

## **Experiment 6**

**Aim:** To Connect Flutter UI with fireBase database

### **Theory:**

Firebase provides a powerful backend solution for Flutter applications, offering two main database options: Cloud Firestore and Realtime Database. These databases enable seamless data synchronization between the app and the cloud, supporting real-time updates and offline capabilities.

### **Firebase Database Options**

- Cloud Firestore: A NoSQL document database that stores data in collections and documents, ideal for complex, hierarchical data with powerful querying capabilities.
- Realtime Database: A low-latency JSON-based database that syncs data in real-time, suitable for applications requiring instant updates (e.g., chat apps, live feeds).

### **Key Features**

- Real-Time Synchronization: Both databases support live data updates, ensuring the UI reflects changes immediately.
- Offline Persistence: Firebase caches data locally, allowing apps to function without an internet connection and sync when online.
- Scalability: Firestore automatically scales to handle large datasets, while the Realtime Database is optimized for speed and simplicity.

### **Firebase-Flutter Integration**

- Setup Process:
  - Add required dependencies (firebase\_core, cloud\_firestore, or firebase\_database).
  - Configure Firebase in the Flutter project using platform-specific files (google-services.json for Android, GoogleService-Info.plist for iOS).
  - Initialize Firebase in the app's main entry point.
- Authentication (Optional): Secure data access by integrating Firebase Auth, ensuring only authorized users can read/write data.

### **CRUD Operations**

- Create: Add new documents (Firestore) or nodes (Realtime Database).
- Read: Fetch data using queries, snapshots, or real-time listeners.
- Update: Modify existing records with transaction support for consistency.

- Delete: Remove data while maintaining referential integrity.

### **Real-Time Data Handling**

- StreamBuilder: A Flutter widget that listens to Firestore streams and rebuilds the UI when data changes.
- State Management: Combine Firebase with Provider, Riverpod, or Bloc for efficient state handling in complex apps.

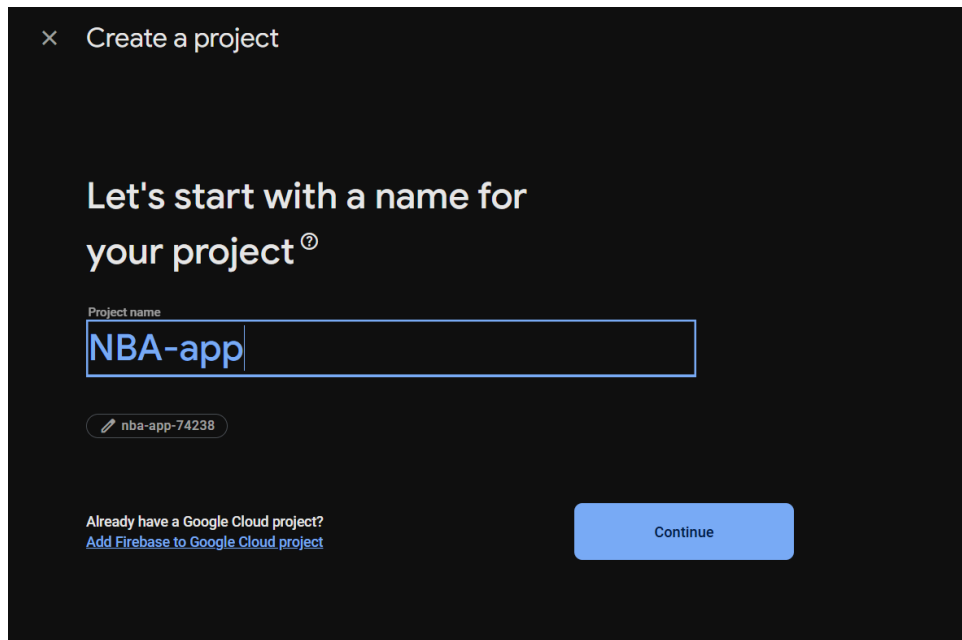
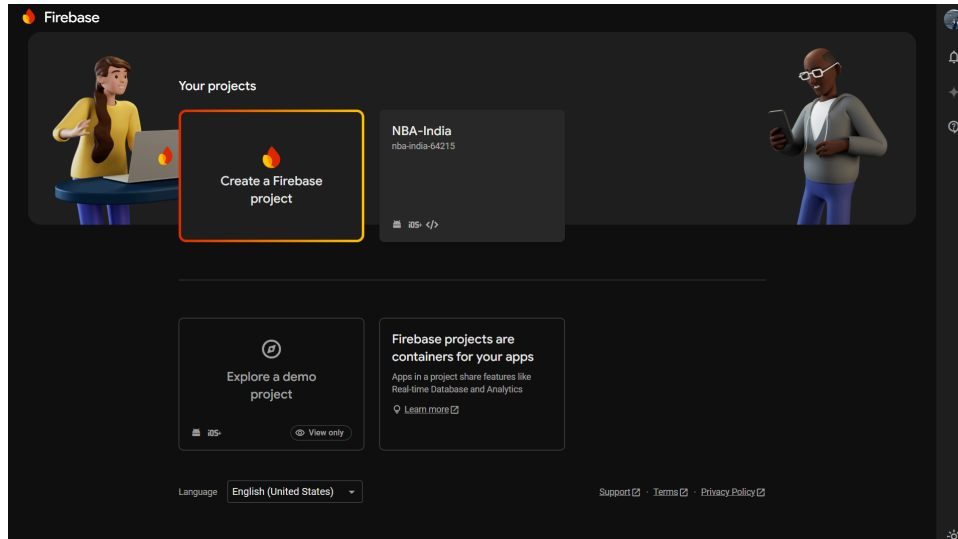
### **Security & Rules**

- Firestore Security Rules: Define granular access controls using Firebase's rule language (e.g., restricting writes to authenticated users).
- Realtime Database Rules: Similar to Firestore but structured as JSON for role-based permissions.

### **Steps:**

#### **1. Create a firebase project.**

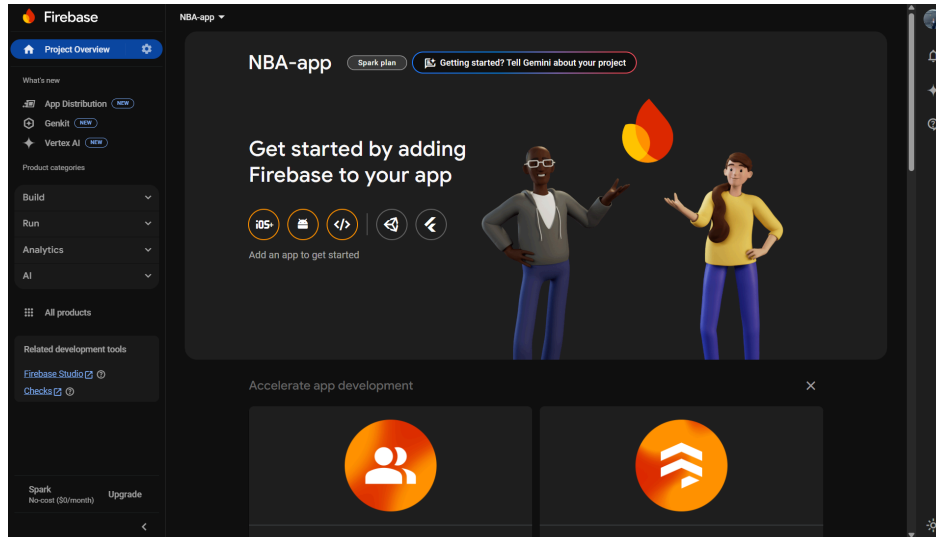
First, log in with your Google account to manage your Firebase projects. From within the Firebase dashboard, select the Create new project button and give it a name:



Click on continue, next it asks to enable Gemini, it can be helpful when we are stuck and are unable to navigate or use the firebase features properly.

Further it will ask to enable google analytics. We do not require this for this project, but if you do, simply check the box.

This is the console of our newly created project.



## 2. Android support

On the dashboard, you will see the android logo, click on that to register an android app.

[illegible]

You will get the Android package name from the app level build.gradle file(android -> app -> build.gradle)

```
android {  
    namespace = "com.example.nba"  
    compileSdk = flutter.compileSdkVersion  
    ndkVersion = flutter.ndkVersion  
}
```

### × Add Firebase to your Android app


✓ Register app  
Android package name: com.example.nba, App nickname: NBA India

2 Download and then add config file [Instructions for Android Studio below](#) | [Unity](#) [C++](#)

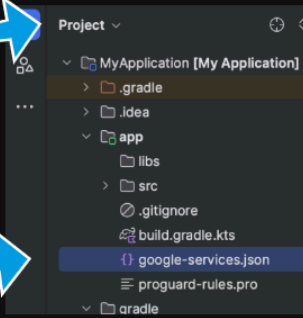
[Download google-services.json](#)

Switch to the Project view in Android Studio to see your project root directory.

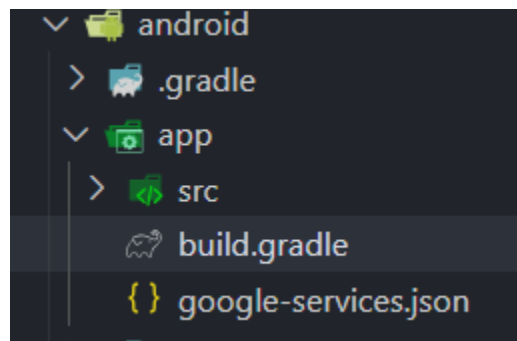
Move your downloaded `google-services.json` file into your module (app-level) root directory.

  
google-services.json

[Next](#)



Download the config file from the button and save in the app directory(android -> app) of your project.



3

Add Firebase SDK

Instructions for Gradle | [Unity](#) [C++](#)

★ Are you still using the `buildscript` syntax to manage plugins? Learn how to [add Firebase plugins](#) using that syntax.

1. To make the `google-services.json` config values accessible to Firebase SDKs, you need the Google services Gradle plugin.

☐ Kotlin DSL (`build.gradle.kts`) ☒ Groovy (`build.gradle`)

Add the plugin as a dependency to your project-level `build.gradle` file:

Root-level (project-level) Gradle file (`<project>/build.gradle`):

```
plugins {
    // ...

    // Add the dependency for the Google services Gradle plugin
    id 'com.google.gms.google-services' version '4.4.2' apply false
}
```

2. Then, in your module (app-level) `build.gradle` file, add both the `google-services` plugin and any Firebase SDKs that you want to use in your app:

Module (app-level) Gradle file (`<project>/<app-module>/build.gradle`):

```
plugins {
    id 'com.android.application'
    // Add the Google services Gradle plugin
    id 'com.google.gms.google-services'
    ...
}

dependencies {
    // Import the Firebase BoM
    implementation platform('com.google.firebase:firebase-bom:33.12.0')

    // TODO: Add the dependencies for Firebase products you want to use
    // When using the BoM, don't specify versions in Firebase dependencies
    // https://firebase.google.com/docs/android/setup#available-libraries
}
```

By using the Firebase Android BoM, your app will always use compatible Firebase library versions. [Learn more](#)

To add the firebase sdk, select Groovy (we are groovy and not kotlin). Add the provided line of codes to you respective files.

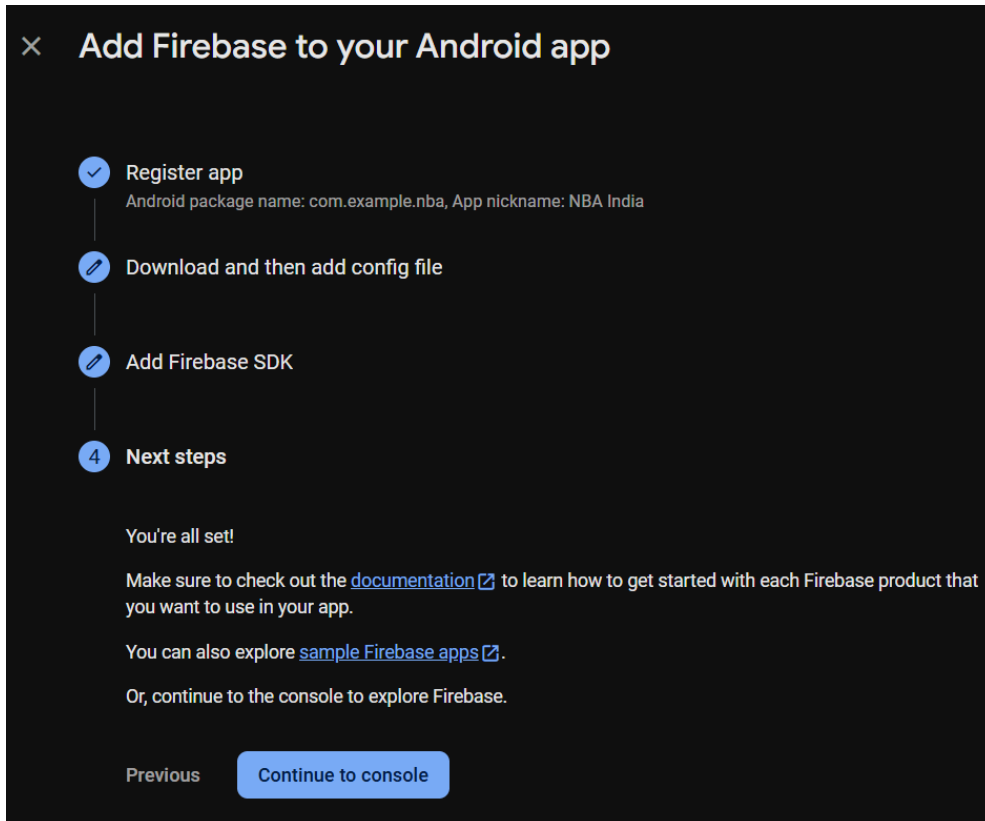
android -> build.gradle:

```
plugins {
    // Add the dependency for the Google services Gradle plugin
    id 'com.google.gms.google-services' version '4.4.2' apply false
}
```

android -> app -> build.gradle

```
plugins {  
    id "com.android.application"  
    id 'com.google.gms.google-services'  
    id "dev.flutter.flutter-gradle-plugin"  
}
```

Firebase has now been added to the project. Click on continue to console to use different options available with firebase



**Authorization and firestore:**

Search by email address, phone number, or user UID					Add user		
Identifier ↓	Providers	Created ↓	Signed In	User UID			
rakshit.honey602@gma...	✉	Apr 15, 2025	Apr 15, 2025	SrzOF13lqdP1SStrbP6ZfZuVG...			
2022.rakshit.sharma@v...	✉	Apr 13, 2025	Apr 15, 2025	ut07t5vvr1hySGiBRZjcKblEVa...			
					Rows per page:	50	1 – 2 of 2

🏠 > users > SrzOF13lqdP1S...			☁ More in Google Cloud	
🏠 (default)	📁 users	☰	📄 SrzOF13lqdP1SStrbP6ZfZuVGL23	⋮
+ Start collection	+ Add document	+ Start collection		
users >	SrzOF13lqdP1SStrbP6ZfZuVGL23 >	+ Add field		
		createdAt: April 15, 2025 at 5:23:28PM UTC+5:30		
		email: "rakshit.honey602@gmail.com"		
		favTeam: ""		
		favTeams		
		0 "DEN"		
		1 "GS"		
		2 "LAL"		
		3 "MIL"		
		lastUpdated: April 15, 2025 at 5:23:44PM UTC+5:30		
		name: "Rakshit Sharma"		

Github link: <https://github.com/Rakshit5467/NBA-India>

### Conclusion:

This experiment successfully demonstrated the integration of Firebase database with a Flutter application, covering project setup, Android configuration, and basic CRUD operations. By implementing Firestore/Realtime Database, we achieved real-time data synchronization and offline persistence, while Firebase security rules ensured data protection. The experiment provides a foundation for building scalable, data-driven Flutter apps with Firebase backend services.