update Student'),

),

],

),

SizedBox(height: 16),

TextField(

controller: \_searchController,

decoration: InputDecoration(labelText: 'Search by Name'),

onChanged: (value) => setState(() {}),

),

SizedBox(height: 16),

Expanded(

child: StreamBuilder<List<Student>>(

stream: \_searchController.text.isEmpty

? \_firestoreService.getStudents()

: \_firestoreService.searchStudents(\_searchController.text),

builder: (context, snapshot) {

if (snapshot.connectionState == ConnectionState.waiting) {

return Center(child: CircularProgressIndicator());

}

if (!snapshot.hasData || snapshot.data!.isEmpty) {

return Center(child: Text('No students found.'));

}

final students = snapshot.data!;

return DataTable(

columns: [

DataColumn(label: Text('ID')),

DataColumn(label: Text('Name')),

DataColumn(label: Text('Age')),

DataColumn(label: Text('Actions')),

],

rows: students.map((student) {

return DataRow(

cells: [

DataCell(Text(student.id)),

DataCell(Text(student.name)),

DataCell(Text(student.age.toString())),

DataCell(

Row(

children: [

IconButton(

icon: Icon(Icons.edit),

onPressed: () {

setState(() {

\_selectedStudentId = student.id;

\_nameController.text = student.name;

\_ageController.text = student.age.toString();

});

},

),

IconButton(

icon: Icon(Icons.delete),

onPressed: () => \_deleteStudent(student.id),

),

],

),

),

],

);

}).toList(),

);

},

),

),

],

),

),

);

}

}