Android config

Gradle.properties:

kotlin.jvm.target.validation.mode = IGNORE

set to fix error that showed up after installing google places sdk <https://stackoverflow.com/questions/77520506/inconsistent-jvm-target-compatibility-detected-for-tasks-despite-already-definin>

# Web

flutter run -d chrome --web-hostname localhost --web-port 60875 --web-browser-flag "--disable-web-security"

# Todo

* Web, iOS Google API Key config
* Set Android API Key restriction (had it set but it broke google places sdk autocomplete)

# Safety Concerns to address

<https://docs.github.com/en/rest/authentication/keeping-your-api-credentials-secure?apiVersion=2022-11-28>

Remove all tokens / API keys from the repo!!

<https://ai.google.dev/gemini-api/docs/get-started/tutorial?lang=dart>

Caution: Using the Google AI SDK for Dart (Flutter) to call the Google AI Gemini API directly from your app is recommended for prototyping only. For non-prototyping use cases, we strongly recommend that you use the SDK to call the Google AI Gemini API only server-side to keep your API key safe. If you embed your API key directly in your mobile or web app or fetch it remotely at runtime, you risk potentially exposing your API key to malicious actors.

**Resolution: Switched from Gemini API to server side Gemini via FirebaseVertexAi**

**Google API Key is hidden by defining it in Firebase Remote Config**

# Gemini

**flutter run --dart-define=API\_KEY=AIzaSyDqpIXIFYrFpZDTA8NtfymlUD8qm9s2YSY**

A screenshot of a social media post

Description automatically generated

# Auth

**1 Firebase Auth**

<https://pub.dev/packages/firebase_auth>

<https://firebase.google.com/products/auth/>

Easy sign-in with any platform Firebase Authentication aims to make building secure authentication systems easy, while improving the sign-in and onboarding experience for end users. It provides an end-to-end identity solution, supporting email and password accounts, phone auth, and Google, Twitter, Facebook, and GitHub login, and more.

**2 Google Sign In**

<https://pub.dev/packages/google_sign_in>

Often, developers use google\_sign\_in in combination with firebase\_auth to handle Google authentication within the Firebase authentication framework. Here's a brief outline of how they can be used together:

1. Google Sign-In: Use google\_sign\_in to authenticate the user with their Google account.
2. Firebase Authentication: Use firebase\_auth to link the Google account to a Firebase user, thereby allowing you to utilize Firebase's authentication management and integration capabilities.

Example Workflow

1. User Signs In with Google:
   1. Use google\_sign\_in to prompt the user to sign in with their Google account.
   2. Retrieve the Google authentication credentials.
2. Link to Firebase:
   1. Pass the Google credentials to firebase\_auth to authenticate the user within the Firebase system.
   2. Firebase handles the session management, providing secure access to Firebase services.

# Storage

## Local

* sharedPrefs
* SQL

## Remote

* Firestore (text data, URLs to media data)
* Firebase Storage (media data)

# Packages Notes