Setup Guide

Lead Zachary Trabookis
Developer jhgilre@g.clemson.edu
Developer ncarte4@g.clemson.edu

Introduction

This Quick Start guide will help you handle the necessary API keys and models to get your ReadRight project running. ReadRight relies on several external Speech-To-Text (STT) and mobile backend services to run correctly. Please read over the external requirements list below and make sure that your forked version of ReadRight includes these items.

External Requirements

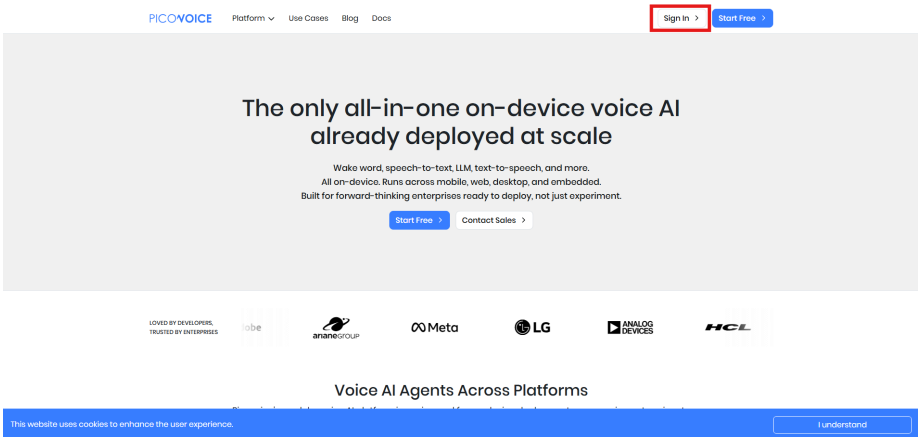
Indicates the following external applications necessary to make the ReadRight application function properly.

Priority	Detailed description
Speech-To-Text (STT)	**Cheetah:** On-Device STT software that's used to create a transcript file from student's audio recording. **Deepgram:** Cloud-based STT software that's used to create a transcript file from student's audio recording.
Authentication	**Firebase Authentication:** Cloud service that includes multiple sign-in providers. The ReadRight application uses the Email/Password provider.

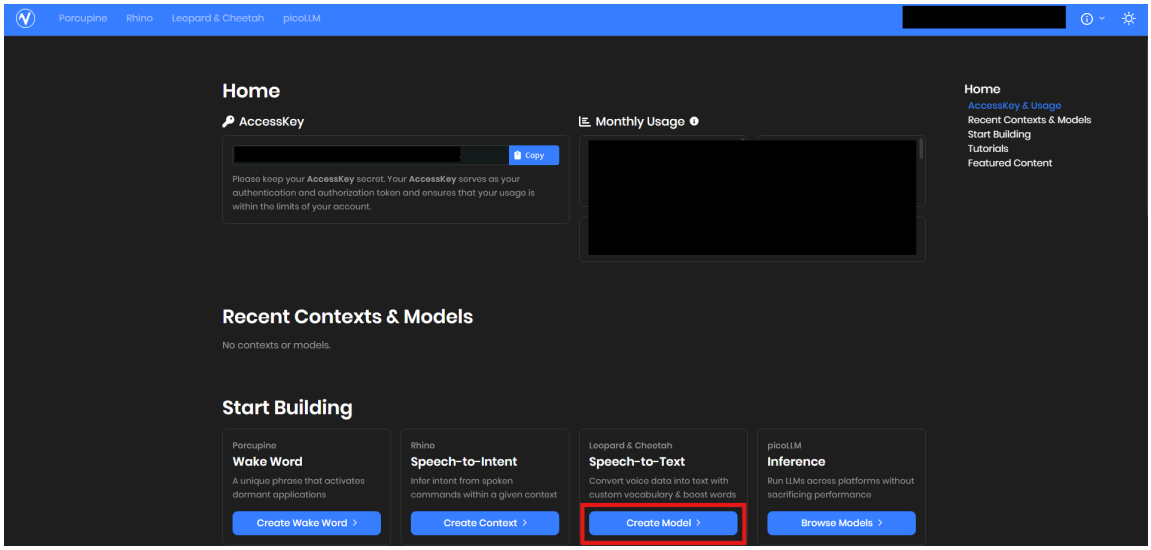
Priority	Detailed description
Database	**Firebase Firestore:** A document store solution for recording documents of metadata for collections (attempts, classes, settings, student.progress, users, words). For details about what each collection includes, please refer to the Data Model Diagram.
Storage	**Firebase Storage:** Holds audio files for each student and their corresponding word attempt codec files (e.g. ACC).

Cheetah

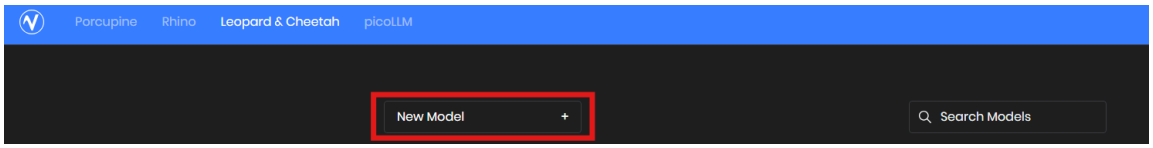
1. Begin by navigating to picovoice.ai/ and clicking the 'Sign In' button in the upper right of the screen:



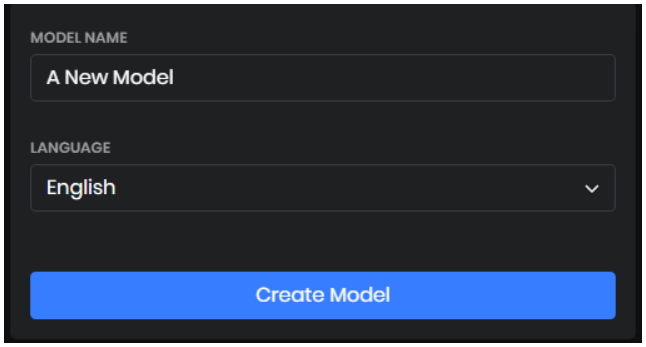
2. If you have not created an account with Picovoice, sign up following the on screen instructions and log in.
3. Upon signing in, you will be greeted with Picovoice's console. First, you must create a Cheetah Speech-to-Text (STT) model by navigating to the 'Create Model' button towards the bottom of the screen:



4. Click the ‘New Model + ’ button to begin creating your model:

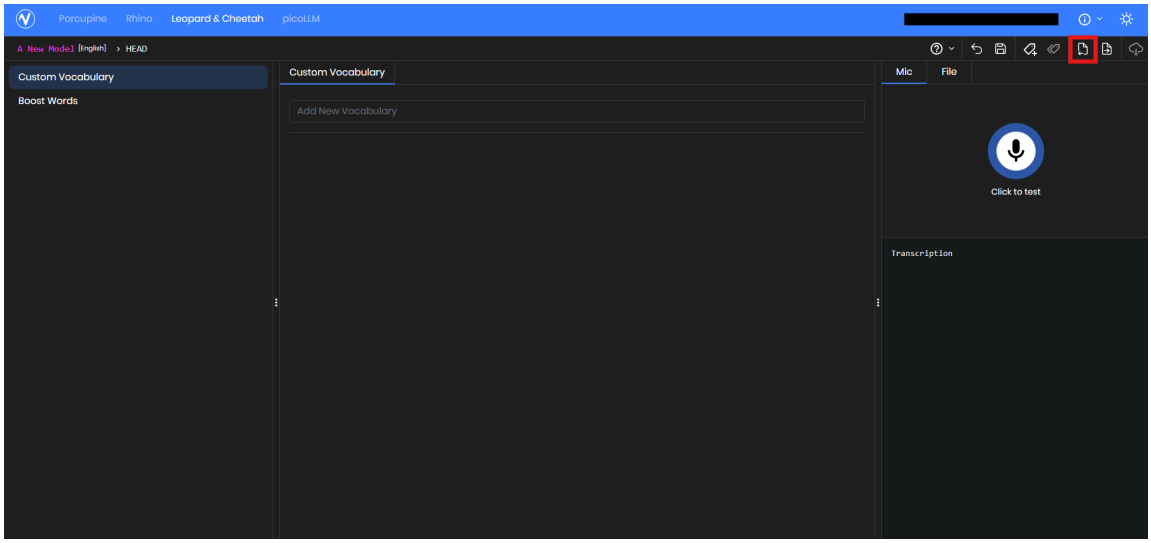


5. Name your model, select English, and click ‘Create Model’:



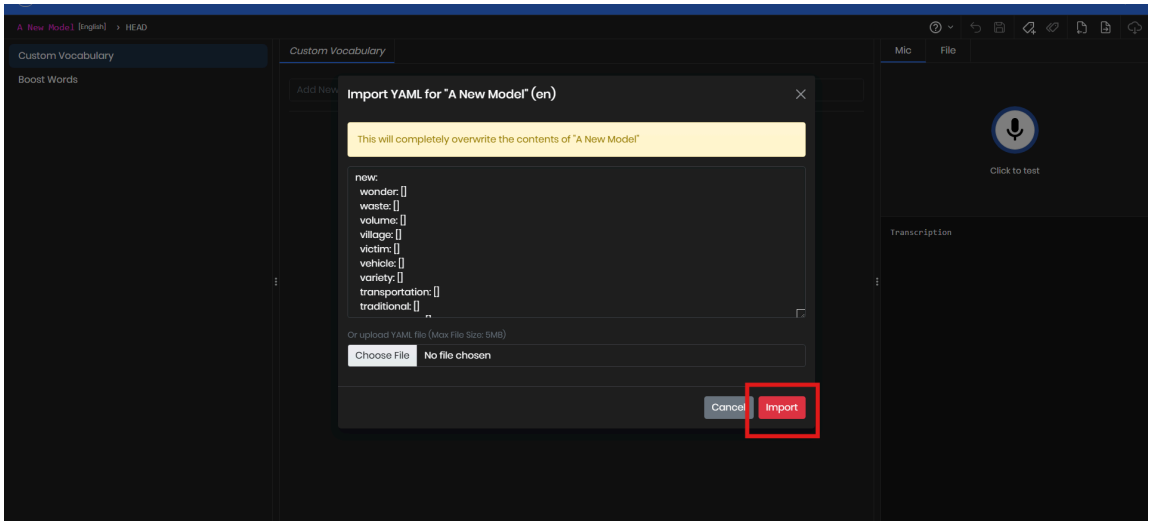
6. Upon creating your model, you will be greeted with an interface to add words to your model. Find the ‘Import YAML’ icon button in the upper right of the screen

and click on it:

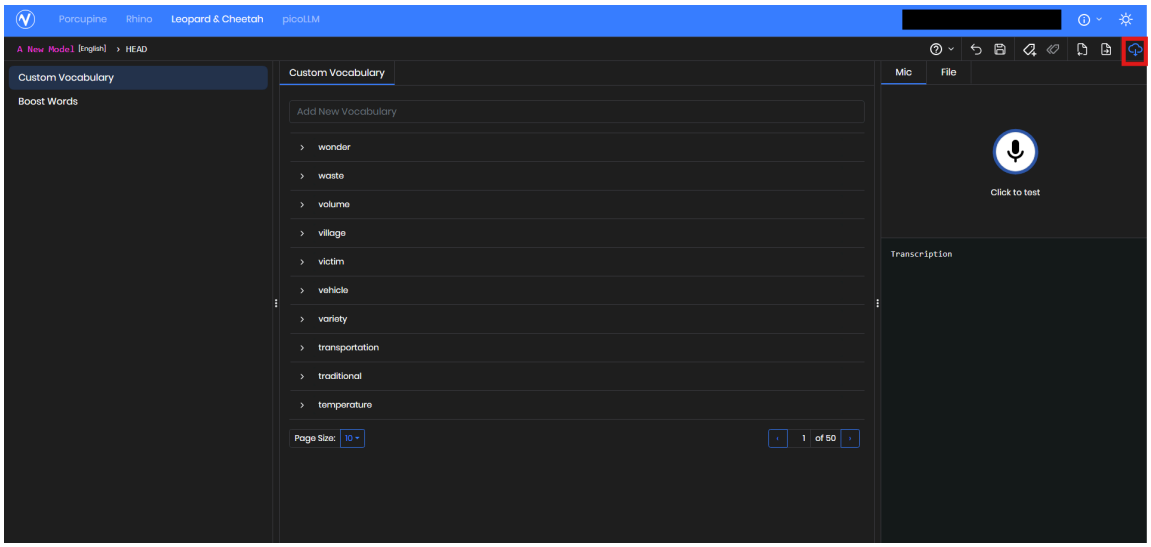


7. With the ‘Import YAML for “Your Model’s Name” (en)’ dialogue open, copy and paste or import the following file into the dialogue: \readright\data\Cheetah_ReadRight.yml

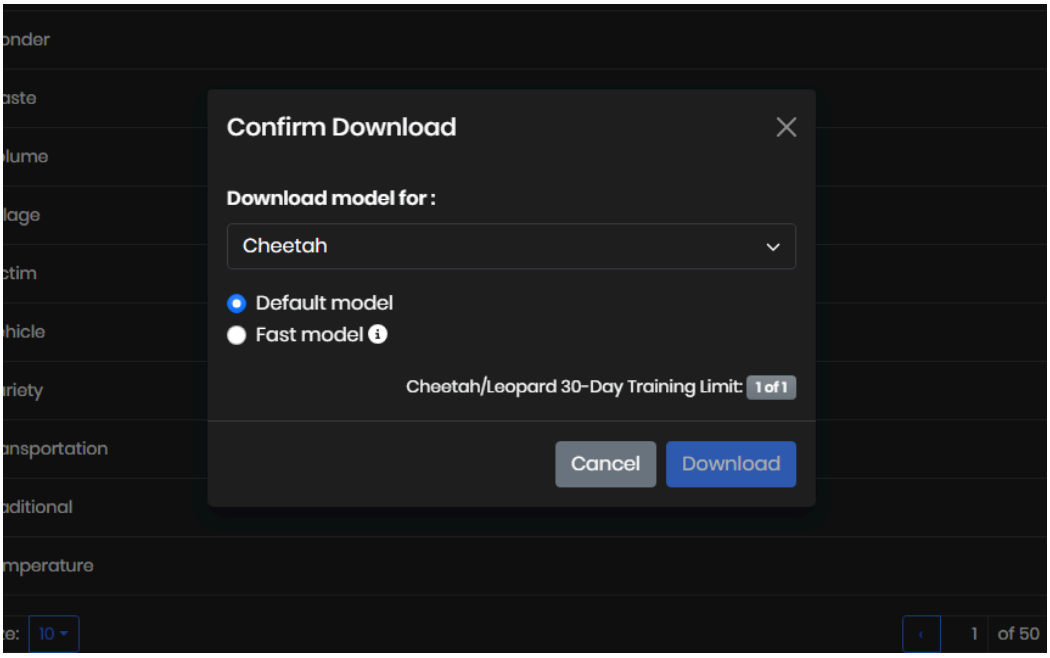
   Then click the ‘Import’ button:



8. Once the words are imported, click the ‘Download Model’ icon button in the upper right of the screen:



9. Select Cheetah from the dropdown menu and the ‘Download’ button to download your model:

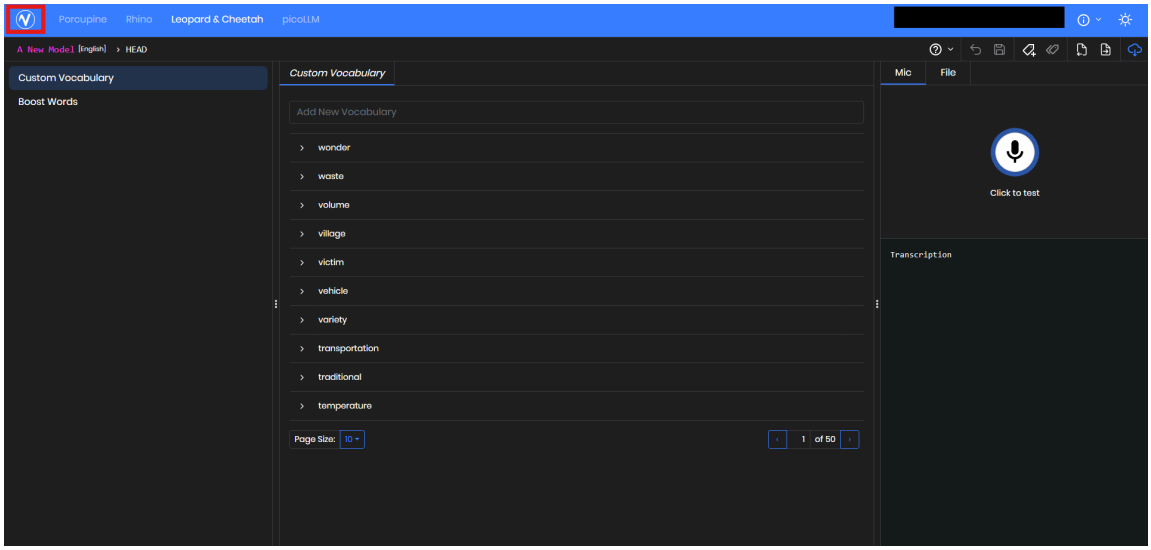


10. Once the model is downloaded, take the .pv file you just downloaded and replace the existing .pv file at the end of this path: readright\data\<File to be replaced>.pv
11. After your new model has replaced the existing model, navigate to \readright\lib\audio\stt\on_device\cheetah_assessor.dart and replace the

modelPath String variable around line 23:

```
class CheetahAssessor implements PronunciationAssessor {
  String accessKey = 'ptQzT1bn5rGVtfHhjX/fUggamdgh+Q0C1apku5C90SCpt81J5aRGdw==';  // Acces
  String modelPath = 'data/Cheetah_ReadRight.pv'; // path relative to the assets folder or

  Cheetah? _cheetah;

  final PcmRecorder pcmRecorder;
  final String practiceWord;
```

12. Now that your model is instantiated, navigate back to the Picovoice console by clicking the logo in the upper left of the screen:

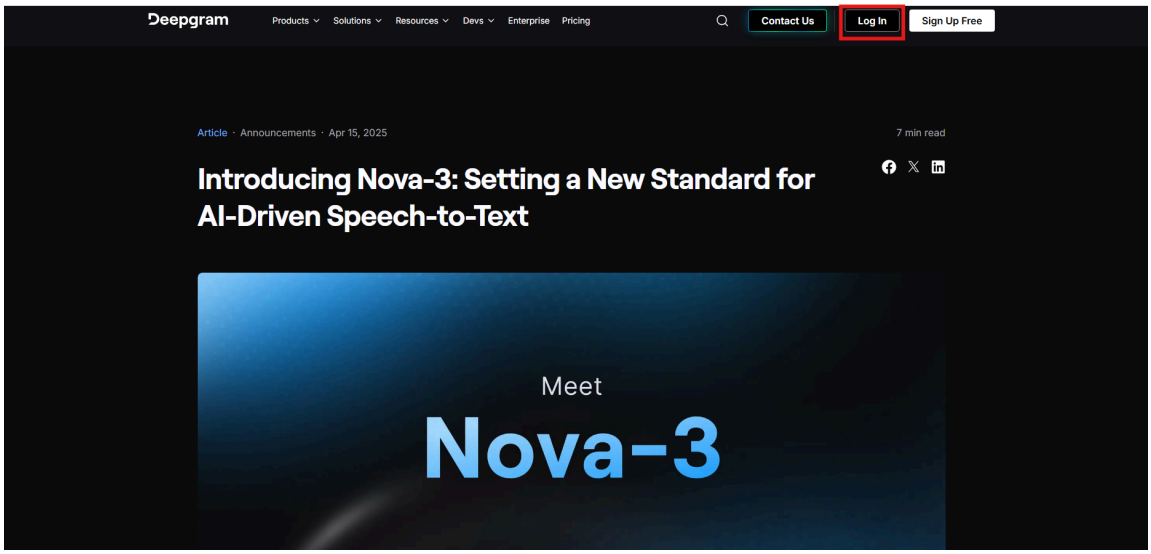


13. The ‘AccessKey’ shown near the center of the screen is your API key for Cheetah. Copy it and navigate to \readright\lib\audio\stt\on_device\cheetah_assessor.dart and replace the accessKey String variable around line 22 with your access key:

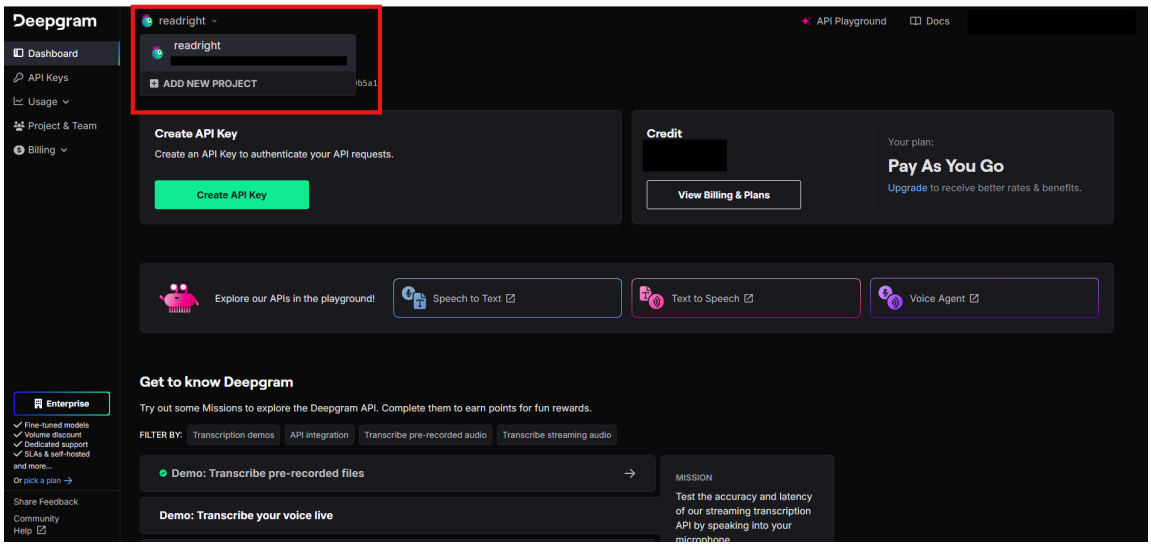
```
class CheetahAssessor implements PronunciationAssessor {
  String accessKey = 'ptQzT1bn5rGVtfHhjX/fUggamdgh+Q0C1apku5C90SCpt81J5aRGdw==';
  String modelPath = 'data/Cheetah_ReadRight.pv'; // path relative to the assets f

  Cheetah? _cheetah;

  final PcmRecorder pcmRecorder;
  final String practiceWord;
```

Deepgram

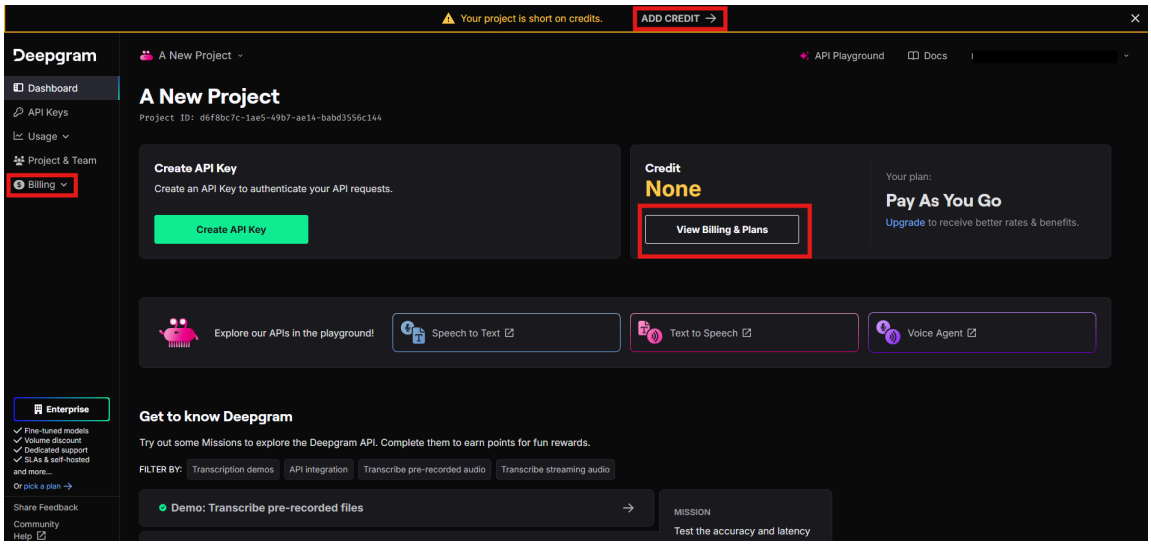
1. Begin by navigating to deepgram.com/learn/introducing-nova-3-speech-to-text-api and clicking on the 'Log In' button at the top right of the page:



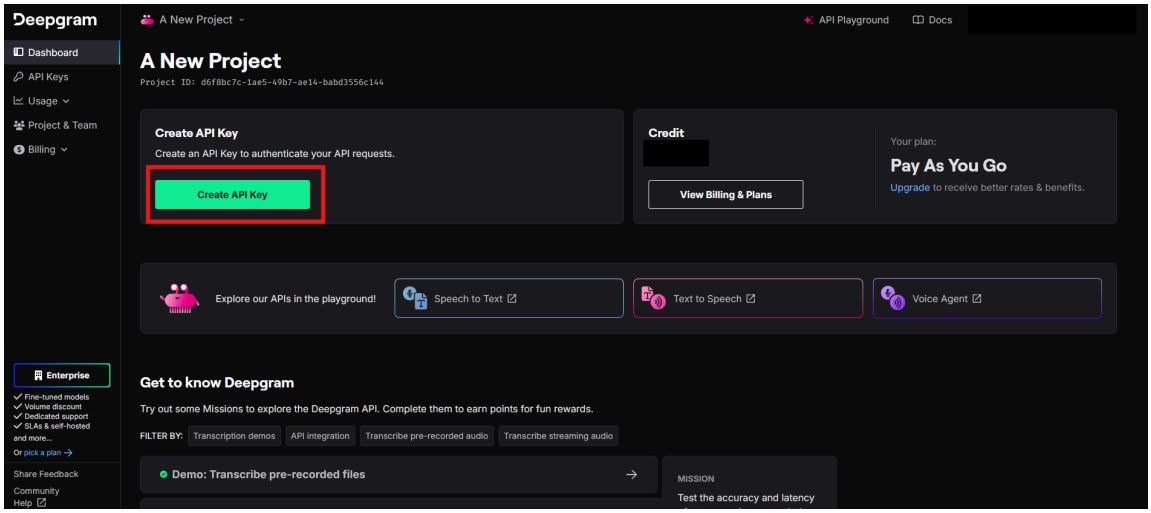
2. If you have not created an account with Deepgram, sign up using the on screen instructions and log in.
3. Upon signing in, you will be greeted by Deepgram's Dashboard. In order to generate a working API key, you MUST create a project. Navigate to the top left of the dashboard and click on the project drop down, then add new project:



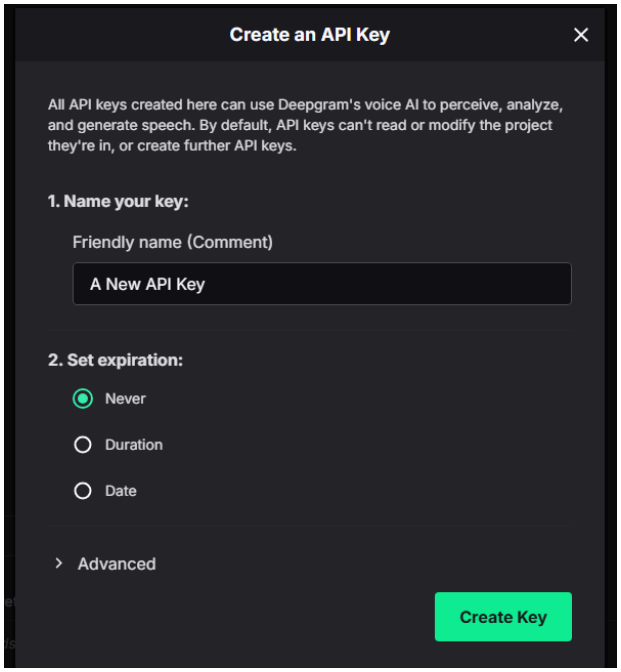
4. Once your project is created, add a credit to your project via the ‘ADD CREDIT’ bar along the top or by navigating to the ‘Billing’ tab:

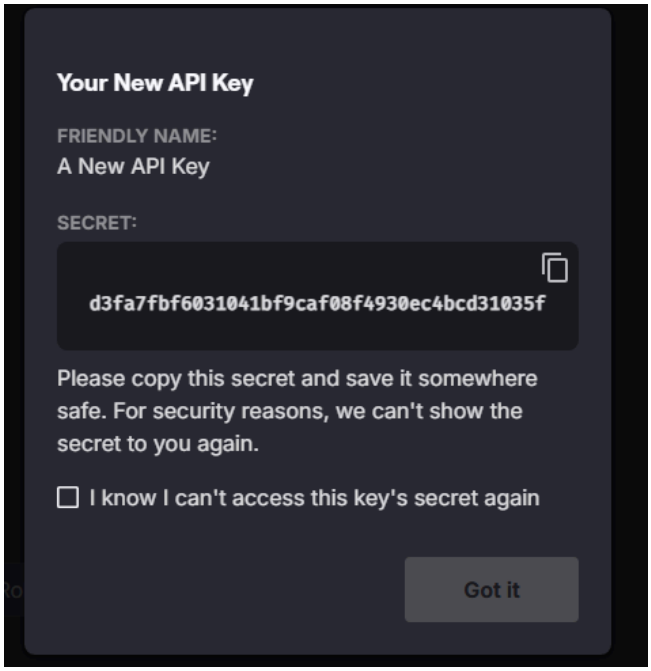


5. Select the billing option that best suits your use case. Follow the on-screen instructions until your project has credits.
6. Once your account and project are created and your project has credits, return to the ‘Dashboard’ tab and press the ‘Create a New API Key’ button.



7. Follow the on screen instructions to name your key, set its expiration date, add tags, and designate the role. Once your settings are in place, press the ‘Create Key’ button. Ensure to copy your key here before acknowledging that you cannot access this full key again.

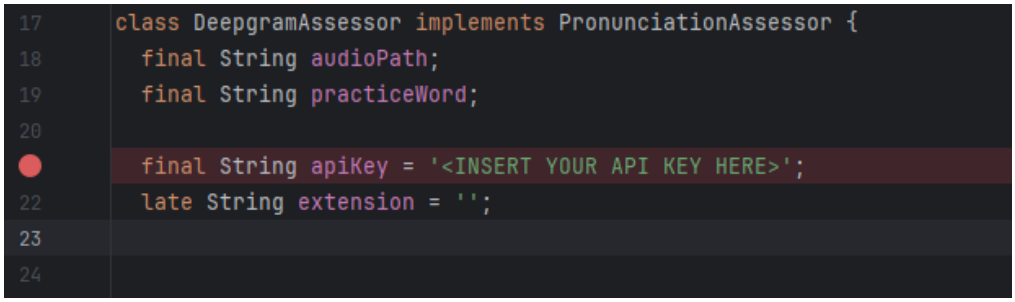




8. Now that you have your API key, navigate to the following path from the project root:

   readright\lib\audio\stt\cloud\deepgram_assessor.dart

9. Around line 21, you will find an apiKey String variable. Simply paste your key here:



Firebase

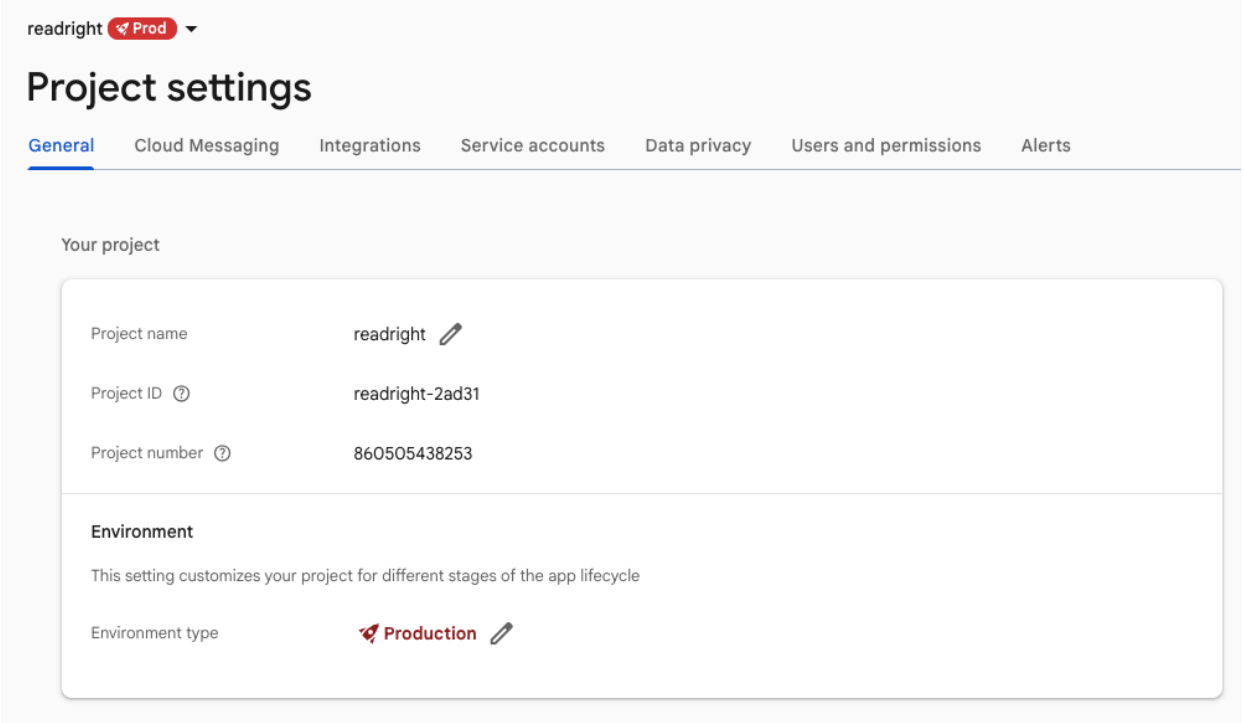
As part of setting up ReadRight's backend, Firebase provides three core services that the ReadRight application relies on: **Authentication**, **Cloud Firestore**, and **Cloud Storage**. Firebase Authentication adds secure sign-in options—such as email/password or social providers like Google and Apple—without building your own login system or handling sensitive credentials. Cloud Firestore serves as the primary database, giving the app scalable, real-time NoSQL datastore for user profiles, student progress tracking, classroom data, etc. Please refer to the Data Model Diagram documentation to see a breakdown of collection document metadata. Firebase Storage allows the application to store student's audio recordings while being protected by the same authentication and security rules. Together, these services give the ReadRight app a robust, easy-to-maintain backend foundation, allowing developers to focus on new or existing features and UI rather than infrastructure.

Follow this guide below to learn how to configure the ReadRight app for Firebase integration. The steps cover setting up Firebase Command Line Tools (CLI), Firebase console configuration, and Firebase Firestore security rules.

Add Firebase to the ReadRight Flutter Application

Create a Firebase console account by logging in with an existing Google account. We chose to use the *Blaze (Pay as you go)* plan to ensure that Firestore, Cloud Storage and additional Firebase services work without restrictions. Make sure to rename the project to *readright* and set the environment type to *Production*.

https://console.firebase.google.com



Following these directions to add Firebase to a Flutter iOS app.
https://firebase.google.com/docs/flutter/setup

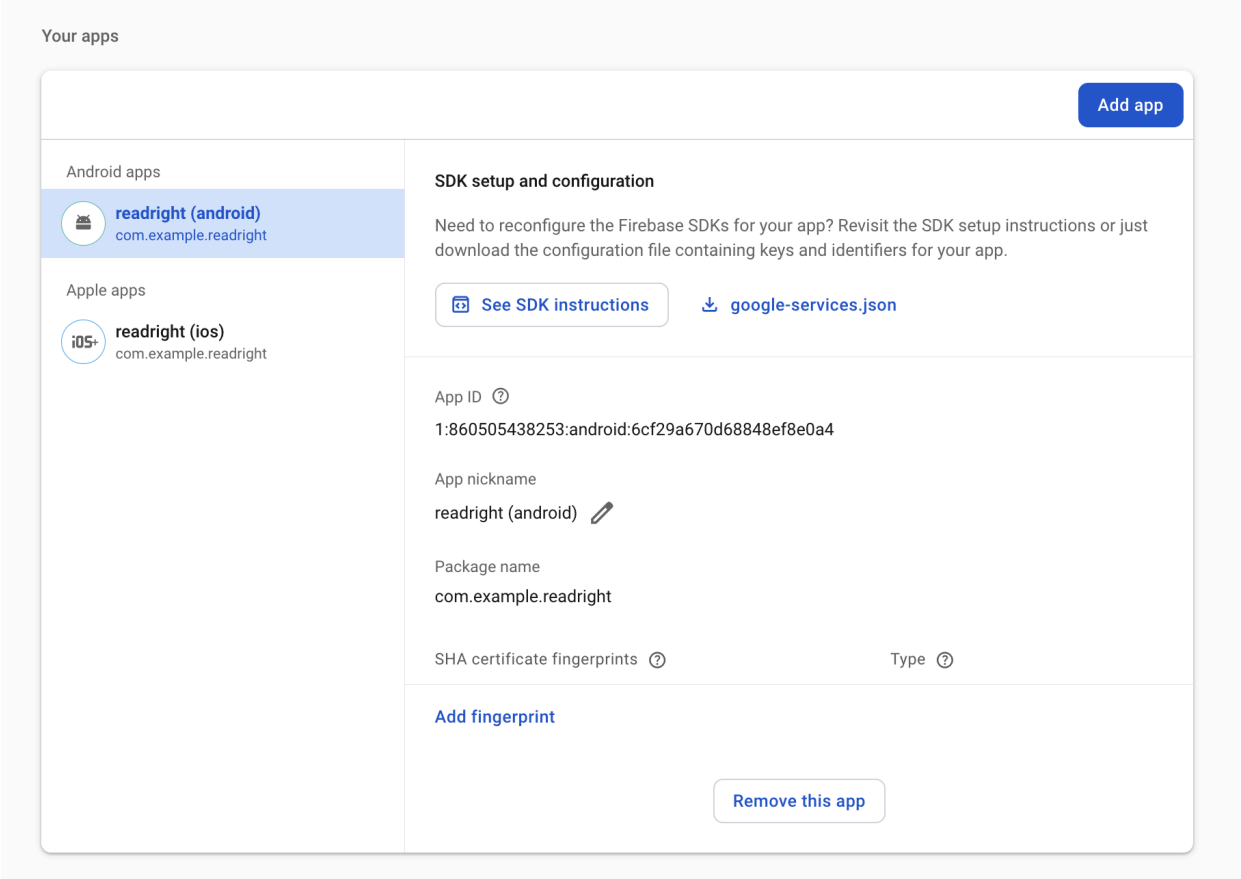
Configure ReadRight to use Firebase

Here is example output that we ran in the Firebase CLI to configure Firebase with Android and iOS applications.

```
% flutterfire configure
i Found 3 Firebase projects.
? Select a Firebase project to configure your Flutter application with ›
  cwt-starter-template-master (cwt-starter-template-master)
❯ readright-2ad31 (readright)
  travelflutter-ca5bf (TravelFlutter)
  <create a new project>

# Selecting android and ios currently. We can add additional support for other OS later.
? Which platforms should your configuration support (use arrow keys & space to select)? ›
✔ android
✔ ios
  macos
```

```
  web
  windows

i Firebase android app com.example.readright is not registered on Firebase project
readright-2ad31.
i Registered a new Firebase android app on Firebase project readright-2ad31.
i Firebase ios app com.example.readright is not registered on Firebase project
readright-2ad31.
i Registered a new Firebase ios app on Firebase project readright-2ad31.

Firebase configuration file lib/firebase_options.dart generated successfully with the
following Firebase apps:

Platform  Firebase App Id
android   1:860505438253:android:6cf29a670d68848ef8e0a4
ios       1:860505438253:ios:034dc00e7b7a1d3cf8e0a4
```

After doing this step, we were able to view the application configurations in the Firebase > Project Settings > General page for each app (Android, iOS). Please note that your App IDs will be different that what is shown below.



Initialize Firebase in ReadRight Flutter Application

Execute the following Firebase CLI commands to initialize ReadRight app with Firebase.

Add `firebase_core` Flutter plugin

```
% flutter pub add firebase_core
```

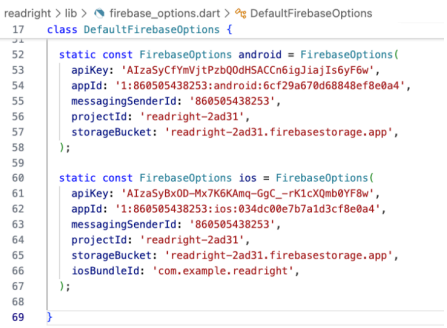
- Ensure that the `readright` Flutter project has latest Firebase configuration.

```
% flutterfire configure
? You have an existing `firebase.json` file and possibly already configured your project for Firebase. Would y
✔ You have an existing `firebase.json` file and possibly already configured your project for Firebase. Would you prefer to reuse the values in your existing `firebase.json` file to configure your project? · yes
```

Here are some additional configuration files to pay attention to after configuring Firebase to work with the ReadRight application. Make sure to update your forked ReadRight app by changing the following settings if the Firebase CLI didn't update them

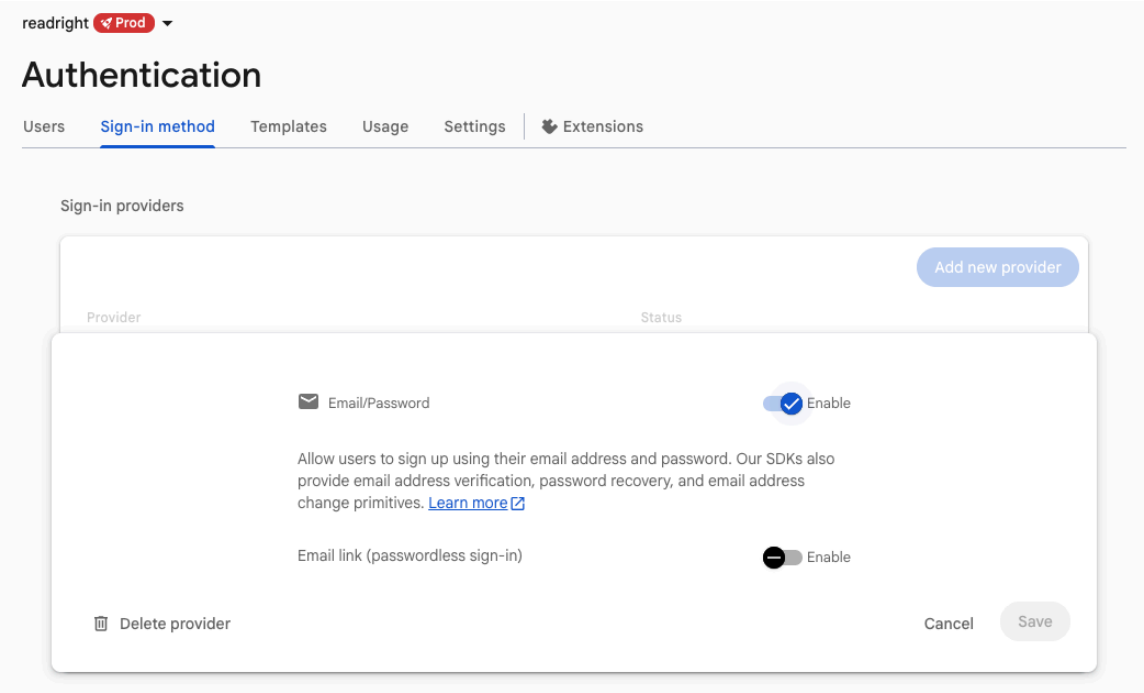
accordingly.

- **readright/firebase.json**
- **readright/lib/firebase_options.dart**
  These files are generated from the previous Firebase CLI run. Verify that the appId and projectId match what is in your *Firebase > Project Settings > General* page.

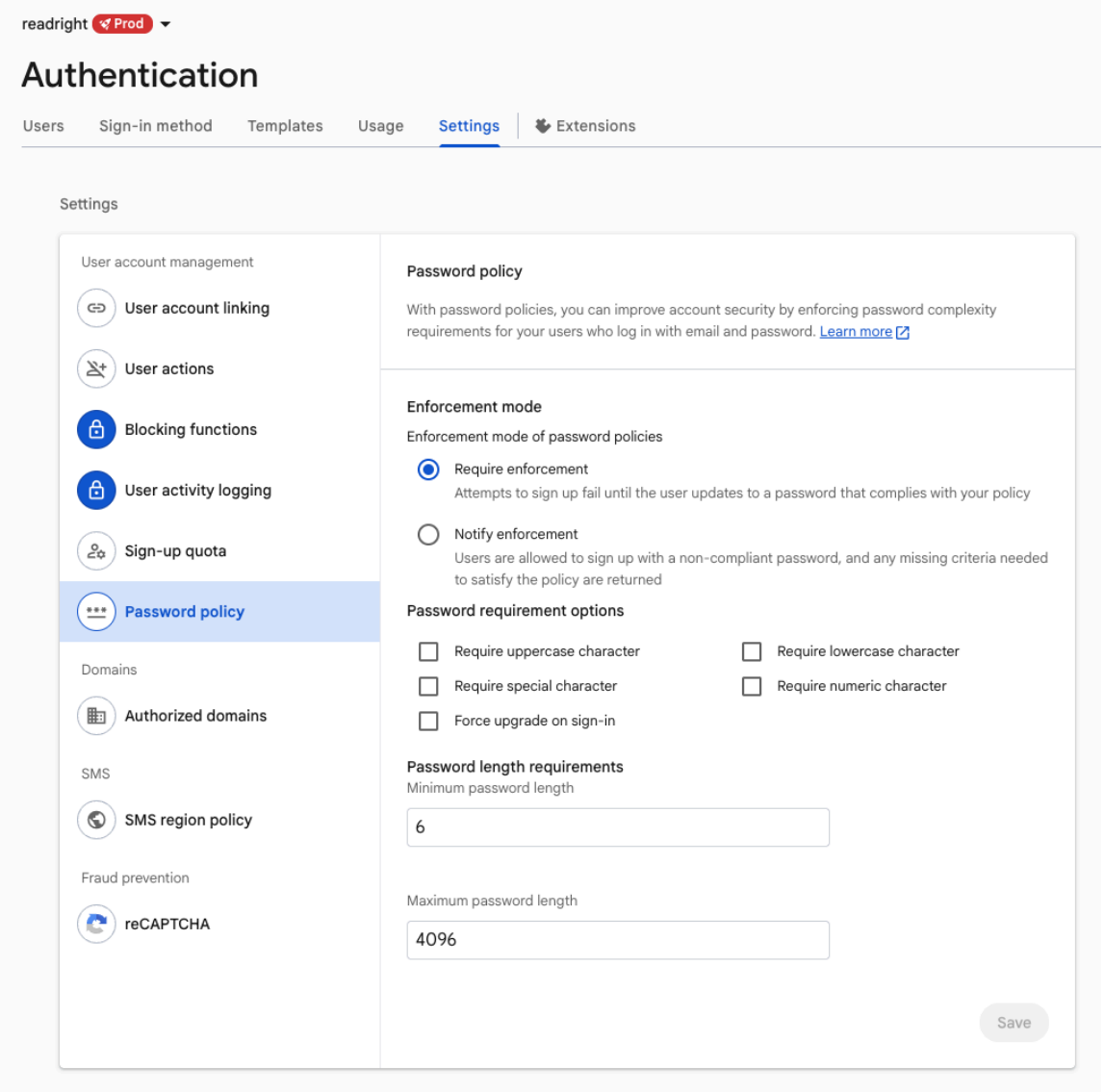


Configure Firebase Authentication

Need to ensure that the *Firebase > Build > Authentication* is configured to use Email/Password provider. Leave *Email link (passwordless sign-in*) disabled.

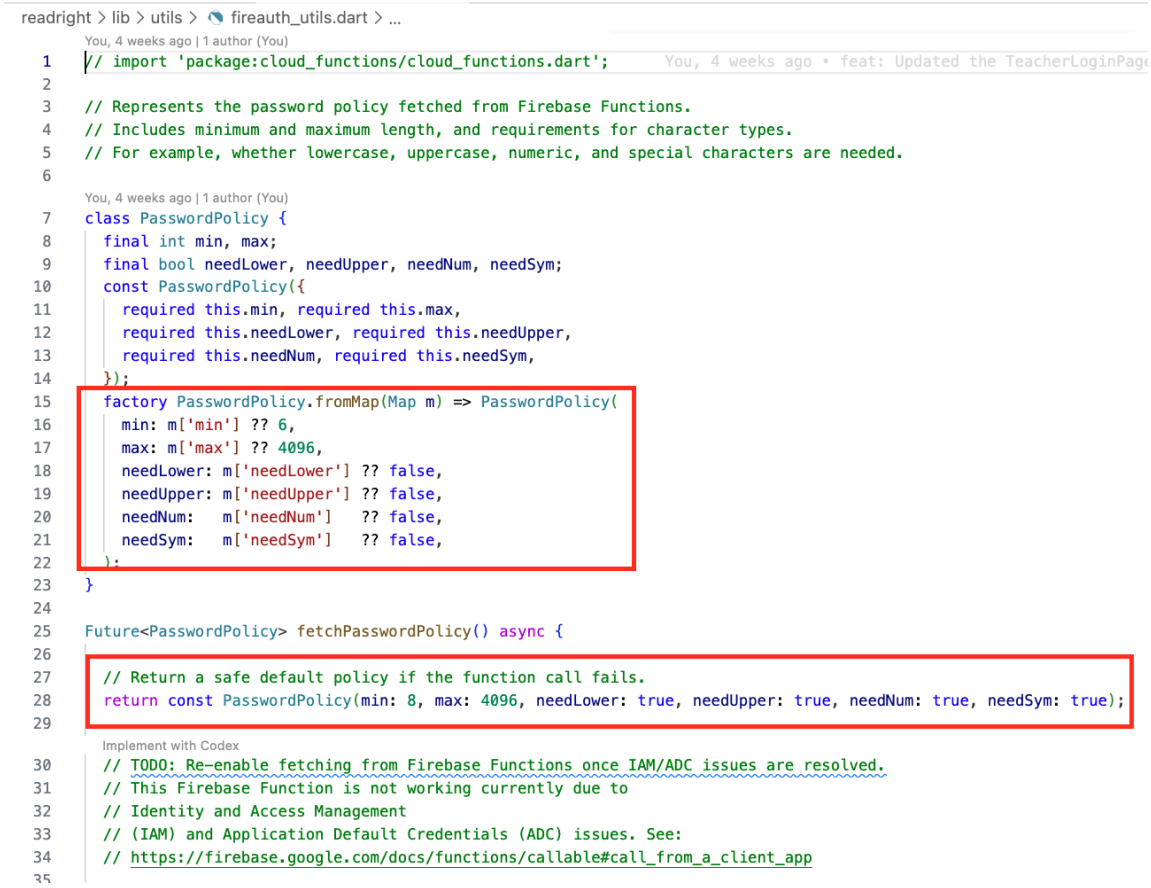


Make sure that you set a Password Policy for ReadRight in the *Firebase Authentication > Settings > Password policy* section. The default settings for ReadRight are indicated below.



The defined Firebase Password Policy is not currently integrated well with the ReadRight application. To ensure that the password policy matches what Firebase Password Policy is doing, make sure to update the following file settings.This file currently does not read a Firebase API to pull this password policy information and therefore the file below handles these localized enforcements.

readright/lib/utils/fireauth_utils.dart
Lines 15 - 22 match what is defined in Firebase as defaults. Line 28 indicates overrides that the ReadRight app will actually adhere too. Please make sure that you update Line 28 to enforce a password policy for the ReadRight app.



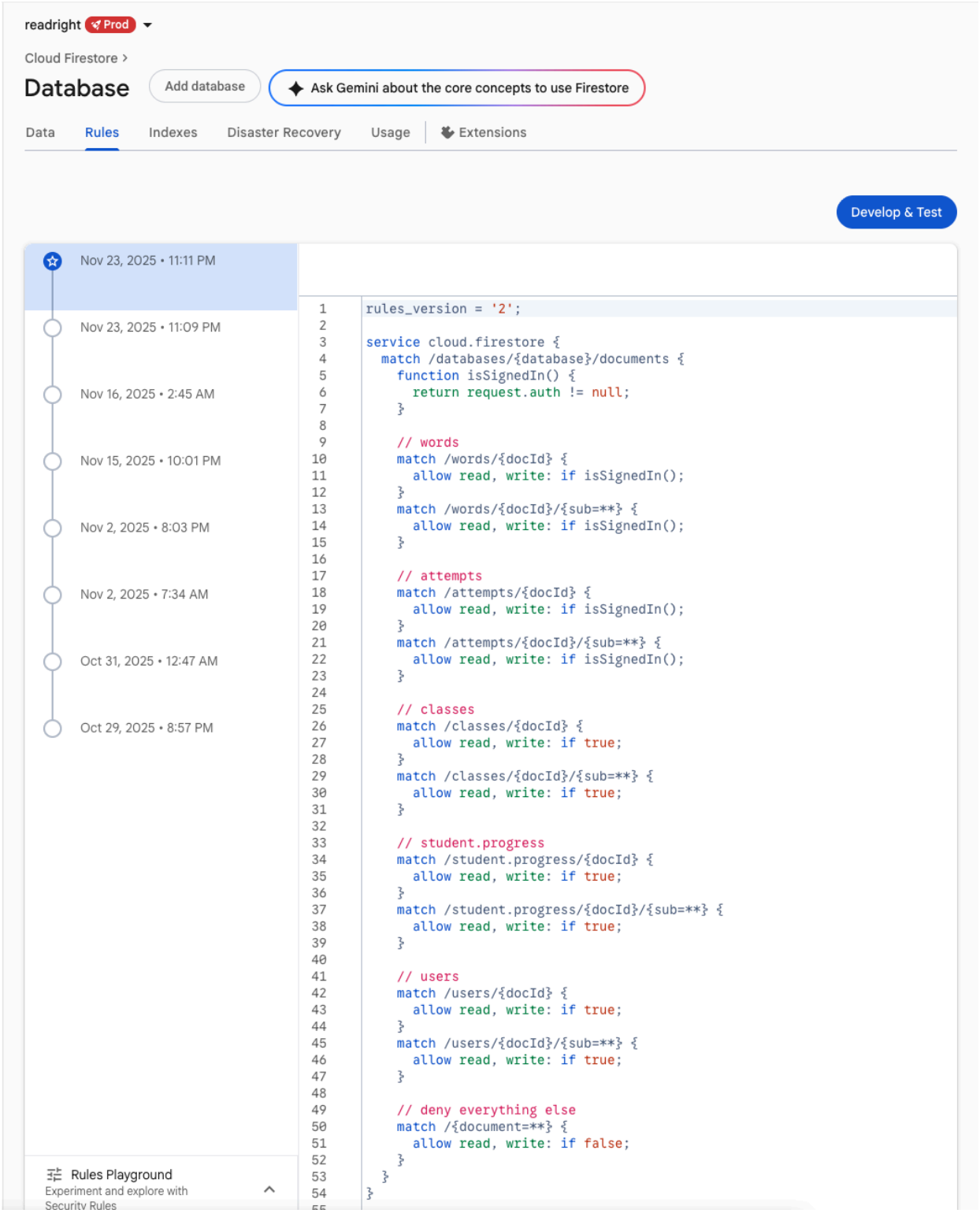
```
readright > lib > utils > fireauth_utils.dart > ...
You, 4 weeks ago | 1 author (You)
// import 'package:cloud_functions/cloud_functions.dart';      You, 4 weeks ago • feat: Updated the TeacherLoginPage

// Represents the password policy fetched from Firebase Functions.
// Includes minimum and maximum length, and requirements for character types.
// For example, whether lowercase, uppercase, numeric, and special characters are needed.

You, 4 weeks ago | 1 author (You)
class PasswordPolicy {
  final int min, max;
  final bool needLower, needUpper, needNum, needSym;
  const PasswordPolicy({
    required this.min, required this.max,
    required this.needLower, required this.needUpper,
    required this.needNum, required this.needSym,
  });
  factory PasswordPolicy.fromMap(Map m) => PasswordPolicy(
    min: m['min'] ?? 6,
    max: m['max'] ?? 4096,
    needLower: m['needLower'] ?? false,
    needUpper: m['needUpper'] ?? false,
    needNum:   m['needNum']   ?? false,
    needSym:   m['needSym']   ?? false,
  );
}

Future<PasswordPolicy> fetchPasswordPolicy() async {

  // Return a safe default policy if the function call fails.
  return const PasswordPolicy(min: 8, max: 4096, needLower: true, needUpper: true, needNum: true, needSym: true);

  Implement with Codex
  // TODO: Re-enable fetching from Firebase Functions once IAM/ADC issues are resolved.
  // This Firebase Function is not working currently due to
  // Identity and Access Management
  // (IAM) and Application Default Credentials (ADC) issues. See:
  // https://firebase.google.com/docs/functions/callable#call_from_a_client_app
```

Configure Firebase Firestore Rules

The database will by default not allow writing in a production setting, so the ReadRight Firestore database will need to be configured to allow this type of access. The example ruleset below can be found at the following location.

readright/lib/services/README_Firestore.md

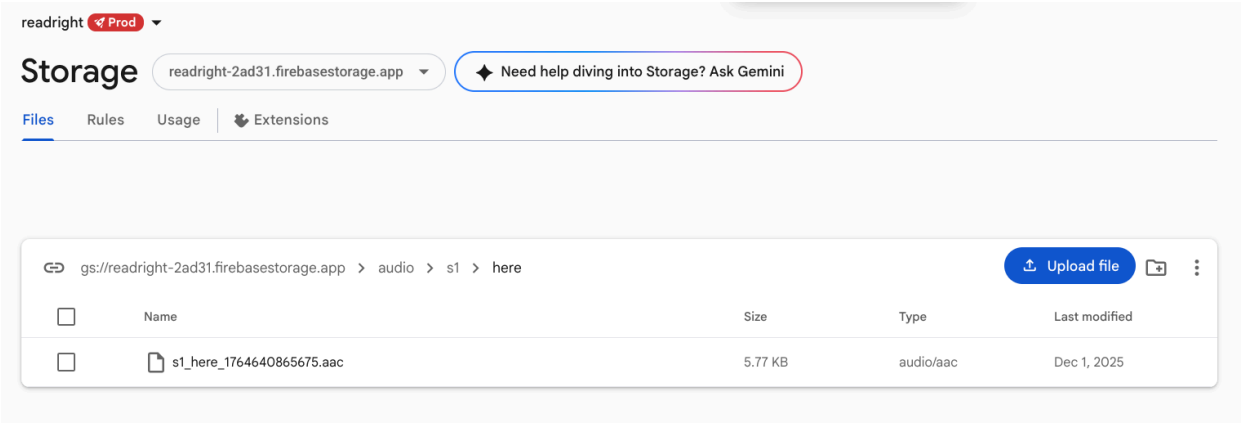
An example ruleset can be found here. Please make adjustments as needed. For additional information please refer to the official documentation guide https://firebase.google.com/docs/firestore/security/get-started.


readright
Prod
Cloud Firestore >
Database
Add database
Ask Gemini about the core concepts to use Firestore
Data
Rules
Indexes
Disaster Recovery
Usage
Extensions
Develop & Test
Nov 23, 2025 • 11:11 PM
Nov 23, 2025 • 11:09 PM
Nov 16, 2025 • 2:45 AM
Nov 15, 2025 • 10:01 PM
Nov 2, 2025 • 8:03 PM
Nov 2, 2025 • 7:34 AM
Oct 31, 2025 • 12:47 AM
Oct 29, 2025 • 8:57 PM
Rules Playground
Experiment and explore with Security Rules
rules_version = '2';
service cloud.firestore {
match /databases/{database}/documents {
function isSignedIn() {
return request.auth != null;
}
// words
match /words/{docId} {
allow read, write: if isSignedIn();
}
match /words/{docId}/{sub=**} {
allow read, write: if isSignedIn();
}
// attempts
match /attempts/{docId} {
allow read, write: if isSignedIn();
}
match /attempts/{docId}/{sub=**} {
allow read, write: if isSignedIn();
}
// classes
match /classes/{docId} {
allow read, write: if true;
}
match /classes/{docId}/{sub=**} {
allow read, write: if true;
}
// student.progress
match /student.progress/{docId} {
allow read, write: if true;
}
match /student.progress/{docId}/{sub=**} {
allow read, write: if true;
}
// users
match /users/{docId} {
allow read, write: if true;
}
match /users/{docId}/{sub=**} {
allow read, write: if true;
}
// deny everything else
match /{document=**} {
allow read, write: if false;
}
}
}

Firebase Storage

The ReadRight application stores student's practice word recordings in the Cloud under the *Build* > *Storage* section. Files are grouped by student username and then by practice word. Multiple recording attempts are store if class audio retention is enabled (off by default). Authentication and security rules also apply to files save to Firebase Cloud Storage.

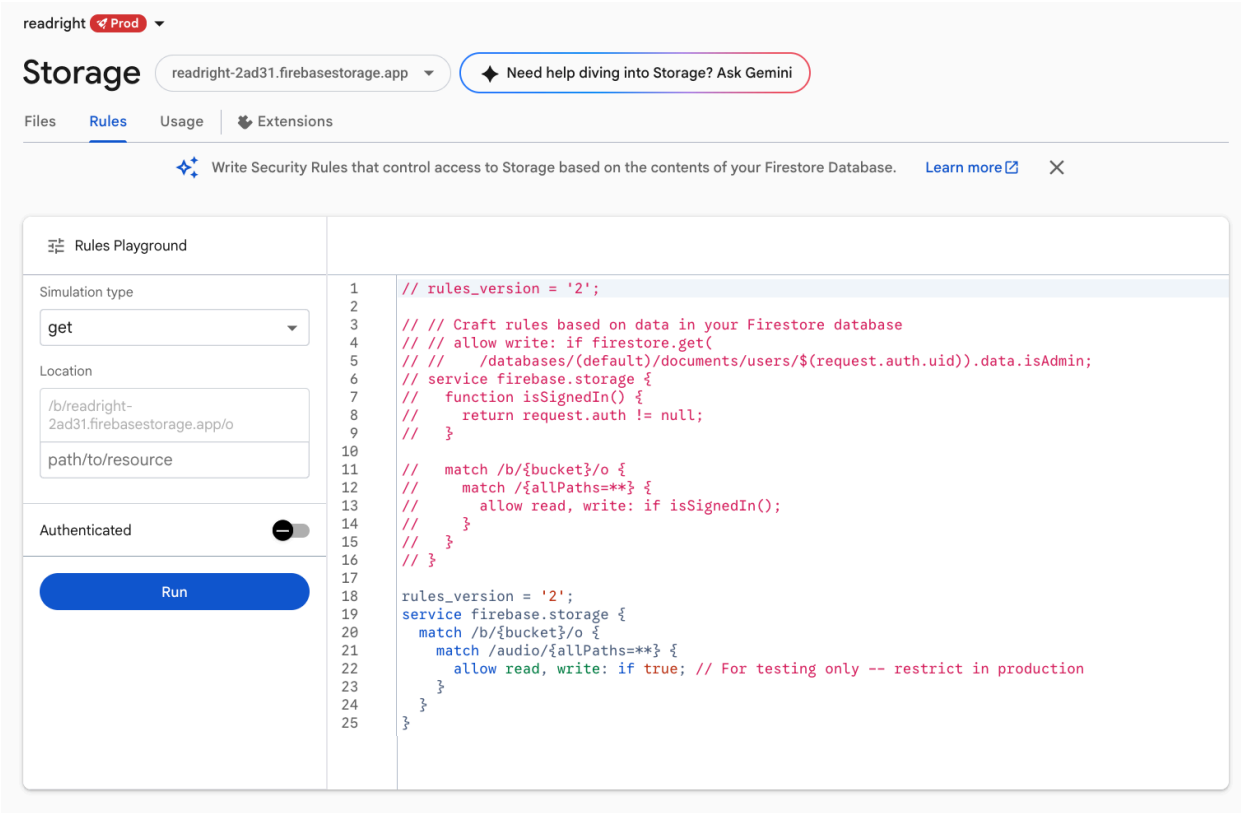
The example below is for */audio/s1/here* which is for student username '*s1'* and the practice word '*here*'. The folder includes one attempt *'s1_here_1764640865675.aac'* which also includes the username and practice word along with Epoch timestamp and AAC codec recording.



Security ruleset for Firebase Cloud Storage is as follows. This limits read and writes to the the '/audio/' folder.

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /audio/{allPaths=**} {
      allow read, write: if true; // For testing only -- restrict in production
    }
  }
```

}


readright
Prod
Storage
readright-2ad31.firebasestorage.app
Need help diving into Storage? Ask Gemini
Files
Rules
Usage
Extensions
Write Security Rules that control access to Storage based on the contents of your Firestore Database.
Learn more
Rules Playground
Simulation type
get
Location
/b/readright-2ad31.firebasestorage.app/o
path/to/resource
Authenticated
Run
1 // rules_version = '2';
2
3 // // Craft rules based on data in your Firestore database
4 // // allow write: if firestore.get(
5 // // /databases/(default)/documents/users/$(request.auth.uid)).data.isAdmin;
6 // service firebase.storage {
7 // function isSignedIn() {
8 // return request.auth != null;
9 // }
10
11 // match /b/{bucket}/o {
12 // match /{allPaths=**} {
13 // allow read, write: if isSignedIn();
14 // }
15 // }
16 // }
17
18 rules_version = '2';
19 service firebase.storage {
20 match /b/{bucket}/o {
21 match /audio/{allPaths=**} {
22 allow read, write: if true; // For testing only -- restrict in production
23 }
24 }
25 }