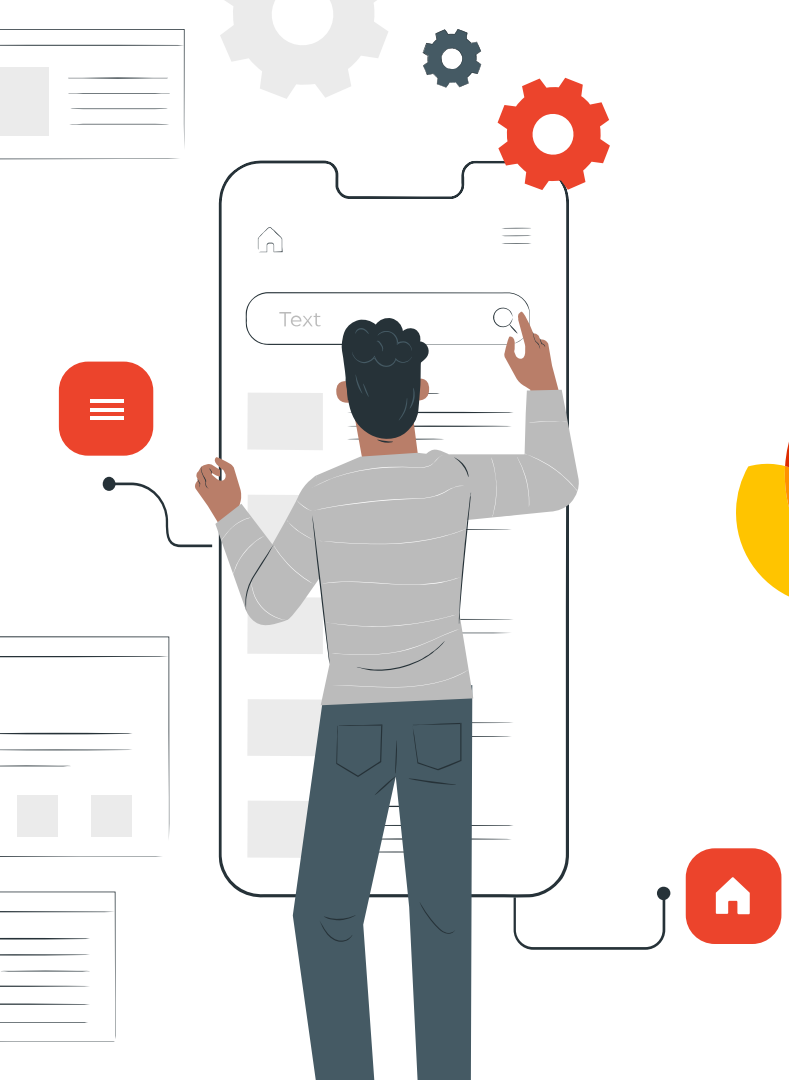


Intruduction to Firebase

Samira Ahamdnezhad

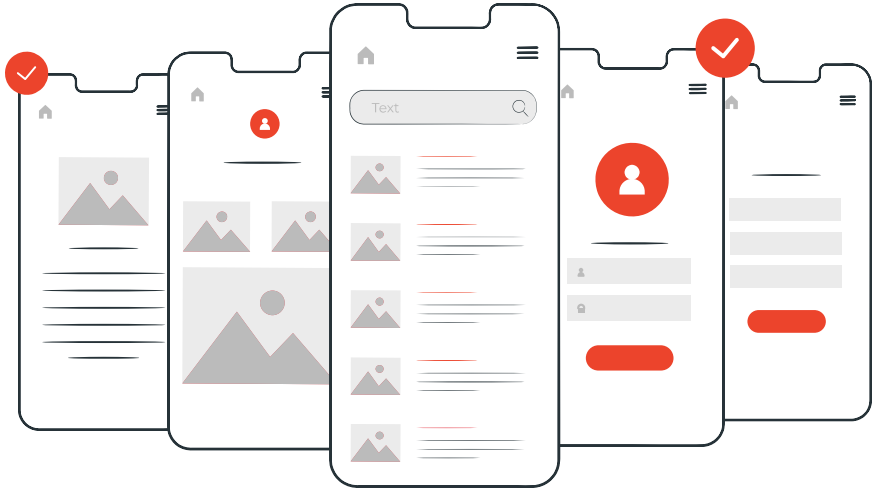
AP Project Fall 403
Dr.Vahidi Asl

What is Firebase?

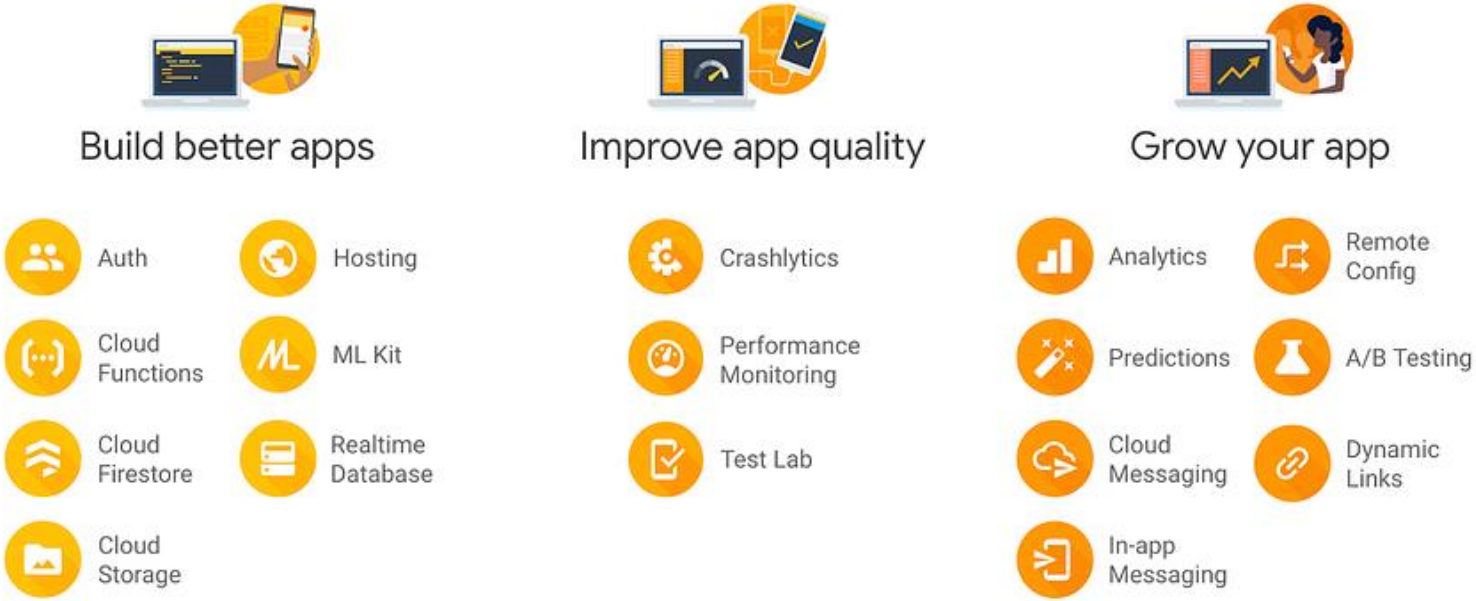
Firebase is Google's mobile application development platform that helps you build, improve, and grow your app.
The tools it gives you cover a large portion of the services that developers would normally have to build themselves, but don't really want to, because they'd rather focus on the app experience itself. This includes things like analytics, authentication, databases, configuration, file storage, push messaging, and the list goes on. The services are hosted in the cloud, and can be scaled with little to no effort from the developer.

What kind of apps can benefit from Firebase?

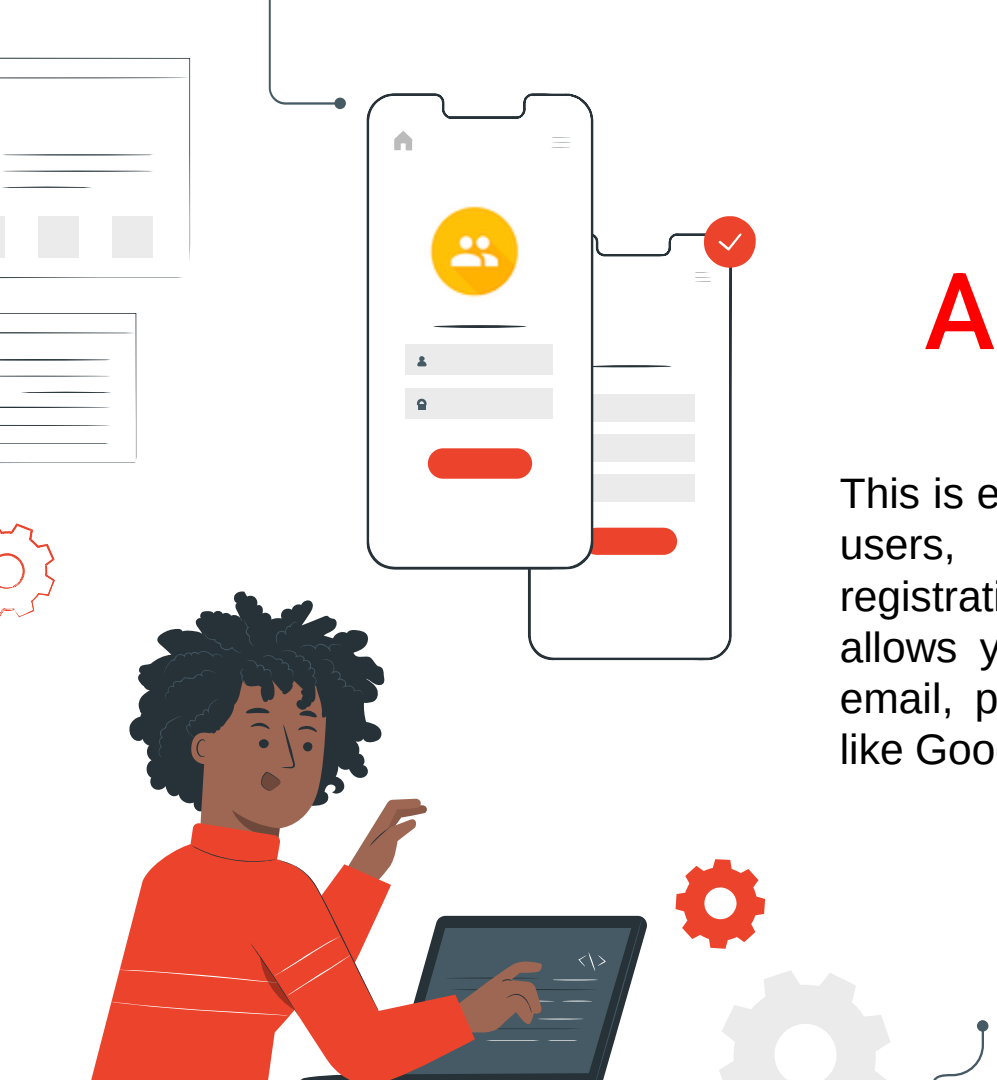
There's really no limit to the types of apps that Firebase products can help. Only the platforms where Firebase can be used, are limited. iOS and Android are the primary targets for the Firebase SDKs, and there's growing support for web, Flutter, Unity, and C++. You should also know that there's an Admin SDK available for a variety of languages, which can be used with any backend components you might require.



products in the Firebase suite



In the following sections, we will introduce 5 key features that form the foundation of many successful applications, and using them can significantly enhance performance, as well as improving user experience.



Authentication

This is essential for any app that interacts with users, as managing secure sign-in and registration is critical. Firebase Authentication allows you to easily implement sign-ins with email, phone number, or third-party services like Google and Facebook.

02

Cloud Firestore Realtime Database

These services enable real-time data storage and synchronization, which is vital for apps with dynamic data. Realtime Database is especially useful for projects requiring live data synchronization between multiple users, while Cloud Firestore offers more advanced features for managing large-scale data.

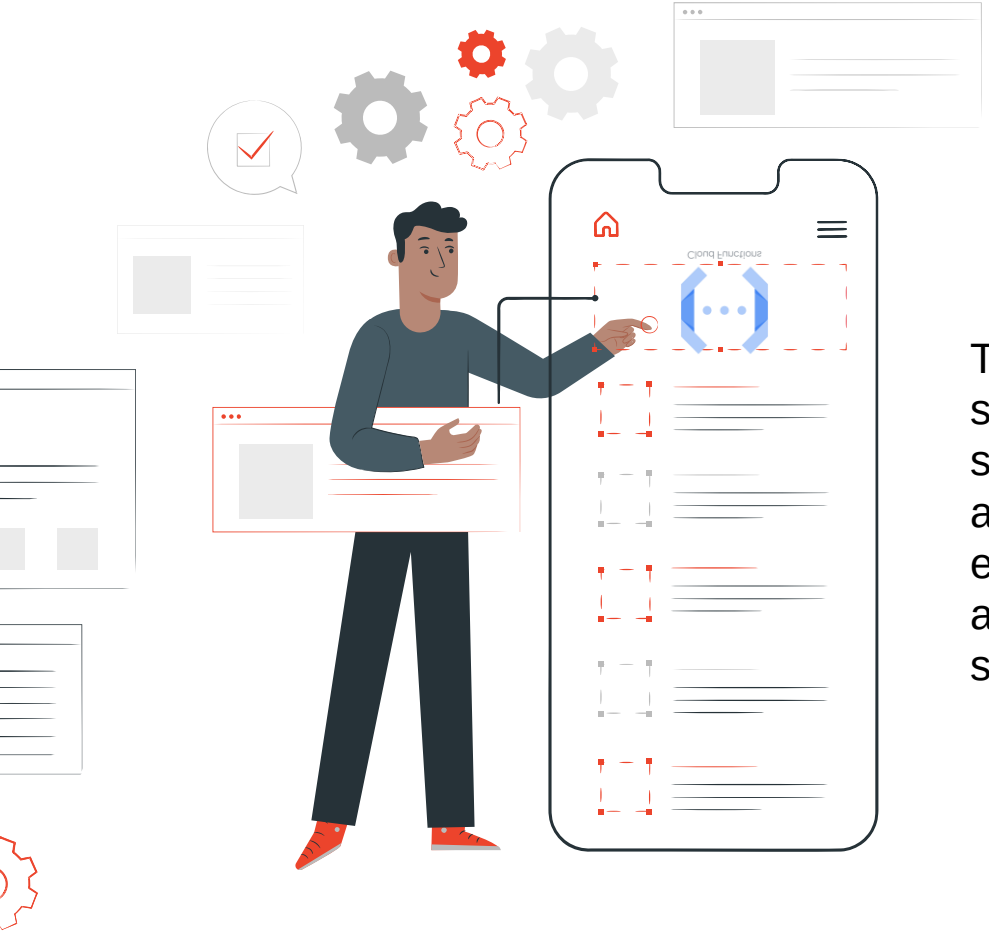
Firebase Analytics

Understanding user behavior and analyzing usage patterns is crucial for the success of any app. Firebase Analytics is a powerful tool for tracking user behavior, measuring engagement, and analyzing the results of marketing campaigns. It helps you optimize the user experience.

04

Firebase Cloud Messaging

If your app needs to send real-time notifications or messages to users, Firebase Cloud Messaging (FCM) is one of the most important services. It allows you to send targeted push notifications and in-app messages, which are key for user engagement.



Cloud Functions

This feature lets you implement server-side logic without managing actual servers. Cloud Functions execute automated backend code in response to events like database changes or user actions, reducing the need for separate server infrastructure.

Pros of Firebase

High scalability

Easily scalable for both small and large applications.

Cross-platform capabilities

Supports Android, iOS, web, and other platforms.

Rapid development

Integrated tools that help developers build apps faster.

Comprehensive analytics

Provides in-depth analytics tools to enhance user experience and increase revenue.

Cons of Firebase

Geographical limitations

Restrictions on using Firebase in some countries (e.g., Iran).

High costs

The costs of using Firebase increase as your app scales.
Pay-as-you-go

Dependence on Google

Your app could be affected if Google changes its policies.

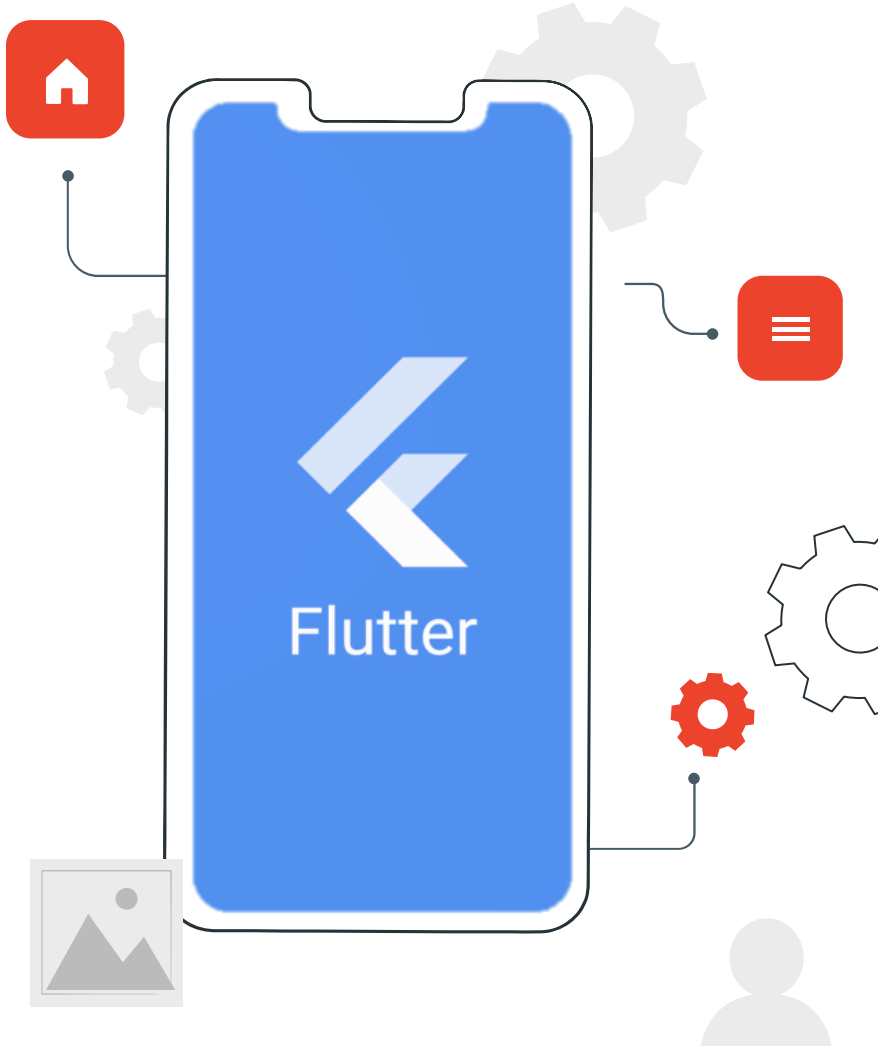
Limited database control

Less control over data structure and queries compared to traditional databases.

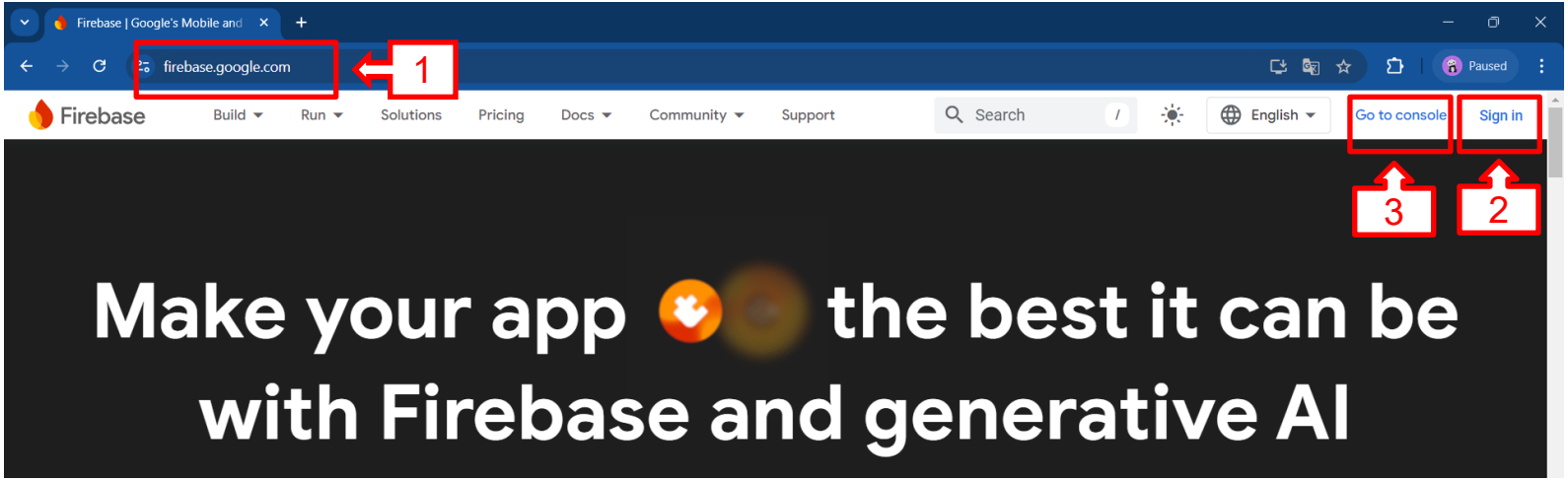
Using Firebase in Flutter Projects

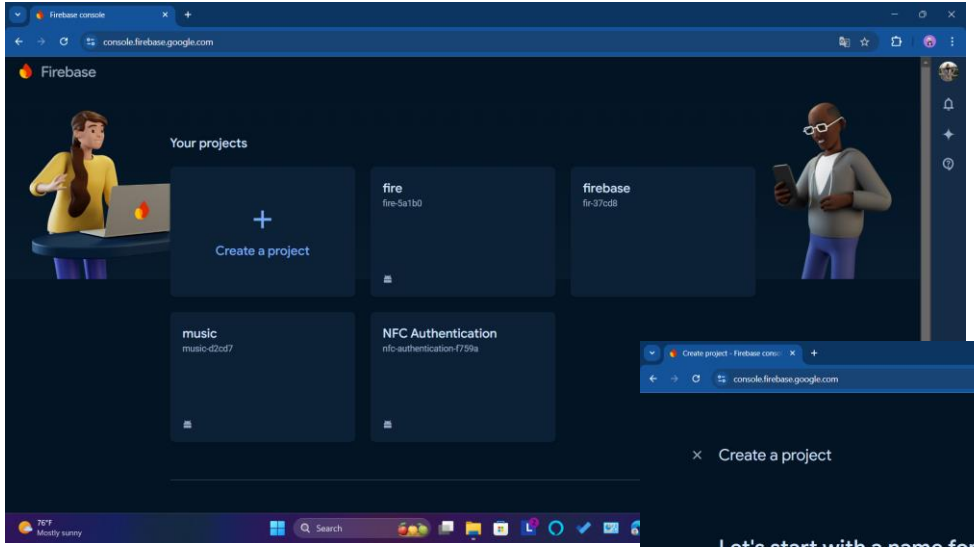
In this document, we will only cover the set-up process of Firebase in a **Flutter** project. After that, you can explore and integrate other Firebase services based on your app's needs. To learn more about each service, check the official Firebase documentation, and select the features that best suit your application.

First, set-up your DNS since Firebase is somewhat restricted. 📲
Let's move on and continue the process together 💻

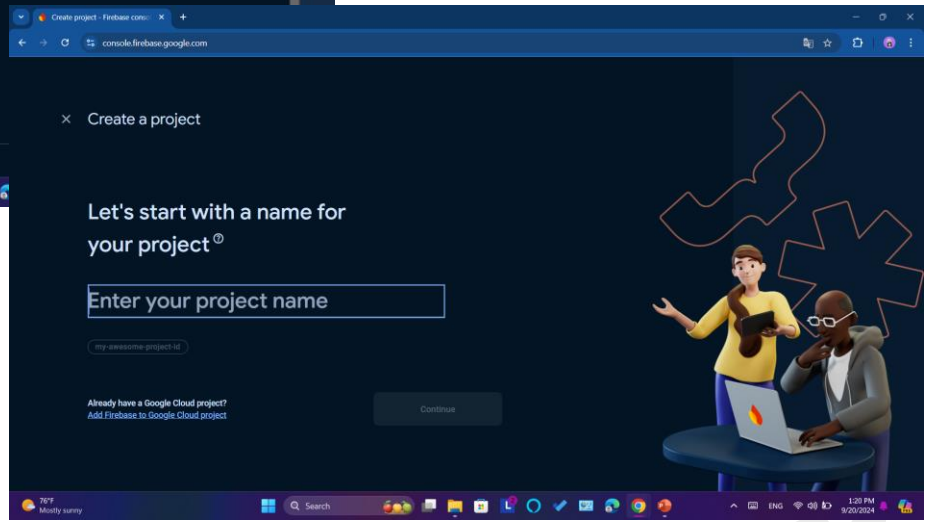


1. Go to the Firebase website and sign in with a new or existing Google Account. And then enter the Firebase console.

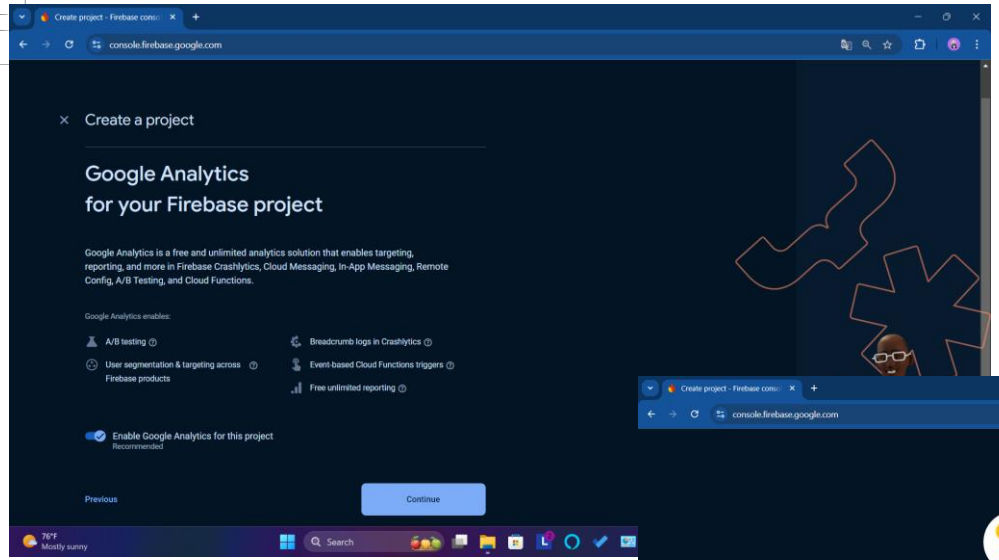




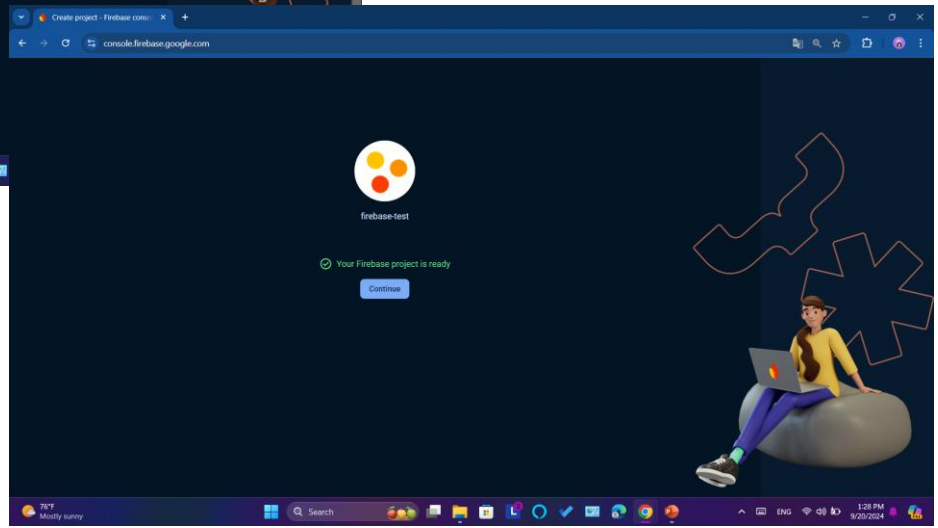
2. Click on Create a Project



3. Enter the name of your project



4. You can disable or enable Google Analytics for your project , according to your needs. We disable this option here and continue.

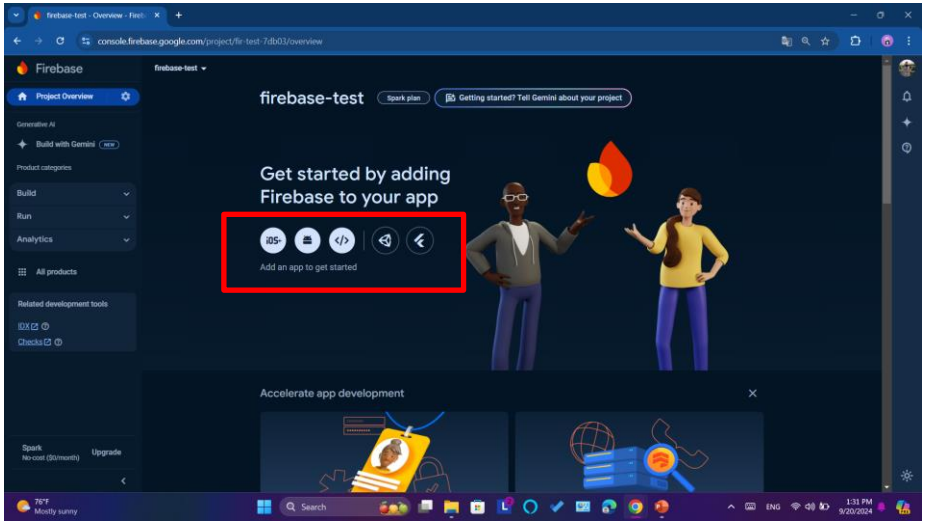


5. Now our project is ready :) All that remains is to connect it to our Flutter project.

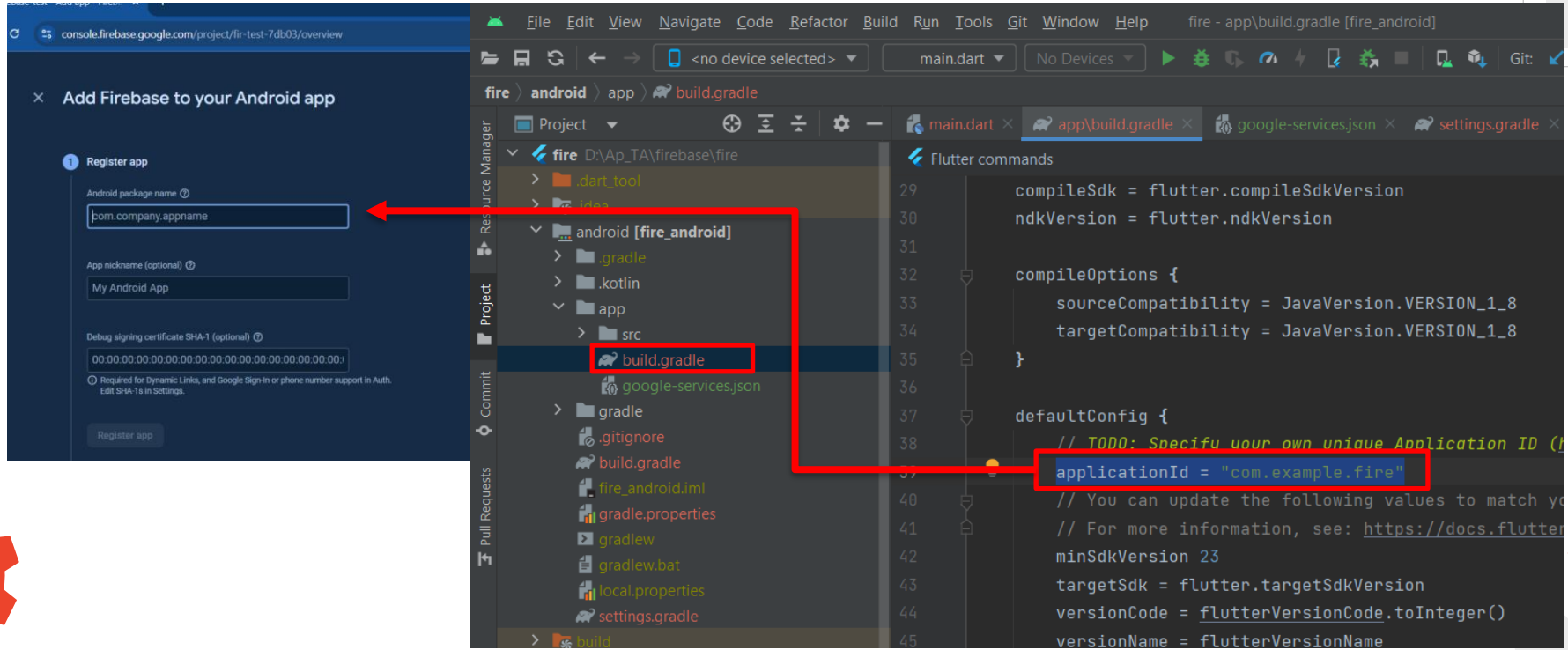
6. After creating the project, click on Add App and select the Android platform

In Flutter projects, since Flutter is cross-platform (Android and iOS), you need to configure Firebase separately for each platform. If you only want to set up Firebase for Android, you can follow the same steps as for native Android projects. However, if your app runs on both Android and iOS, you will need to complete the setup for both platforms. Here, we will go through the steps for Android. If you want to set it up for iOS, you can follow similar steps based on Firebase's documentation.

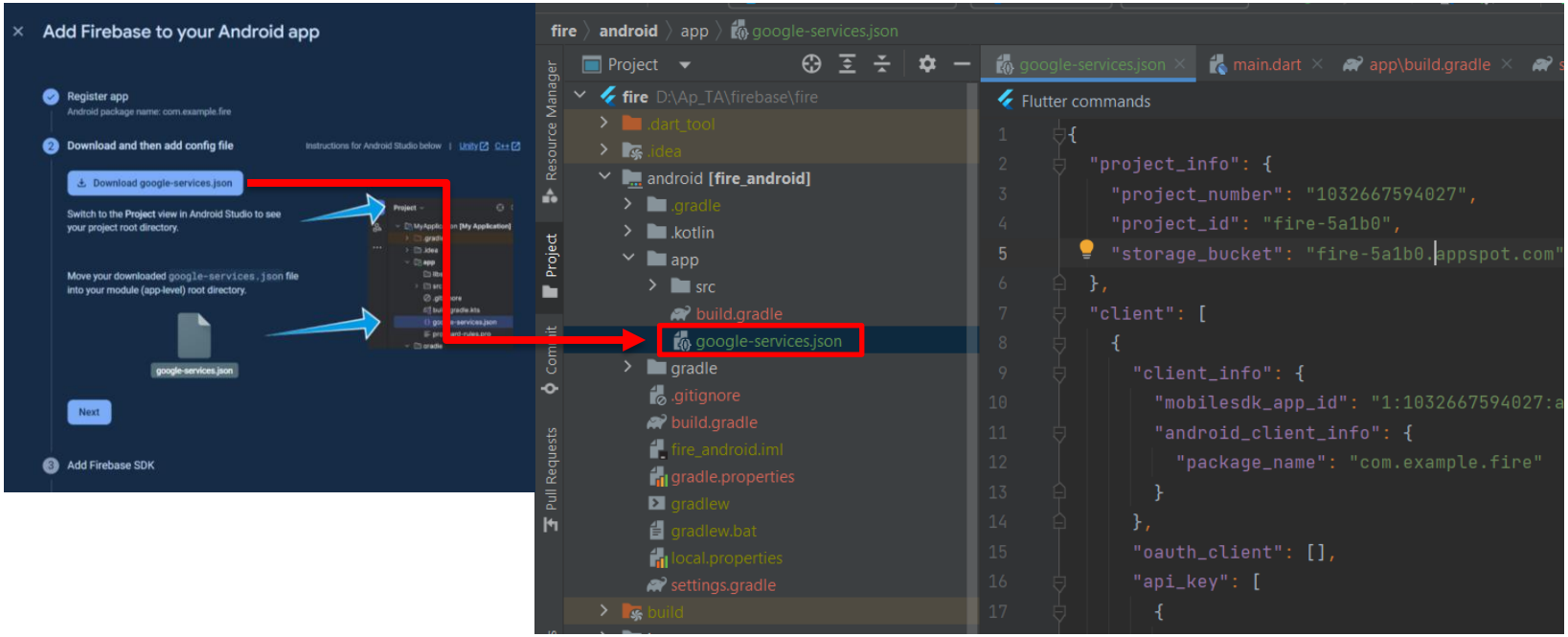
Firebase also fully supports configuration for complete Flutter projects, but we won't be covering that for now.



7. Enter your package name (found in android/app/build.gradle in your Flutter project)
App nickname and SHA-1 are optional; click "Register App" afterward



8. Once the Android app is registered, download the google-services.json file. Add this file to the android/app/ directory of your Flutter project.



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
}
```

9. Go to the android/settings.gradle file and add the following line under plugins.Change the version to '4.3.13'.

10. Go to the Kotlin website and copy the latest version.
Go to the android/settings.gradle file and change the kotlin version under plugins.
Change its version to the latest version

kotlinlang.org/docs/releases.html#release-details

Create with Kotlin Multiplatfo

v2.0.20

Build info	Build highligh
2.0.20 Released: **August 22, 2024** Release on GitHub ↗	bug fixes. Fea collector, UUI updates, and Learn more al

settings.gradle | google-services.json | main.dart | app\build.gradle | android\build.gradle

Flutter commands

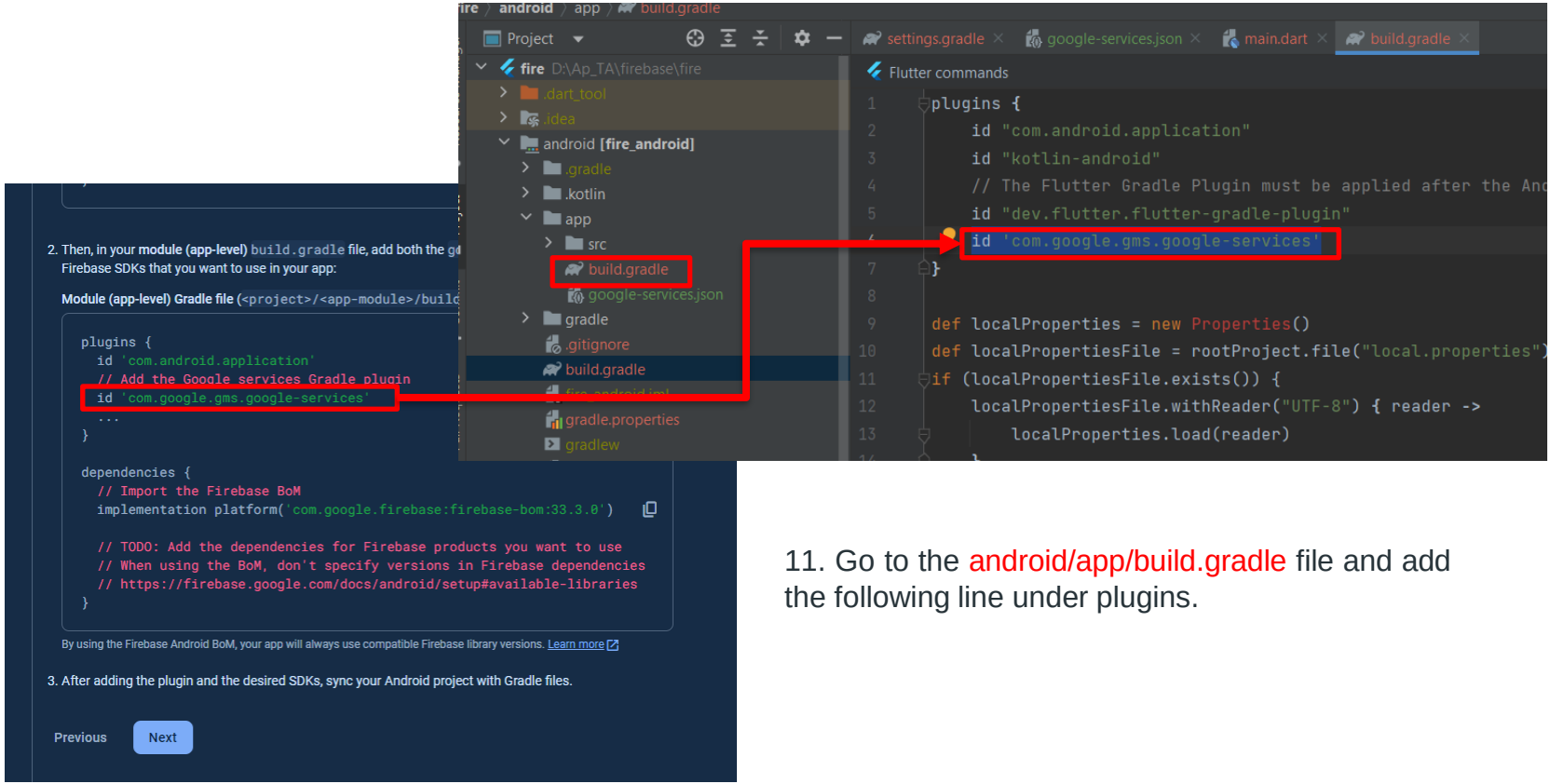
```
        def properties = new Properties()
        file("local.properties").withInputStream { properties.load(it) }
        def flutterSdkPath = properties.getProperty("flutter.sdk")
        assert flutterSdkPath != null, "flutter.sdk not set in local.properties"
        return flutterSdkPath
    }()

    includeBuild("$flutterSdkPath/packages/flutter_tools/gradle")

    repositories {
        google()
        mavenCentral()
        gradlePluginPortal()
    }
}

plugins {
    id "dev.flutter.flutter-plugin-loader" version "1.0.0"
    id "com.android.application" version "7.3.0" apply false
    id "org.jetbrains.kotlin.android" version "2.0.20" apply false
    id 'com.google.gms.google-services' version '4.3.13' apply false
}
```

- .gradle
- .kotlin
- app
  - src
  - build.gradle
  - google-services.json
- gradle
- .gitignore
- build.gradle
- fire_android.iml
- gradle.properties
- gradlew
- gradlew.bat
- local.properties
- settings.gradle
- test
- web



11. Go to the android/app/build.gradle file and add the following line under plugins.

Now our Flutter project is ready to be connected to Firebase ☺

In order to make sure that we have done the settings correctly, we need to add the firebase_core package to the project and write the following code in the main file.

main.dart × settings.gradle × google-services.json × pubspec.yaml × bui

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
```

Build Variants

Structure

- .gitignore
- .metadata
- analysis_options.yaml
- fire.iml
- pubspec.lock
- pubspec.yaml
- README.md
- External Libraries
- Scratches and Consoles

```yaml
  # The following adds the Cupertino Icons
  # Use with the CupertinoIcons class for i
  cupertino_icons: ^1.0.6
  firebase_core: ^3.5.0
  cloud_firestore: ^5.4.2
  firebase_auth: ^5.3.0

dev_dependencies:
  flutter_test:
    sdk: flutter

  # The "flutter_lints" package below conta
  # encourage good coding practices. The li
  # activated in the `analysis_options.yaml
  # package. See that file for information
  # rules and activating additional ones.
  flutter_lints: ^3.0.0

# For information on the generic Dart part
```

Document 1/1 › dependencies: › firebase_core: › ^3.5.0

If the application runs without any errors, it means that our Flutter project is correctly connected to Firebase

Useful Links

https://firebase.google.com/docs

https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0

https://youtu.be/Hf8qwqjRIdo?si=jL4NirUxFmu54gFT