

Aim: To integrate Firebase Firestore with a Flutter application in order to store, retrieve, and manage user data in real time using a cloud-based NoSQL database.

Theory: Firebase Firestore is a scalable, cloud-based NoSQL database that allows mobile and web applications to store and sync data in real-time. In Flutter, this is accomplished using the `cloud_firestore` plugin which provides an API to perform Create, Read, Update, and Delete (CRUD) operations. Firestore organizes data in documents and collections, making it ideal for structured and semi-structured data.

Firestore allows multiple users to see live changes without reloading the app, making it an excellent choice for apps that require collaboration, chat features, or dynamic updates. The integration includes Firebase Authentication for user management and Firestore for data storage.

Steps to Perform:

1. Create and Configure Firebase Project:

- Visit Firebase Console.
- Create a new project, name it, and disable Google Analytics if not needed.
- Click on "Add App" → select Flutter/Android/iOS as per your platform.
- Register your app and download the `google-services.json` or `GoogleService-Info.plist` file.

2. Add Firebase SDK and Plugins:

Add the Firebase and Firestore dependencies in `pubspec.yaml`:

dependencies:

`firebase_core: ^2.0.0`

`cloud_firestore: ^4.0.0`

-
- Run `flutter pub get` to install the packages.

3. Initialize Firebase in `main.dart`:

Import required packages:

```
import 'package:firebase_core/firebase_core.dart';
```

○

Modify `main()`:

```
void main() async {  
  WidgetsFlutterBinding.ensureInitialized();  
  await Firebase.initializeApp();  
  runApp(MyApp());  
}
```

○

4. CRUD Operations with Firestore:

Create/Insert Data:

```
FirebaseFirestore.instance.collection('users').add({  
  'name': 'Alice',  
  'email': 'alice@example.com',  
});
```

○

Read Data (StreamBuilder):

```
StreamBuilder(  
  stream: FirebaseFirestore.instance.collection('users').snapshots(),  
  builder: (context, snapshot) {  
    if (!snapshot.hasData) return CircularProgressIndicator();  
    return ListView(  
      children: snapshot.data!.docs.map((doc) => Text(doc['name'])).toList(),  
    );  
  },  
);
```

○

Update Data:

```
FirebaseFirestore.instance.collection('users').doc(docId).update({  
  'email': 'new@example.com',  
});
```

○

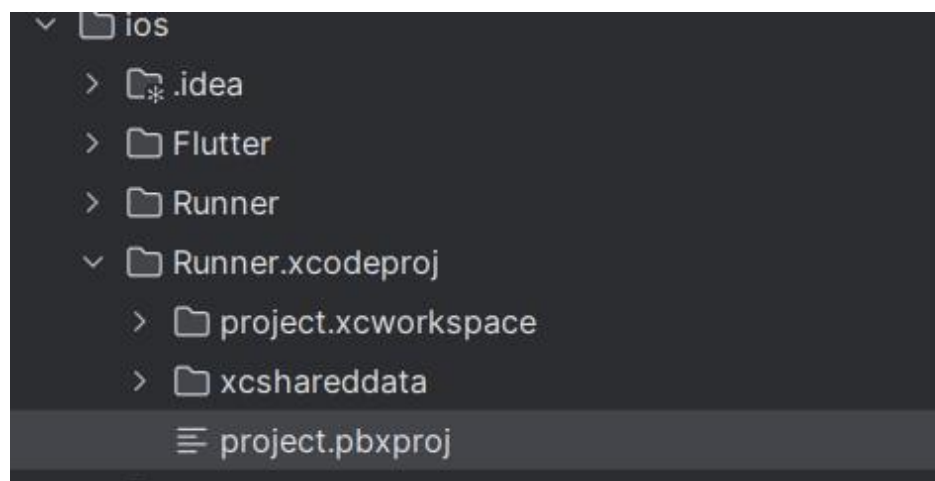
Delete Data:

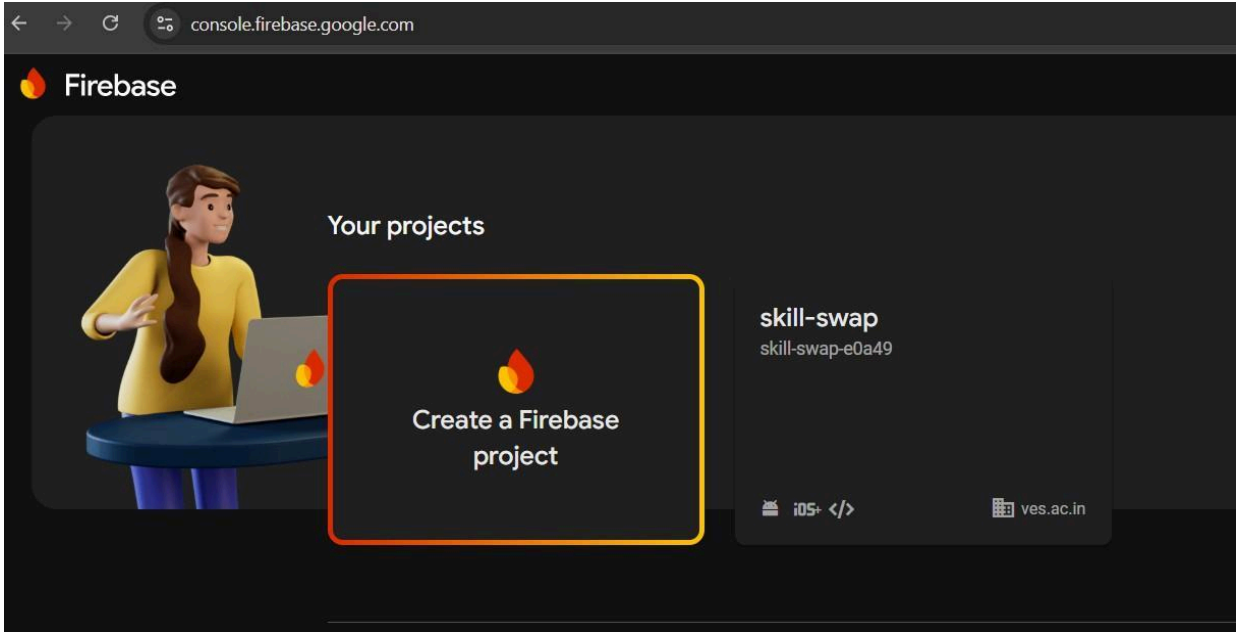
```
FirebaseFirestore.instance.collection('users').doc(docId).delete();
```

○

Key Features:

- **Real-Time Syncing:** Data changes instantly reflect across connected clients.
- **Cloud-Hosted and Scalable:** Ideal for apps of all sizes.
- **Simple Integration:** Seamless with Firebase Authentication and Storage.
- **Structured Collections/Documents:** Easy data modeling.
- **Offline Persistence:** Works offline and syncs when reconnected.

















skill-swap

Spark plan

Getting started

1 app visible (max 3)

4 apps in project

		skill_swap (android) com.example.skill_swap	
		com.example.skillSwap	
		skill_swap (web) Web App	
		skill_swap (windows) Web App	

Note: all apps are included in project-level metrics below, but only selected apps above are broken out