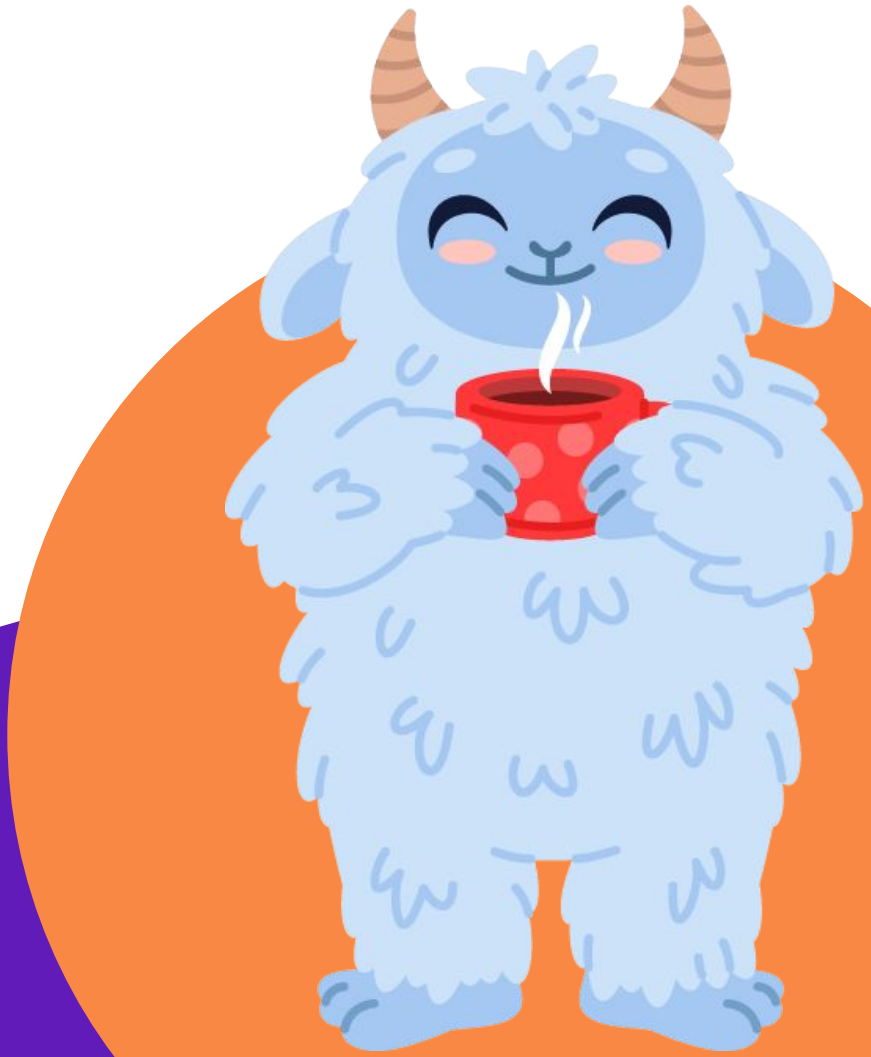
Read Right

Team Project Final Video Presentation

Group 4: Grads Rock



Grads Rock
(Group 4)

Team Lead: Zachary Trabookis
Developer: Jonathan Gilreath
Developer: Noah Carter

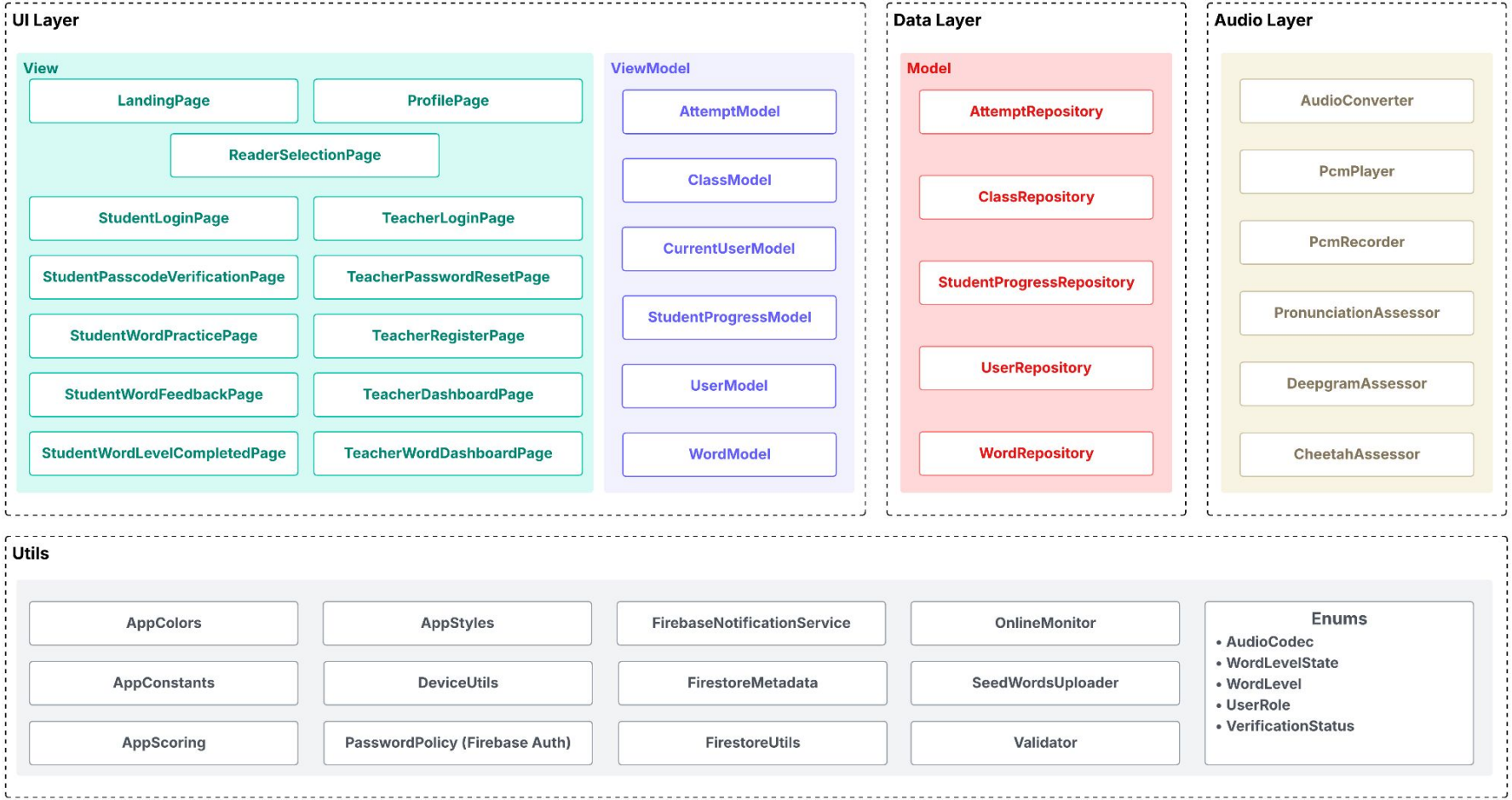
A team of 3 graduate student developers for the CPSC 6150 class. Built a Flutter app *ReadRight* that can be deployed to both Android and iOS platforms.

Release 1.0

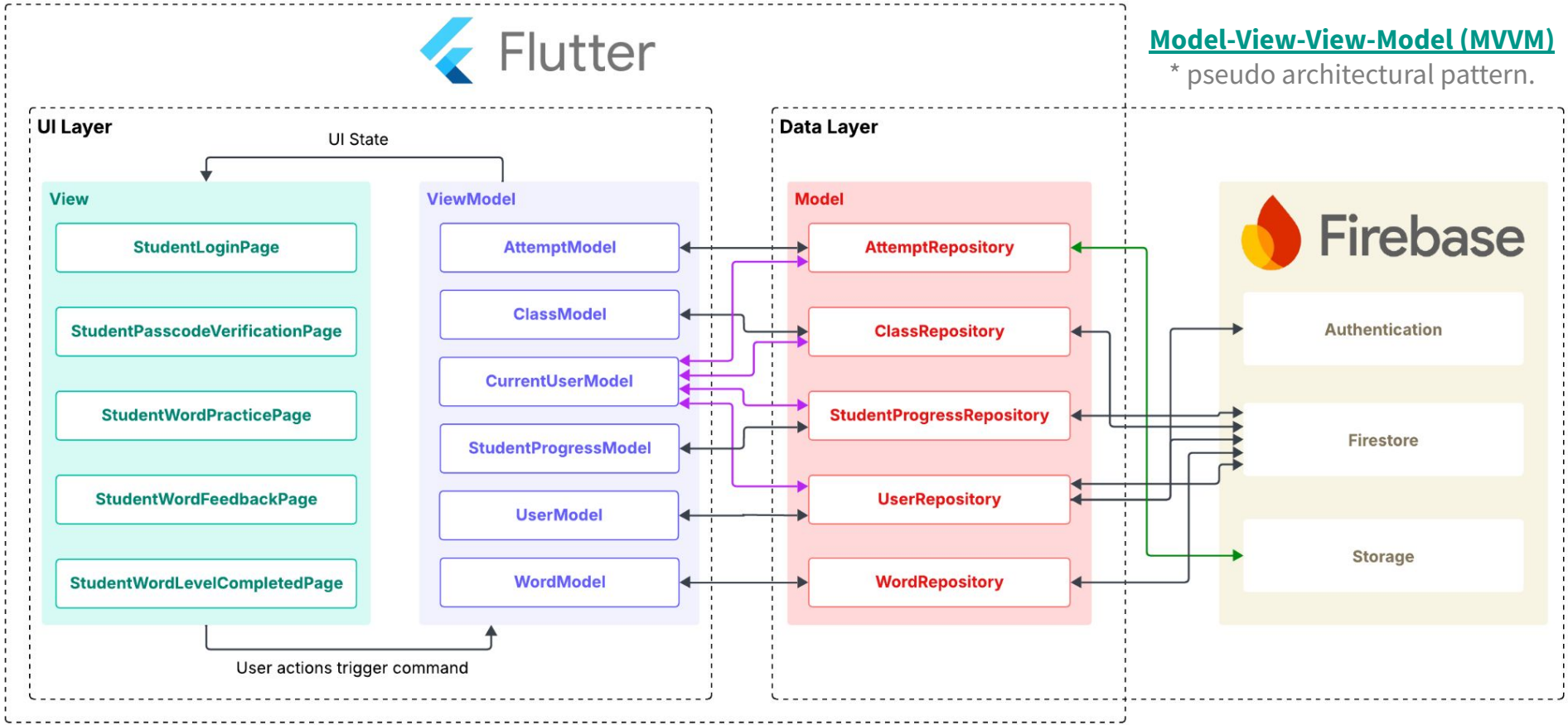
https://github.com/ztraboo/readright/releases/tag/release-1.0

Technical Architecture Overview

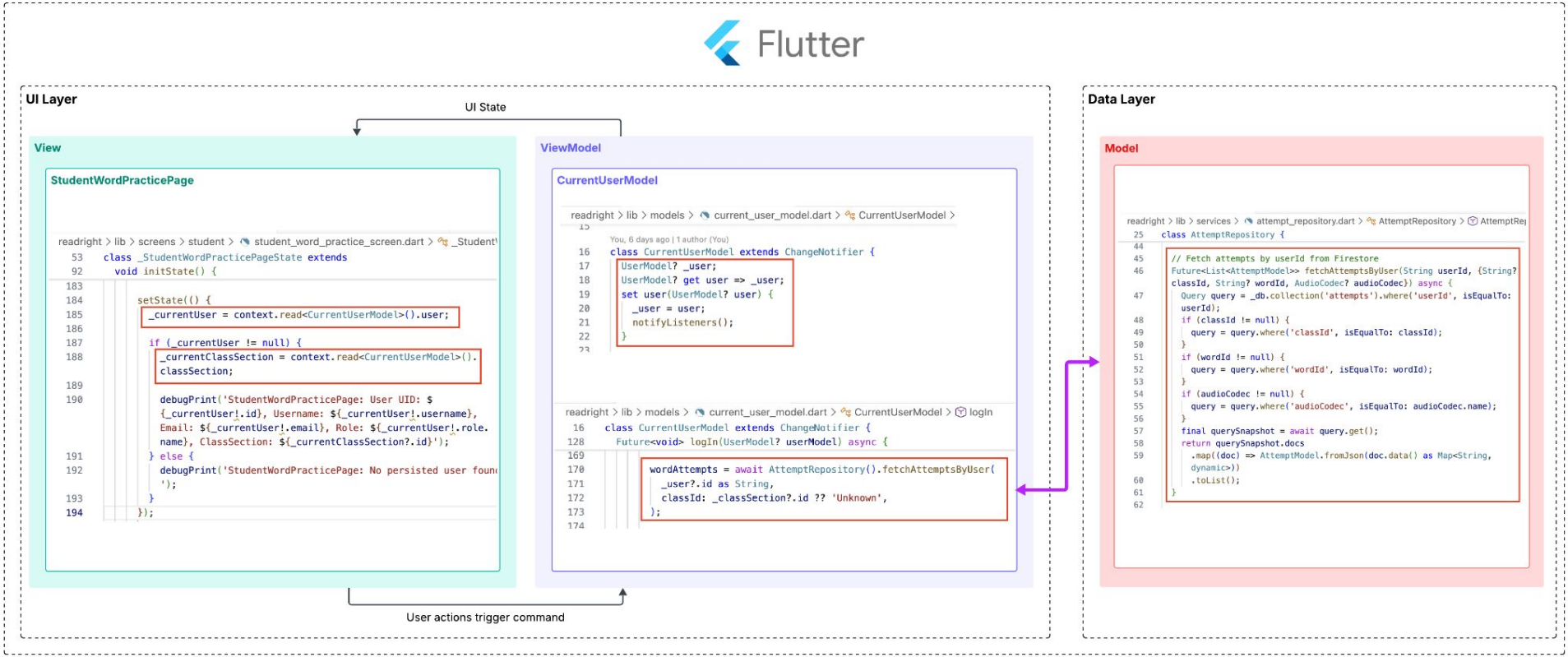
ReadRight - Audio, Models, Screens, Services, Utils



ReadRight - Architecture

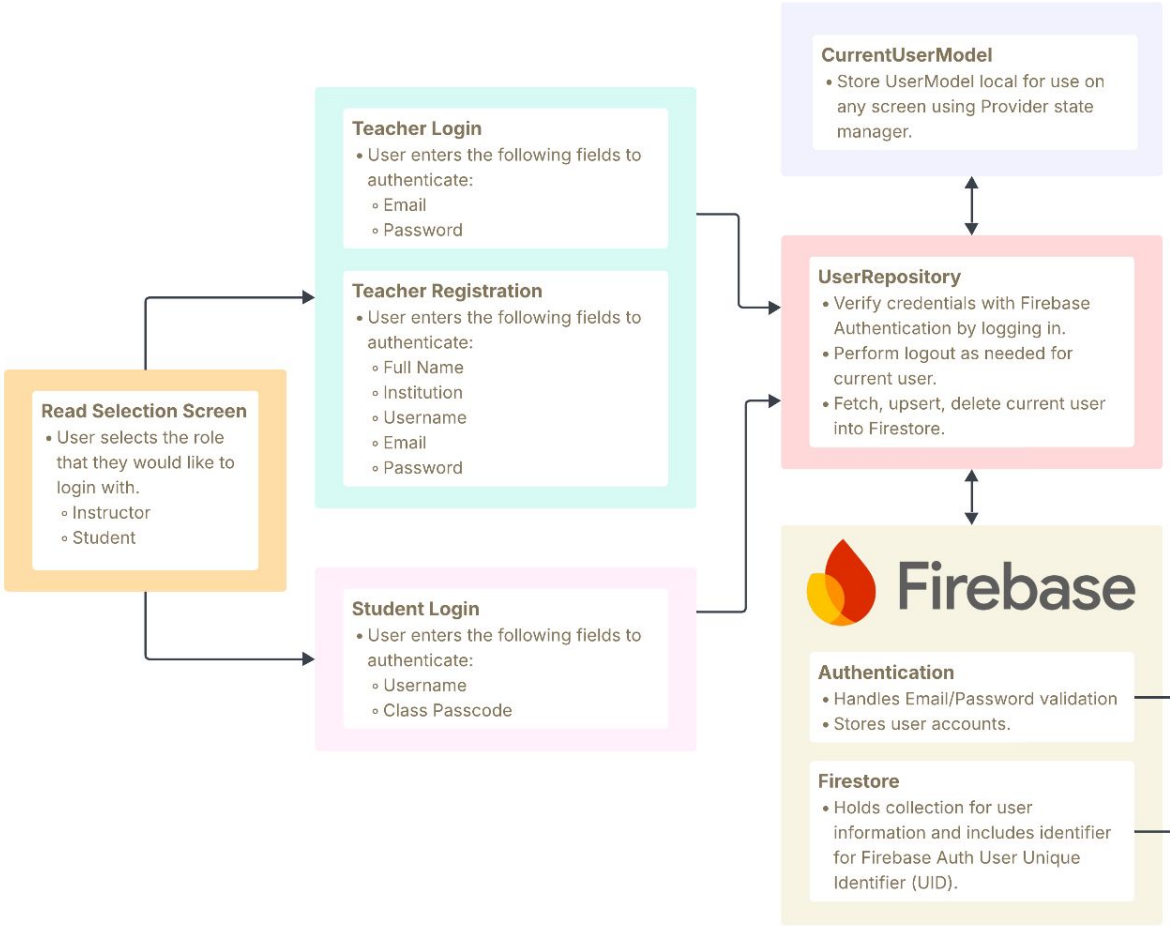


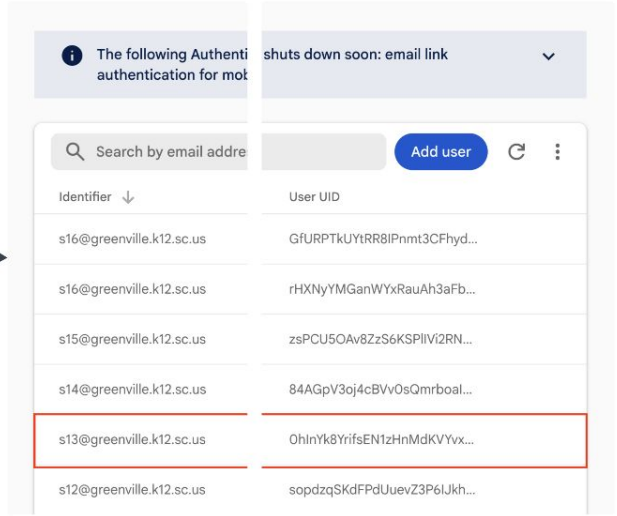
Design Pattern - Model-View-ViewModel (MVVM): Example

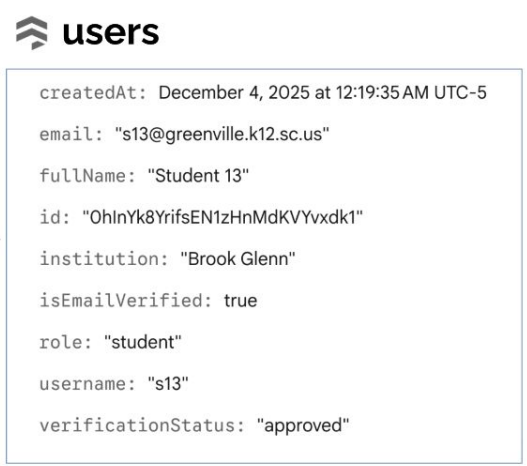


Major Modules

Major Module - Authentication







Major Module - Audio Recording, Playback, Assess

Audio Stream Recorder
- Int16 Pulse-code modulation (PCM) Dart audio stream (mono, 16 kHz) that's ideal for both student recording and playback and Speech-To-Text (STT) providers.
- Handle device microphone permission status.
- Provides start, stop, and clear buffer operations for recording audio to memory stream.
- Convert audio stream PCM bytes to Int16List for STT.

Audio Stream Player
- Subscribe to PCM audio stream recorder.
- Providers play and stop operations of PCM audio stream recorder.

Audio Converters
Need a way to conver the audio stream information to encoded format for STT and audio playback.
- On device audio conversion.
- PCM to WAV
- PCM to AAC
- WAV to AAC

On-Device Accessor (Cheetah)
- Failover accessor when app cannot connect to cloud accessor.
- Accepts PCM audio stream as input for transcribing.
- Provides a real-time transcription with low latency on-device.
- Transcribes voice as it's spoken with instantaneous text.
- Cross-platform.
- Audio processing never leaves device for added security.
- Assesses practice word against transcribe text of audio input to evaluate student's attempt.

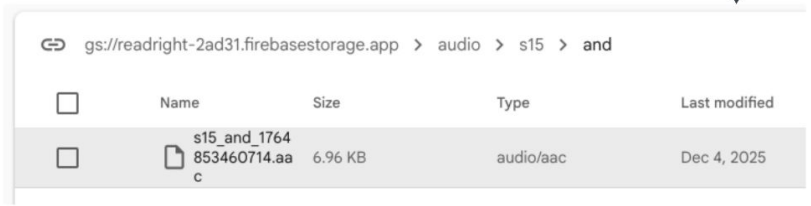
Cloud Accessor (Deepgram)
- Primary accessor for app.
- Accepts PCM audio stream as input for transcribing.
- Provides a real-time transcription with low latency from cloud.
- Transcribes voice with fewer errors and rich transcription features.
- Cross-platform.
- Assesses practice word against transcribe text of audio input to evaluate student's attempt.

Firebase

Firestore
- Holds collection for attempts for a student.

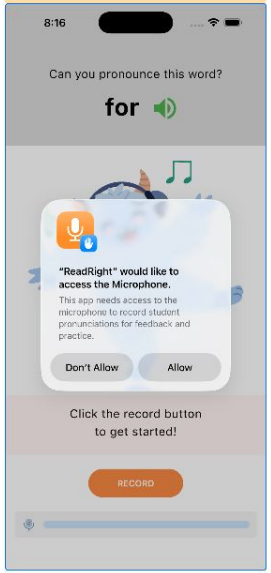
attempts

```
audioCodec: "AAC"
audioPath: "audio/s15/and/s15_and_1764853460714.aac"
classId: "3d6310ad23f6a29ae2352d5e6749df6258ab8a1ca3a34f92de88d6e24a8479cf"
confidence: 0.9739615
createdAt: December 4, 2025 at 8:04:23 AM UTC-5
createdBy: "zsPCU5OAv8ZzS6KSPlIVi2RNPdw1"
deviceOS: "Version 26.1 (Build 23B86)"
devicePlatform: "iOS"
```

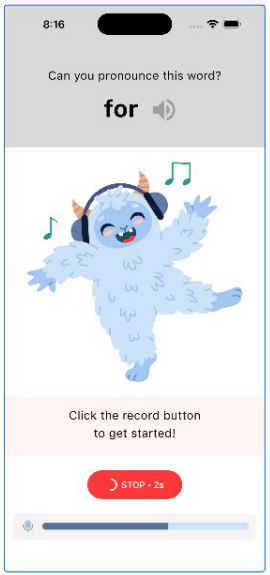


Major Module - Student: Recording and Assess

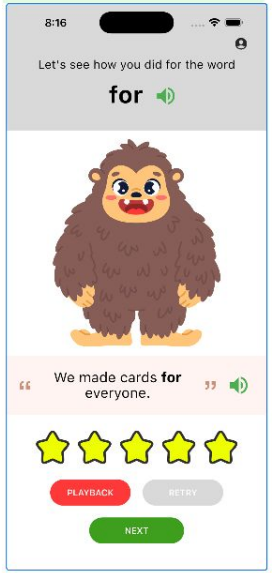
Word Displayed
- The next practice word is displayed to the student but no audio feedback is provided.
- An audio button is displayed for help only. When clicked it will provided audio feedback of the practice word.
- Microphone permissions need to be granted to record.
- Student clicks RECORD to begin recording.



Recording Student
- Audio is being recorded from the student and they have 3 seconds to respond with the practice word. They can click STOP button to end the recording early.
- The audio button for the practice word is disabled to prevent being picked up by the microphone.
- Microphone decipher feedback adjusts as the student records to help them visually see their levels and show that they are recording.



Pronunciation Evaluation
- Recorded audio is transribed by Speech-To-Text (STT) service
- The STT access then compares the transcript with practice word providing a score value that is stored in Firebase Firestore attempts collection.
- A text string length evaluation is 30% and phonetic pronunciation of word is 70% of the grade weight.
- If audio retention is enabled, then the audio is saved in Firebase Storage for playback by the teacher.



Firebase

Firestore
- Holds collection for words, attempts, student.progress for a student.

attempts

```
audioCodec: "AAC"
audioPath: "audio/s15/for/s15_for_1765242987341.aac"
classId: "3d6310ad23f6a29ae2352d5e6749df6258ab8a1ca3a34f92de88d6e24a8479cf"
confidence: 0.7798461
createdAt: December 8, 2025 at 8:16:29 PM UTC-5
createdBy: "zsPCU5OAv8ZzS6KSPlIVi2RNPdw1"
deviceOS: "Version 26.1 (Build 23B86)"
devicePlatform: "iOS"
durationMS: 2901
id: "d7f9ba8d3b77a3d5ca26e5242bf95ff47341434a57302a4db3e22d76d0e2d8b8"
score: 1
speechToTextTranscript: "for"
updatedAt: December 8, 2025 at 8:16:29 PM UTC-5
```

gs://readright-2ad31.firebasestorage.app › audio › s15 › for

Name	Size	Type	Last modified
s15_for_1765242987341.aac	10.04 KB	audio/aac	Dec 8, 2025

words

```
createdAt: November 16, 2025 at 2:50:44 AM UTC-5
createdBy: "LtJ3FyRTJaQ7nZpBHrgInX7a72I3"
id: "98a507ac16b2c3c6840ebda34b5e06e25e06d37d5676e9199bdb19f77ef79d55"
level: "Pre-Primer"
levelOrder: 10
sentences
  0: "This gift is for you."
  1: "Save a seat for me."
  2: "We made cards for everyone."
text: "for"
updatedAt: December 1, 2025 at 11:02:32 PM UTC-5
```

student.progress

```
averageWordAttemptScore: 0.7262500000000001
class: "3d6310ad23f6a29ae2352d5e6749df6258ab8a1ca3a34f92de88d6e24a8479c
countWordsAttempted: 16
countWordsCompleted: 11
createdAt: December 4, 2025 at 1:57:35 AM UTC-5
currentWordLevel: "Pre-Primer"
uid: "zsPCU5OAv8ZzS6KSPlIVi2RNPdw1"
updatedAt: December 8, 2025 at 8:16:30 PM UTC-5
wordAttemptIds: ["2a884e3519845596be1000fb...]
wordCompletedIds: ["b659df576895028358c48b2c...]
wordLevelsCompleted: {Custom: true}
wordStruggledIds: ["b5dcab9bd8d1e0a3b111a077...]
```

Major Module - Student: Evaluation

Firebase

Firestore
- Holds collection for words, attempts, student.progress for a student.

attempts

```
audioCodec: "AAC"
audioPath: "audio/s15/for/s15_for_1765242987341.aac"
classId: "3d6310ad23f6a29ae2352d5e6749df6258ab8a1ca3a34f92de88d6e24a8479cf"
confidence: 0.7798461
createdAt: December 8, 2025 at 8:16:29 PM UTC-5
createdBy: "zsPCU5OAv8ZzS6KSPIlVi2RNPdw1"
deviceOS: "Version 26.1 (Build 23B86)"
devicePlatform: "iOS"
durationMS: 2901
id: "d7f9ba8d3b77a3d5ca26e5242bf95ff47341434a57302a4db3e22d76d0e2d8b8"
score: 1
speechToTextTranscript: "for"
updatedAt: December 8, 2025 at 8:16:29 PM UTC-5
```

gs://readright-2ad31.firebasestorage.app > audio > s15 > for

Name	Size	Type	Last modified
s15_for_1765242987341.aac	10.04 KB	audio/aac	Dec 8, 2025

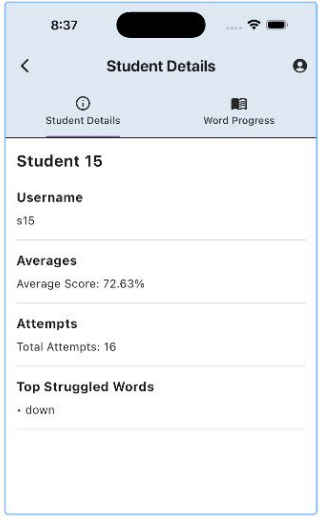
words

```
createdAt: November 16, 2025 at 2:50:44 AM UTC-5
createdBy: "LtJ3FyRTJaQ7nZpBHrgInX7a72I3"
id: "98a507ac16b2c3c6840ebda34b5e06e25e06d37d5676e9199bdb19f77ef79d55"
level: "Pre-Primer"
levelOrder: 10
sentences
  0: "This gift is for you."
  1: "Save a seat for me."
  2: "We made cards for everyone."
text: "for"
updatedAt: December 1, 2025 at 11:02:32 PM UTC-5
```

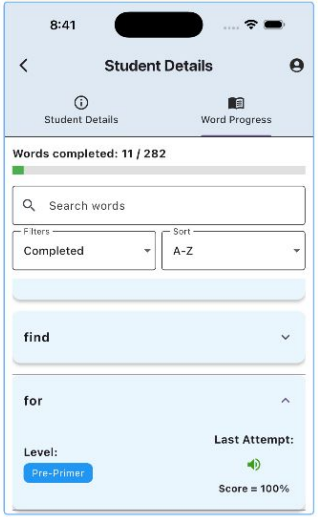
student.progress

```
averageWordAttemptScore: 0.7262500000000001
class: "3d6310ad23f6a29ae2352d5e6749df6258ab8a1ca3a34f92de88d6e24a8479c
countWordsAttempted: 16
countWordsCompleted: 11
createdAt: December 4, 2025 at 1:57:35 AM UTC-5
currentWordLevel: "Pre-Primer"
uid: "zsPCU5OAv8ZzS6KSPIlVi2RNPdw1"
updatedAt: December 8, 2025 at 8:16:30 PM UTC-5
wordAttemptIds: ["2a884e3519845596be1000fb...]
wordCompletedIds: ["b659df576895028358c48b2c...]
wordLevelsCompleted: {Custom: true}
wordStruggledIds: ["b5dcab9bd8d1e0a3b111a077...]
```

Student Progress Overview
- Attempt progress and average score percentages are displayed to the teacher student details page.
- Strugged words are also shown.



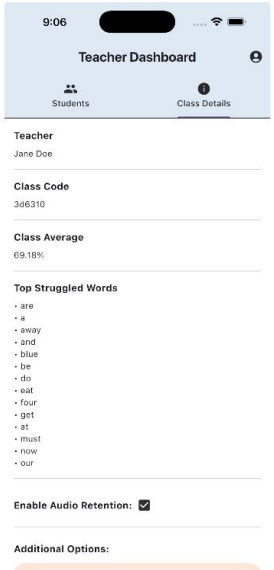
Student Progress Word Progress
- The teacher can view the student practice words completed, see individual word scores, and play back recorded audio from the learner if audio retention is enabled.



Major Module - Teacher: Class Progress

Student Dashboard
- Shows all students progress of Dolche word completion for the course.
- Adding new students and filter is available.

Class Details
- Displays teacher, class code for student passcode credential.
- Class Average is shown for all student word attempts.
- Top struggling words for all students in the course are shown (minimal 1 attempt failing to show).
- Enable Audio Retention (enabled/disabled).
- Word List and Export CSV options.



Firebase

Firestore
- Holds collection for words, attempts, student.progress for a student.

attempts


```
audioCodec: "AAC"
audioPath: "audio/s15/for/s15_for_1765242987341.aac"
classId: "3d6310ad23f6a29ae2352d5e6749df6258ab8a1ca3a34f92de88d6e24a8479cf"
confidence: 0.7798461
createdAt: December 8, 2025 at 8:16:29 PM UTC-5
createdBy: "zsPCU5OAv8ZzS6KSPlIVi2RNPdw1"
deviceOS: "Version 26.1 (Build 23B86)"
devicePlatform: "iOS"
durationMS: 2901
id: "d7f9ba8d3b77a3d5ca26e5242bf95ff47341434a57302a4db3e22d76d0e2d8b8"
score: 1
speechToTextTranscript: "for"
updatedAt: December 8, 2025 at 8:16:29 PM UTC-5
```


words


```
createdAt: November 16, 2025 at 2:50:44 AM UTC-5
createdBy: "LtJ3FyRTJaQ7nZpBHrgInX7a72I3"
id: "98a507ac16b2c3c6840ebda34b5e06e25e06d37d5676e9199bdb19f77ef79d55"
level: "Pre-Primer"
levelOrder: 10
sentences
  0: "This gift is for you."
  1: "Save a seat for me."
  2: "We made cards for everyone."
text: "for"
updatedAt: December 1, 2025 at 11:02:32 PM UTC-5
```


classes


```
audioRetention: true
averageClassWordAttemptScore: 0.6918097352917425
classCode: "3d6310"
createdAt: December 1, 2025 at 7:52:22 PM UTC-5
createdBy: "2dqDBNAZzbfYH0ez7zOeszHFfT92"
id: "3d6310ad23f6a29ae2352d5e6749df6258ab8a1ca3a34f92de88d6e24a8479cf"
institutionId: "Brook Glenn"
sectionId: "001"
studentIds: ["3ACNI3MOL8dRUOMxaKiXkVAh...]
teacherId: "2dqDBNAZzbfYH0ez7zOeszHFfT92"
totalWordsToComplete: 282
updatedAt: December 4, 2025 at 4:59:13 PM UTC-5
wordStruggledIds: ["b158420c94e5932b0aa9dc39...]
```


student.progress

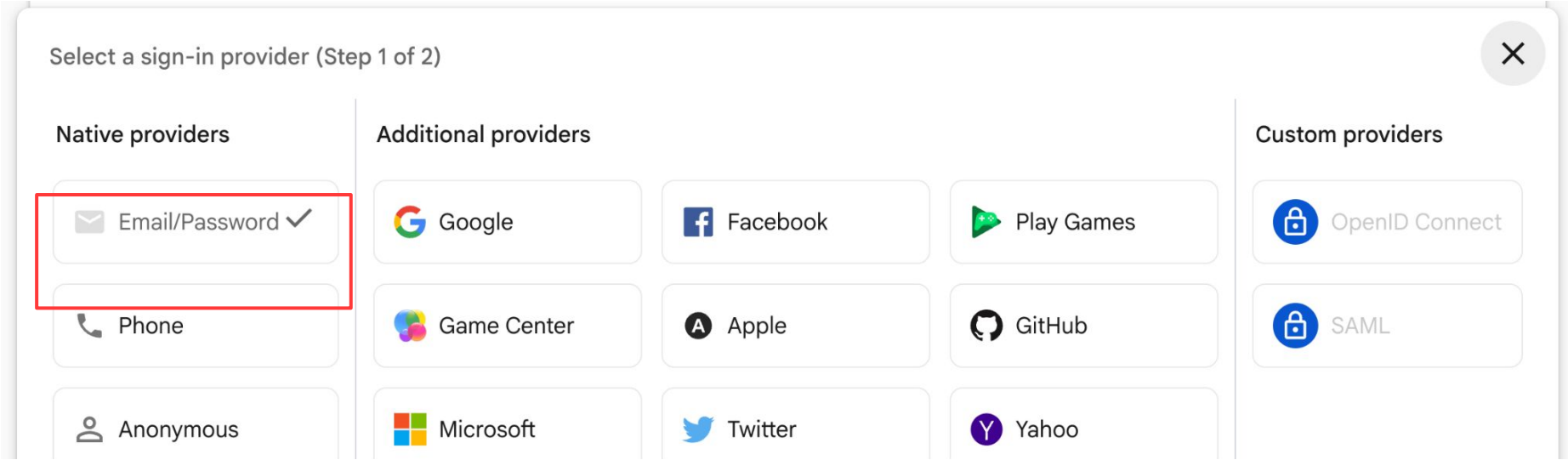

```
averageWordAttemptScore: 0.7262500000000001
class: "3d6310ad23f6a29ae2352d5e6749df6258ab8a1ca3a34f92de88d6e24a8479c
countWordsAttempted: 16
countWordsCompleted: 11
createdAt: December 4, 2025 at 1:57:35 AM UTC-5
currentWordLevel: "Pre-Primer"
uid: "zsPCU5OAv8ZzS6KSPlIVi2RNPdw1"
updatedAt: December 8, 2025 at 8:16:30 PM UTC-5
wordAttemptIds: ["2a884e3519845596be1000fb...]
wordCompletedIds: ["b659df576895028358c48b2c...]
wordLevelsCompleted: {Custom: true}
wordStruggledIds: ["b5dcab9bd8d1e0a3b111a077...]
```

Authentication Providers

Firebase Authentication - Providers

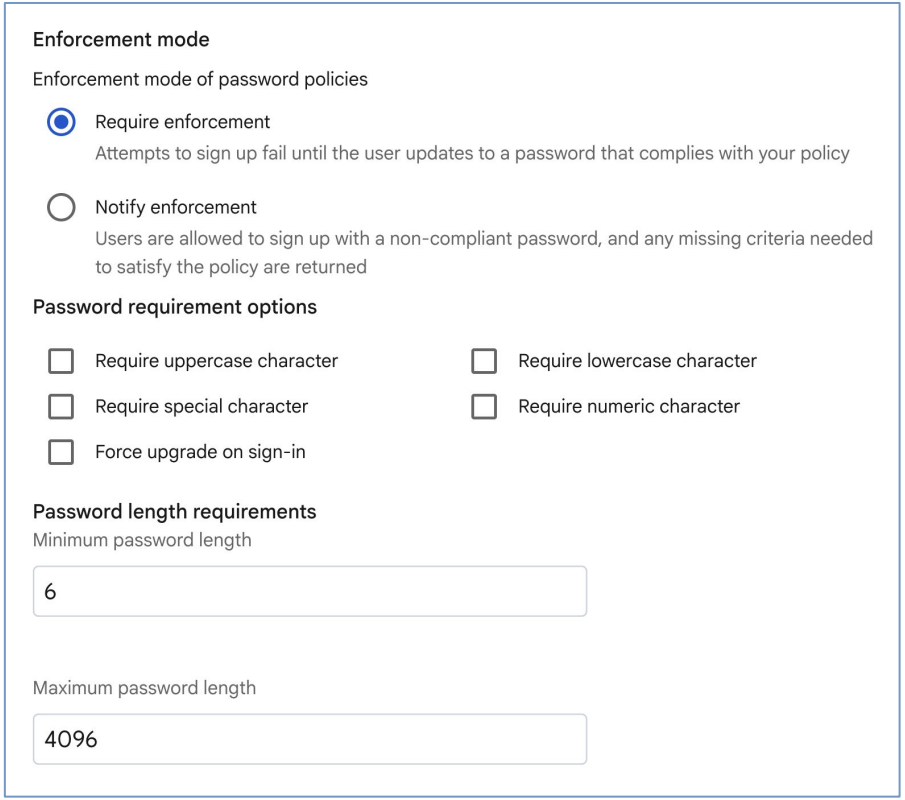
Overview

Here are the different Providers offered by Firebase Authentication. ReadRight uses **Email/Password** native provider.



Firebase Authentication - Password Policy

Student Password Policy



Instructor Password Policy

readright > lib > utils > fireauth_utils.dart > ...

You, last month | 1 author (You)

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

// 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.

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,
  );
}
```

You, last month • feat: Updated the TeacherLoginPage to validate …

```dart
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);

}
```

Data Structure And Storage

Firebase Firestore - collections

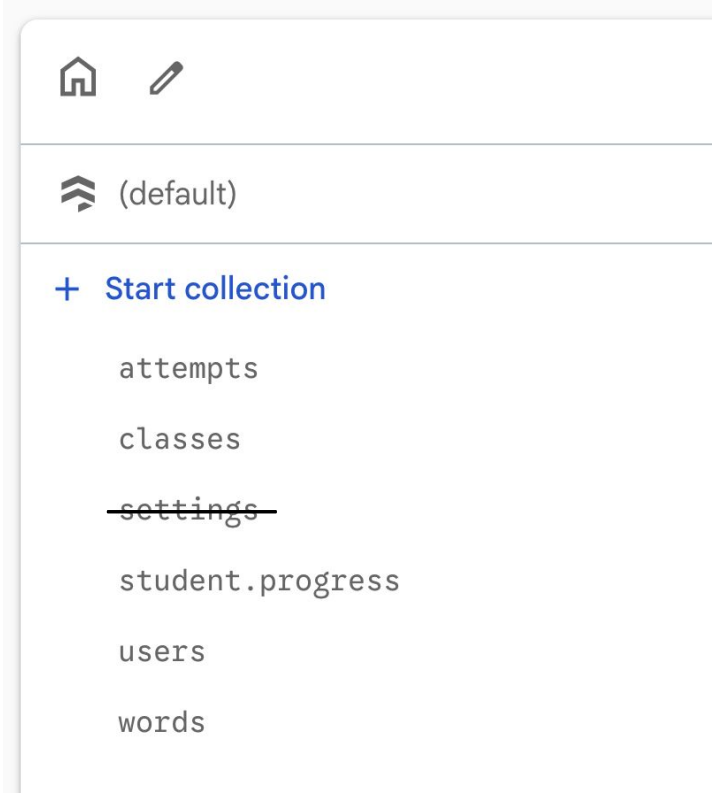
Metadata

Refer to the Data Model Diagram for detailed breakdown of each field.

Overview

Holds majority of the data for the ReadRight app. Data is segregated into different collections for easy lookup by the application.

- **attempts** holds student practice word attempts.
- **classes** holds details about class section, passcode for student login and average word score for all students.
- **student.progress** holds student attempts and keeps track of completed and struggled practice words.
- **users** holds additional metadata about Firebase authentication account for app.
- **words** holds practice word information to include text word and sentences.



Firebase Firestore - users

Overview
Holds user details for the ReadRight app outside of the Firebase Authentication service.

- The **id** field reflects the Firebase authentication **User Unique Identifier (UUID)** value.
- Other fields like **email**, **fullName**, **username** help identify the learner.
- The **role** represents application level control (e.g. teacher, student).

```
createdAt: December 4, 2025 at 12:19:35 AM UTC-5
email: "s13@greenville.k12.sc.us"
fullName: "Student 13"
id: "OhInYk8YrifsEN1zHnMdKVYvxdk1"
institution: "Brook Glenn"
isEmailVerified: true
role: "student"
username: "s13"
verificationStatus: "approved"
```

Flutter - Model and Service - Users

readright > lib > models > user_model.dart > UserModel

You, last month | 1 author (You)

```dart
import 'package:readright/utils/enums.dart';


/// Model class representing user data.
class UserModel {
  // This 'id' value will be updated with the FirestoreAuth UID
  // upon user creation. No need to set it manually.
  final String? id;

  final String fullName;
  final String email;
  final UserRole role;
  final String local;
  final String institution;
  final String username;

  final bool isEmailVerified;
  final VerificationStatus verificationStatus;

  /// Constructor for UserModel.
  UserModel({
    String? id,
    required this.email,
    this.fullName = '',
    this.role = UserRole.student, // Default role is student, teacher will be assi
    this.local = 'en-US',
    this.institution = '',
    this.username = '',
    this.isEmailVerified = false,
```

readright > lib > services > user_repository.dart > ...

```dart
/// Repository for managing user data in Firestore
/// Handles CRUD operations for UserModel instances.
///
/// Collection: 'users'
///
/// createdAt: timestamp, servertimestamp when the document was created.
/// createdBy: string, Firebase authentication account User UID      You, last month • feat:
/// email: string, user's email address.
/// fullName: string, user's full name.
/// id: string, Firebase authentication account User UID
/// isEmailVerified: boolean, whether the user's email has been verified.
/// local: string, user's locale (e.g. "en-US").
/// role: string, user's role (e.g. "student", "teacher").
/// updatedAt: timestamp, servertimestamp when the document was updated.
/// verificationStatus: string, user's verification status (e.g. "unknown", "pending", "veri
```

You, 2 weeks ago | 2 authors (You and one other)

```dart
class UserRepository {
  // Singleton pattern so we don't open multiple DB connections accidentally.
  UserRepository._internal({FirebaseFirestore? firestore, FirebaseAuth? auth})
    : _db = firestore ?? FirebaseFirestore.instance,
      _auth = auth ?? FirebaseAuth.instance;

  // Factory constructor for UserRepository allowing optional injection
  // of FirebaseFirestore and FirebaseAuth instances for testing.
  factory UserRepository({FirebaseFirestore? firestore, FirebaseAuth? auth}) =>
      UserRepository._internal(firestore: firestore, auth: auth);

  /// Create a testable instance backed by an injected [FirebaseFirestore].
  /// Use this in unit tests with a mock Firestore and Auth implementation.
  UserRepository.withFirestoreAndAuth(FirebaseFirestore firestore, FirebaseAuth auth)
```

Firebase Firestore - classes

Overview

Holds a class details that include:

- **averageClassWordAttemptScore** indicates word score average for the class after a word practice attempt is performed.
- **classCode** is the passcode for the students.
- **institutionId** is the school name and **sectionId** indicates section information.
- **teacherId** and **studentIds** are document identifiers for users.

```
audioRetention: true
averageClassWordAttemptScore: 0.6918097352917425
classCode: "3d6310"
createdAt: December 1, 2025 at 7:52:22 PM UTC-5
createdBy: "2dqDBNAZzbfYH0ez7zOeszHFfT92"
id: "3d6310ad23f6a29ae2352d5e6749df6258ab8a1ca3a34f92de88d6e24a8479cf"
institutionId: "Brook Glenn"
sectionId: "001"
studentIds: ["3ACNI3M0L8dRUOMxaKiXkVAh...]
teacherId: "2dqDBNAZzbfYH0ez7zOeszHFfT92"
totalWordsToComplete: 282
updatedAt: December 4, 2025 at 4:59:13 PM UTC-5
```

Flutter - Model and Service - Classes

readright > lib > models > class_model.dart > ClassModel > addAttemptId

You, 3 days ago | 1 author (You)

```dart
import 'package:readright/services/user_repository.dart';
import 'package:readright/services/student_progress_repository.dart';
import 'package:readright/utils/app_scoring.dart';
import 'package:readright/utils/firestore_utils.dart';

/// Model class representing a Class entity
```

You, 3 days ago | 1 author (You)

```dart
class ClassModel {

  // This 'id' value will be the Firestore document ID.
  late final String? id;
  final bool audioRetention;
  final double averageClassWordAttemptScore;
  late final String? classCode;
  String institution;
  final String teacherId;
  String sectionId;
  List<String> studentIds;
  List<String> wordStruggledIds;
  int totalWordsToComplete;

  ClassModel({
    String? id,
    this.audioRetention = false,
    this.averageClassWordAttemptScore = 0.0,
    String? classCode,
    this.institution = '',
    this.teacherId = '',
    this.sectionId = '',
    this.studentIds = const [],
```

readright > lib > services > class_repository.dart > ...

```dart
/// Repository for managing class data in Firestore
/// Handles CRUD operations for ClassModel instances.
///
/// Collection: 'classes'
///
/// createdAt: timestamp, servertimestamp when the document was created.
/// createdBy: string, Firebase authentication account User UID
/// id: string, document id (may be generated or provided).
/// name: string, human-friendly class name.
/// averageClassWordAttemptScore: double, average class score (0..100).
/// classCode: string, join code / passcode for the class.
/// teacherId: string, UID of the teacher who owns the class.
/// studentIds: array of student UIDs.
/// topStrugglingWords: array of words for the class.
/// totalWordsToComplete: integer, planned total words for the class.
```

You, 2 weeks ago | 1 author (You)

```dart
class ClassRepository {
  // Singleton pattern so we don't open multiple DB connections accidentally.
  ClassRepository._internal({FirebaseFirestore? firestore, FirebaseAuth? auth})
      : _db = firestore ?? FirebaseFirestore.instance,
        _auth = auth ?? FirebaseAuth.instance;

  // Factory constructor for ClassRepository allowing optional injection
  // of FirebaseFirestore and FirebaseAuth instances for testing.
  factory ClassRepository({FirebaseFirestore? firestore, FirebaseAuth? auth}) =>
      ClassRepository._internal(firestore: firestore, auth: auth);

  /// Create a testable instance backed by an injected [FirebaseFirestore].
  /// Use this in unit tests with a mock Firestore and Auth implementation.
  ClassRepository.withFirestoreAndAuth(FirebaseFirestore firestore, FirebaseAuth auth)
```

Firebase Firestore - words

Overview
Holds individual Dolche word level details that include:

- The **level** identifier identifies difficulty level for the Dolch lists.
- The **levelOrder** indicates when the word should be presented to the student.
- The **sentences** provides multiple examples that the student can view how the practice word is used.
- The **text** holds the String representation of the practice word.

```
createdAt: November 16, 2025 at 2:50:53 AM UTC-5
createdBy: "LtJ3FyRTJaQ7nZpBHrgInX7a72I3"
id: "016bb836a43426c8c78f0777435992e003b9d459191b5669b45f3090f321d
level: "Primer"
levelOrder: 7
sentences
  0 "The night sky is black."
  1 "I have a black backpack."
  2 "The cat has black fur."
text: "black"
updatedAt: December 1, 2025 at 11:02:43 PM UTC-5
```

Flutter - Model and Service - Words

readright > lib > models > word_model.dart > WordModel > WordModel

You, 4 weeks ago | 1 author (You)

```dart
import 'package:readright/utils/enums.dart';
import 'package:readright/utils/firestore_utils.dart';

/// Model class representing a word with its properties.
You, 4 weeks ago | 1 author (You)
class WordModel {
  final String id;
  final String text;
  final WordLevel level;
  final int levelOrder;
  final List<String> sentences;

  /// If [id] is omitted, a deterministic id from `text` and `levelName`. will be
  /// Id generation is delegated to [FirestoreUtils].
  WordModel({
    String? id,
    required this.text,
    required this.level,
    required this.levelOrder,      You, 4 weeks ago • feat: Update `data/seed_word
    required this.sentences,
  }) : id = id ?? FirestoreUtils.generateDeterministicWordId(text, level.name);

  // Create a copy of the current WordModel with optional new values.
  WordModel copyWith({
    String? id,
    String? text,
    WordLevel? level,
    int? levelOrder,
    List<String>? sentences,
```

readright > lib > services > word_respository.dart > ...

```dart
/// Repository for managing word data in Firestore
/// Handles CRUD operations for WordModel instances.
///
/// Collection: 'words'
///
/// createdAt: timestamp, servertimestamp when the document was created.
/// createdBy: string, Firebase authentication account User UID
/// id: string, deterministic SHA-256 based on (namespace, word text, and word category) - u
/// level: string, category of word (e.g. "Sight Words, Phonics Pattern")
/// sentences: array of strings, word used in different sentences.
/// text: string, the word itself.
/// updatedAt: timestamp, servertimestamp when the document was updated.

You, 3 weeks ago | 1 author (You)
class WordRepository {
  // Singleton pattern so we don't open multiple DB connections accidentally.
  WordRepository._internal({FirebaseFirestore? firestore, FirebaseAuth? auth})
    : _db = firestore ?? FirebaseFirestore.instance,
      _auth = auth ?? FirebaseAuth.instance;

  // Factory constructor for WordRepository allowing optional injection
  // of FirebaseFirestore and FirebaseAuth instances for testing.
  factory WordRepository({FirebaseFirestore? firestore, FirebaseAuth? auth}) =>
      WordRepository._internal(firestore: firestore, auth: auth);

  /// Create a testable instance backed by an injected [FirebaseFirestore].
  /// Use this in unit tests with a mock Firestore and Auth implementation.
  WordRepository.withFirestoreAndAuth(FirebaseFirestore firestore, FirebaseAuth auth)
    : _db = firestore, _auth = auth;
```

Firebase Firestore - attempts

Overview

Holds student practice attempt details:

- **audioCodec** and **audioPath** hold audio recording file path and Firebase Storage location.
- The **classId**, **userId**, **wordId** refers to the class, user, and user identifiers to relative collection documents.
- **confidence** and **score** values are from the word pronunciation accessor.
- **deviceOS** and **devicePlatform** indicate mobile device used for the app.
- **durationMS** indicates milliseconds taken to record student for practice word.
- **speechToTextTranscript** indicates the Speech-To-Text (STT) representation of the recorded word.

```
audioCodec: "AAC"
audioPath: "audio/s15/and/s15_and_1764853460714.aac"
classId: "3d6310ad23f6a29ae2352d5e6749df6258ab8a1ca3a34f92de
confidence: 0.9739615
createdAt: December 4, 2025 at 8:04:23 AM UTC-5
createdBy: "zsPCU5OAv8ZzS6KSPlIVi2RNPdw1"
deviceOS: "Version 26.1 (Build 23B86)"
devicePlatform: "iOS"
durationMS: 1877
id: "09bc141f5463a3dbf4280bdea11b1c412409b4f345e60c9cc4a43a1e
score: 1
speechToTextTranscript: "and"
updatedAt: December 4, 2025 at 8:04:23 AM UTC-5
userId: "zsPCU5OAv8ZzS6KSPlIVi2RNPdw1"
wordId: "c463527d23240a381ef16d14410d1158a90b0b5c84e5e9c64e4
```

Flutter - Model and Service - Attempts

readright > lib > models > attempt_model.dart > ...

You, 4 weeks ago | 1 author (You)

```dart
import 'package:readright/utils/enums.dart';
import 'package:readright/utils/firestore_utils.dart';

You, 4 weeks ago | 1 author (You)
class AttemptModel {
  final String id;
  final String classId;
  final String userId;
  final String wordId;
  final String speechToTextTranscript;
  final AudioCodec audioCodec;
  final String audioPath;
  final int durationMS;
  final double confidence; // 0..1 if available
  final double score; // normalized similarity 0..1
  final String devicePlatform;
  final String deviceOS;

  AttemptModel({
    String? id,
    required this.classId,
    required this.userId,
    required this.wordId,
    required this.speechToTextTranscript,
    required this.audioCodec,
    required this.audioPath,
    required this.durationMS,
    required this.confidence,
    required this.score,
```

readright > lib > services > attempt_repository.dart > AttemptRepository > AttemptRepository._internal

You, 4 weeks ago | 1 author (You)

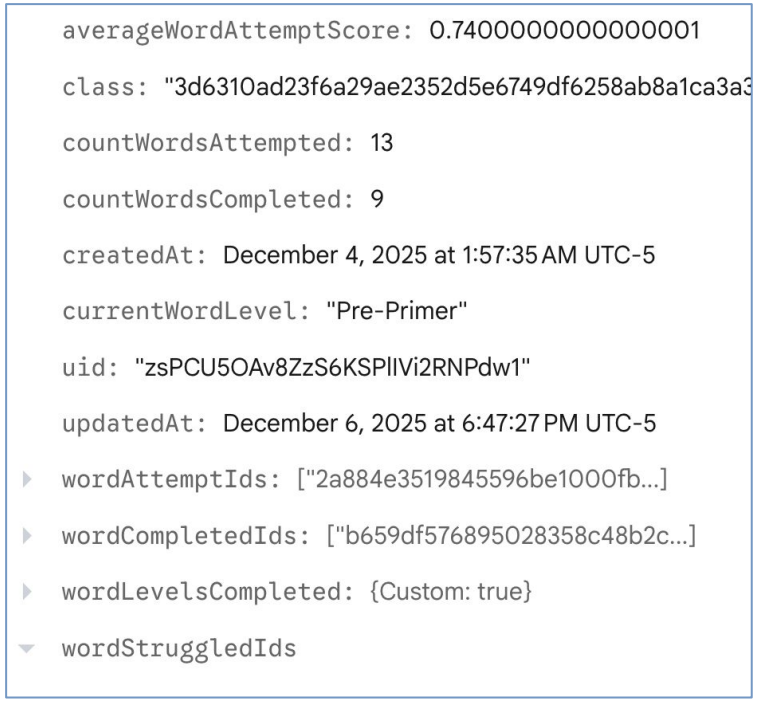
```dart
class AttemptRepository {
  // Singleton pattern so we don't open multiple DB connections accidentally.
  AttemptRepository._internal({FirebaseFirestore? firestore, FirebaseAuth? auth})
      : _db = firestore ?? FirebaseFirestore.instance,
        _auth = auth ?? FirebaseAuth.instance;

  // Factory constructor for AttemptRepository allowing optional injection
  // of FirebaseFirestore and FirebaseAuth instances for testing.
  factory AttemptRepository({FirebaseFirestore? firestore, FirebaseAuth? auth}) =>
      AttemptRepository._internal(firestore: firestore, auth: auth);

  /// Create a testable instance backed by an injected [FirebaseFirestore].
  /// Use this in unit tests with a mock Firestore and Auth implementation.
  AttemptRepository.withFirestoreAndAuth(FirebaseFirestore firestore, FirebaseAuth auth)
      : _db = firestore, _auth = auth;

  // Initialize instance of Cloud Firestore
  final FirebaseFirestore _db;
  final FirebaseAuth _auth;

  // Fetch attempts by userId from Firestore
  Future<List<AttemptModel>> fetchAttemptsByUser(String userId, {String? classId, String?
  wordId, AudioCodec? audioCodec}) async {
    Query query = _db.collection('attempts').where('userId', isEqualTo: userId);
    if (classId != null) {
      query = query.where('classId', isEqualTo: classId);
    }
    if (wordId != null) {
      query = query.where('wordId', isEqualTo: wordId);
```

Firebase Firestore - student.progress

Overview

Holds student practice attempt details:

- **averageWordAttemptScore** indicates average score for individuals across attempts.
- **class** and **uid** are document id references.
- **countWordsAttempted** and **countWordsCompleted** represent count of words attempted or completed by student.
- **currentWordLevel** indicates current Dolch level that the student is in-progress on.
- **wordLevelsCompleted** holds mapping of Dolche levels and their completion status.
- **wordAttemptIds**, **wordCompletedIds**, **wordStruggledIds** holds word document id references for attempted, completed, or non-passing words for student.



Flutter - Model and Service - Student Progress

readright > lib > models > student_progress_model.dart > StudentProgressModel > addAttempt

You, 6 days ago | 1 author (You)

```dart
import 'package:readright/utils/app_scoring.dart';
import 'package:readright/utils/enums.dart';

/// Model representing student progress statistics stored in Firestore.
///
/// - `wordAttemptIds` is a List<String> represents AttemptModel IDs (attempt docu
/// - Internally the model maintains a Set<String> of completed word IDs (wordId)
///   derived from attempts when attempt data is available.
```

You, 6 days ago | 1 author (You)

```dart
class StudentProgressModel {
  /// Average score for the student (e.g. 92.0)
  final double averageWordAttemptScore;

  /// Current word level the student is on.
  WordLevel? currentWordLevel;

  /// Mapping WordLevel and whether the level has been completed.
  Map<WordLevel, bool> wordLevelsCompleted;

  /// Internal list of Attempt IDs (AttemptModel document IDs). When serialized to
  /// is stored as a list of AttemptModel.id strings (document references).
  final List<String> wordAttemptIds;

  /// Internal unique set of completed word IDs (derived from AttemptModel.wordId)
  final Set<String> _wordsCompleted;

  /// Returns a list view of the internally-tracked unique completed word IDs.
  List<String> get wordsCompleted => _wordsCompleted.toList();

  /// Distinct count of completed words (derived from the unique set).
```

readright > lib > services > student_progress_repository.dart > ...

You, last week | 1 author (You)

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

import 'package:readright/models/student_progress_model.dart';
import 'package:readright/utils/firestore_metadata.dart';

/// Repository for managing student progress stored in Firestore.
/// Collection: 'student.progress'
```

You, last week | 1 author (You)

```dart
class StudentProgressRepository {
  StudentProgressRepository._internal({FirebaseFirestore? firestore, FirebaseAuth? auth})
      : _db = firestore ?? FirebaseFirestore.instance,
        _auth = auth ?? FirebaseAuth.instance;

  factory StudentProgressRepository({FirebaseFirestore? firestore, FirebaseAuth? auth}) =>
      StudentProgressRepository._internal(firestore: firestore, auth: auth);

  StudentProgressRepository.withFirestoreAndAuth(FirebaseFirestore firestore, FirebaseAuth a
      : _db = firestore,
        _auth = auth;

  final FirebaseFirestore _db;
  final FirebaseAuth _auth;

  String get _collection => 'student.progress';

  /// Fetch a StudentProgressModel by the student's uid (document id)
  Future<StudentProgressModel?> fetchProgressByUid(String uid) async {
```

Flutter - ViewModel - Current User

Overview

This is the start of implementing a ViewModel for holding information for the current logged in user in regard to metadata around the following:

- **_user** holds user metadata stored in a UserModel.
- **_classSection** holds class metadata that the user is associated with in a ClassModel.
- **_currentWordLevel** holds current Dolche word level completed as a WordLevel enum value.
- **_wordLevelsCompleted** holds Map of <WordLevel, bool> indicating completed word levels.
- **_wordAttempts** indicates all practice word attempts held in a List<AttemptModel>.
- **logIn** and **logOut** are helper methods to load/unload this ViewModel with information from Firestore.
- **loadStudents** fetches studentIds for _classSection from Firestore.
- **fetchUsersNextPracticeWord** fetches the current user's next practice word in the level.

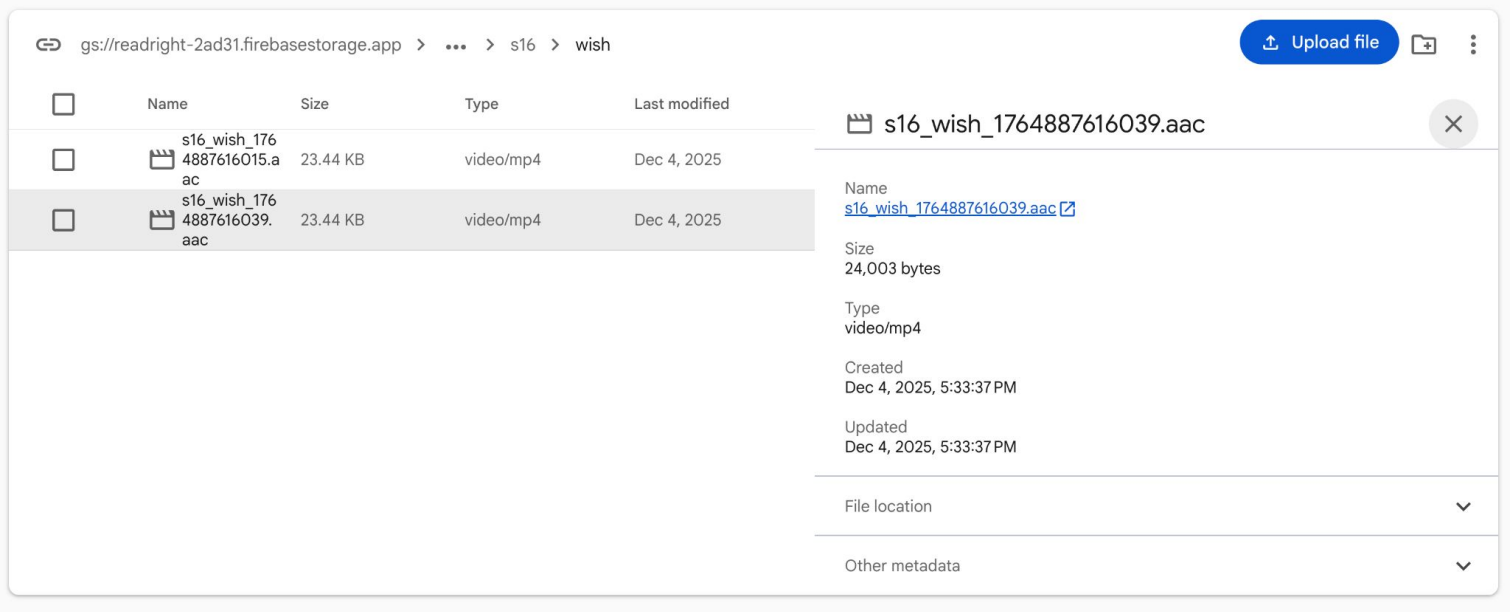
readright > lib > models > current_user_model.dart > CurrentUserModel

```dart
You, 5 days ago | 1 author (You)
class CurrentUserModel extends ChangeNotifier {
  UserModel? _user;      You, 3 weeks ago • feat: Added Provider to handle CurrentUserModel…
UserModel? get user => _user;
  set user(UserModel? user) {
    _user = user;
    notifyListeners();
  }

  ClassModel? _classSection;
  set classSection(ClassModel? classModel) {
    _classSection = classModel;

    // Load students when class section is set
    // You can add logic here to load students related to the class section if needed
    if (_user?.role == UserRole.teacher) {
      loadStudents().then((_) {
        notifyListeners();
      });
    }
  }
  ClassModel? get classSection => _classSection;

  WordLevel? _currentWordLevel;
  WordLevel? get currentWordLevel => _currentWordLevel;
  set currentWordLevel(WordLevel? level) {
    if (_currentWordLevel != level) {
      debugPrint('CurrentUserModel: Changing current word level from ${_currentWordLevel?.name} to ${level?.name}');

      _currentWordLevel = level;

      if (level != null) {
        debugPrint('CurrentUserModel: Current word level set to ${level.name}');
        updateCurrentWordLevel();
      } else {
        debugPrint('CurrentUserModel: Current word level set to null');
      }

      notifyListeners();
    }
  }
```

Firebase Storage - audio retention

Overview

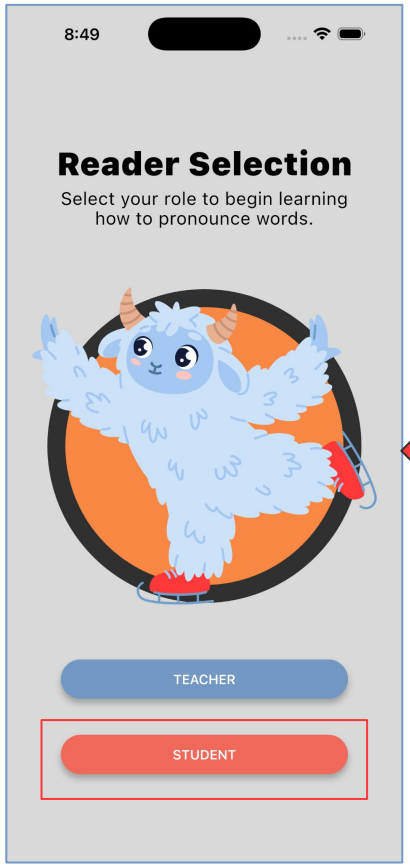
Holds student attempt audio recordings grouped by username then by practice word saved in AAC format.

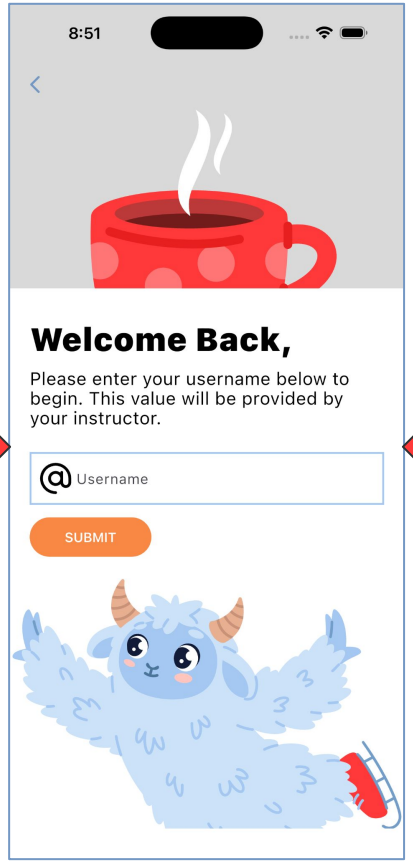


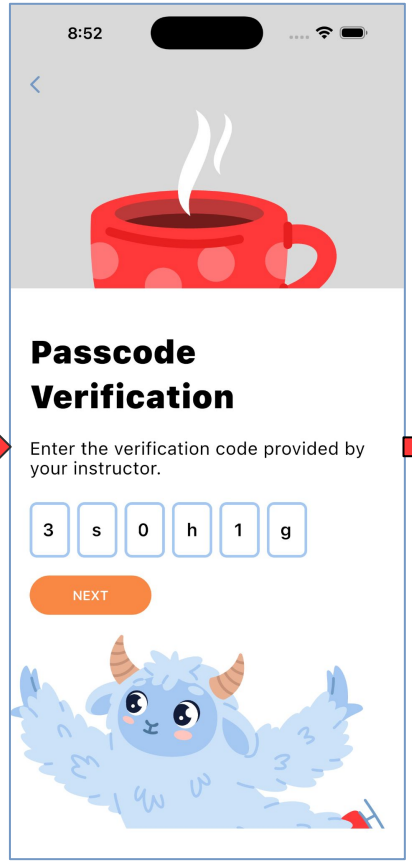
Critical Student Functionality Walkthrough

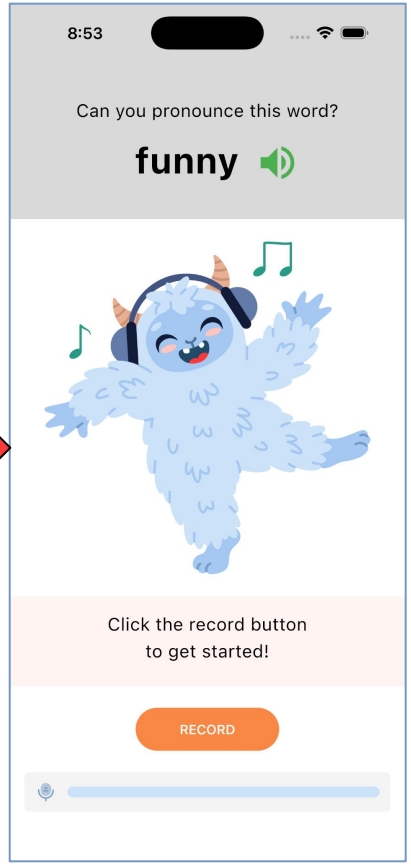
Student Login

Passcode is provided by the instructor (Class Code)

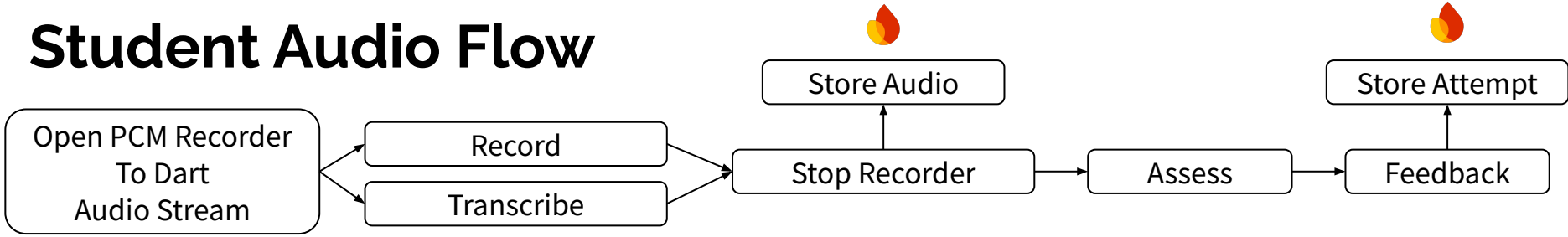


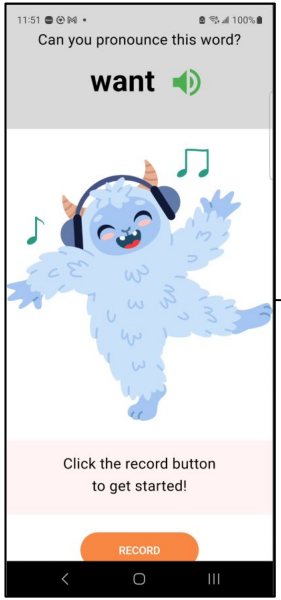


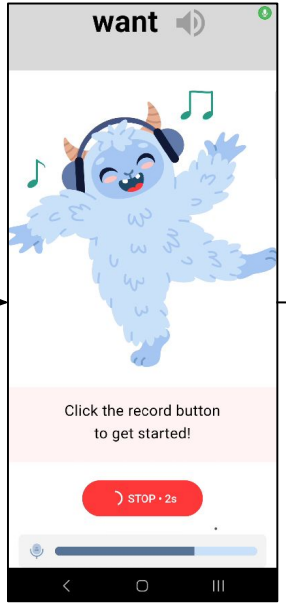




Student Audio Flow







Deepgram

PICOVOICE Cheetah

Audio Pseudo

readright_0.5\readright\lib\audio\stt\pronunciation_assessor.dart

```dart
class AssessmentResult {
  final String recognizedText;
  final double confidence; // 0..1 if available
  final double score; // normalized similarity 0..1
  final Map<String, dynamic>? details;

  AssessmentResult({
    required this.recognizedText,
    required this.confidence,
    required this.score,
    this.details,
  });
}
```

student_word_practice_screen:

- _handleRecord()
  - // check permissions and start the recorder
  - // call _stopRecording after 3 seconds
- _stopRecording()
  - // stop the recorder
  - // convert audio file
  - // POST audio file in Firebase (if audio retention is enabled)
  - // if an internet connection exists:
    - // transcribe recording with Deepgram
  - // else internet connection is disconnected:
    - // transcribe recording with PicoVoice Cheetah
  - // assess with CMUDict and Levenshtein: readright\lib\audio\stt\cloud\deepgram_assessor.dart
  - // return an AssessmentResult
  - // POST attempt result for student in Firebase Firestore attempts, students, and class collections.

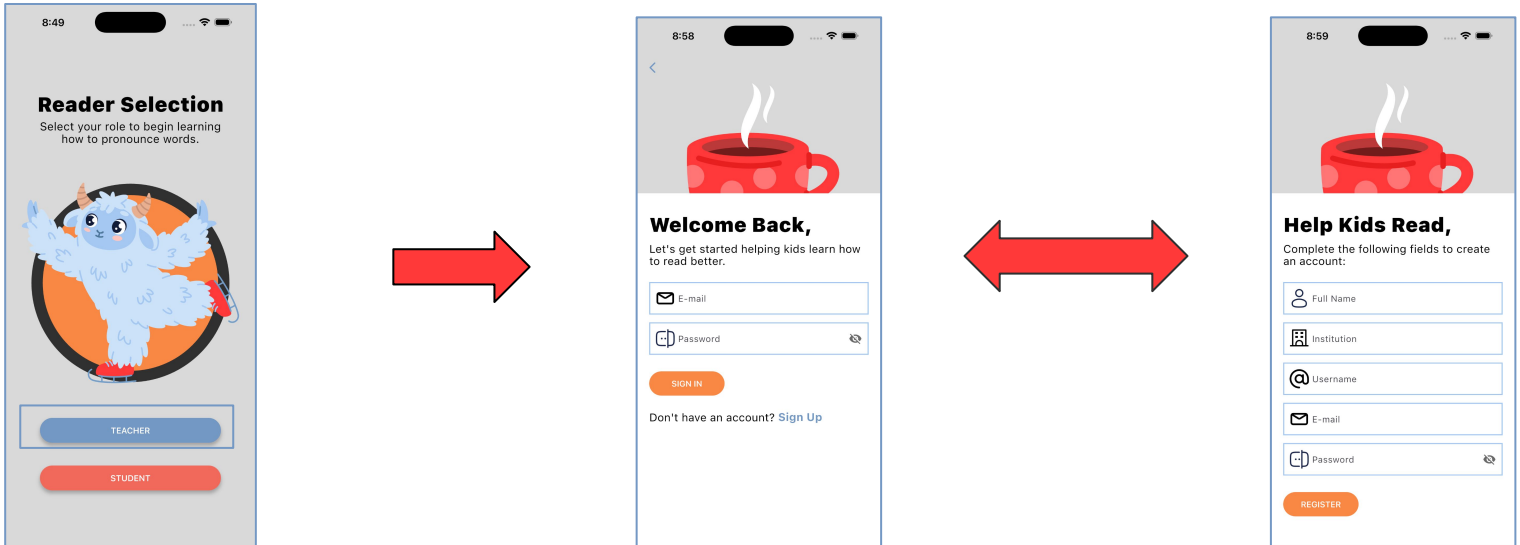
student_word_feedback_screen:

- Interpret and display results of passed AssessmentResult

Critical Teacher Functionality Walkthrough

Teacher Login and Registration

- From the Homepage select the Teacher Button
- Teacher Sign-In
  - Only the email and password are needed
- Teacher Registration:
  - Enter required fields and press the register button
  - A teacher role account gets added to the Firebase users collections and they get assigned to a class which is also created in Firebase under the classes collection

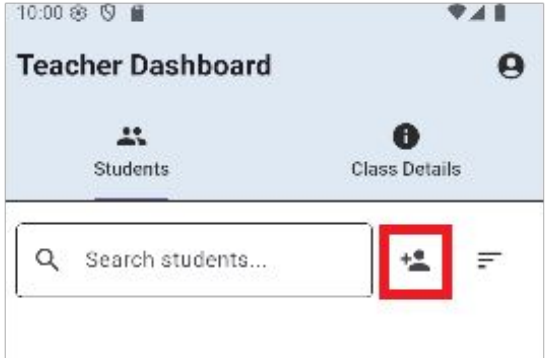


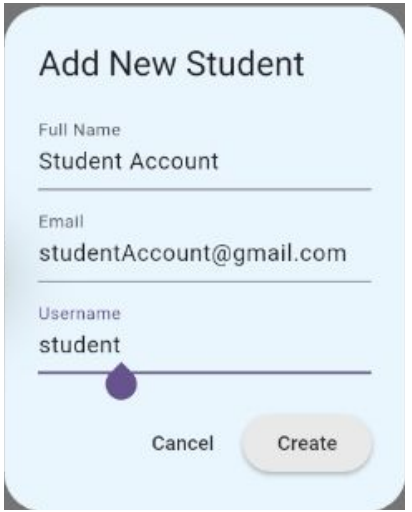
Teacher Functionality

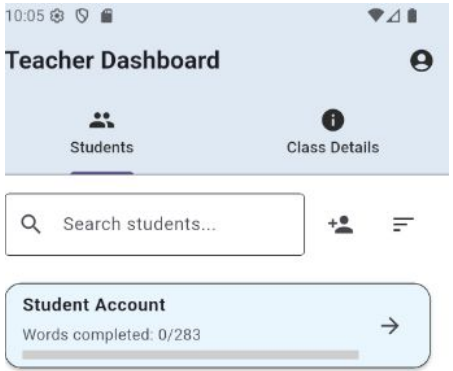
- Teacher Dashboard - Screen that acts as the home base for teachers.
  - Students - Contains a list of students and the ability to register students to the class
  - Class Overview - Contains class progress and additional class options/features such as audio retention, export to .csv & Word Dashboard access
- Student Details - Can be access via the Student Cards on the Teacher Dashboard. Here individual student details/progress can be shown.
  - Student Details - Contains additional student information and progress
  - Word Progress - Contains Student Word Progress as well as the ability to listen to the latest word attempt
- Word Dashboard - Contains a list of words for each grade level, and the example sentences students will be tasked to complete

Adding a Student

- A student can be created on the Teacher Dashboard > Students tabs by tapping the Add student icon
- Fill out the following fields:
  - Full Name
  - Email
  - Username
- Press Create to create a student account
- The student account password will be set to the Class Code
- When a student is added:
  - A new student user gets added to the Firebase users collection
  - An associated Firebase students.progress record gets added
  - New student user gets added to the appropriate classroom under the Firebase classes collection







Exporting Results

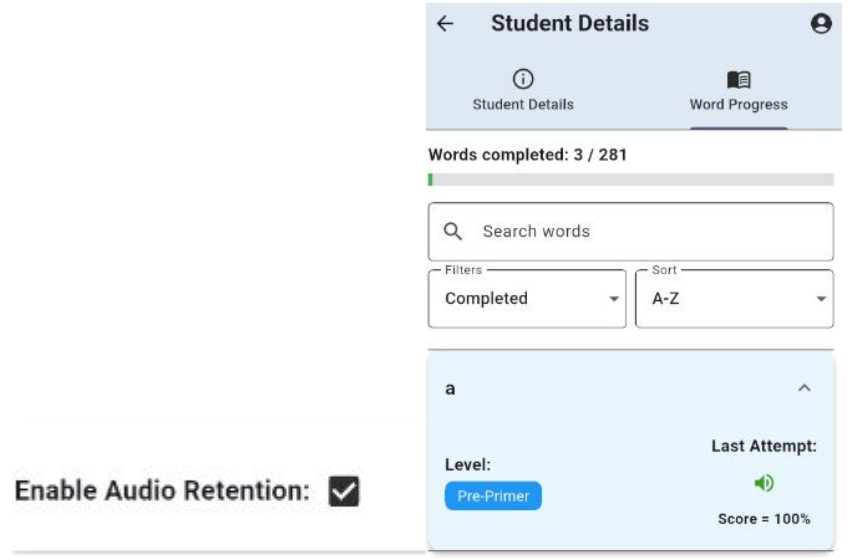
- A teacher can export the class history and progress by using the Export Class Progress to .csv button
- This uses share_plus & the csv packages to create the .csv and share/store the created file
- The data is derived from looking at all the students assigned to the class and looking at each attempt the made. This comes from the Firebase/Firestore databases

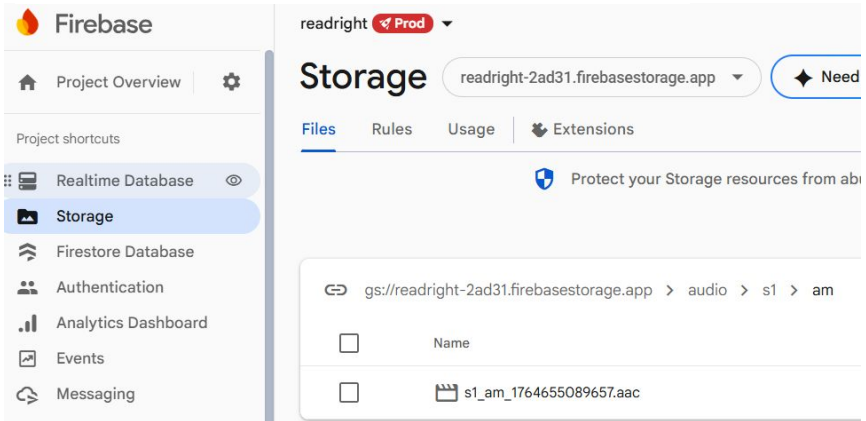
Name	Word	Word Transcript	Accuracy	Correct?	Date
Student 1	wish	dish	66.67	No	12/1/25 19:55:27
Student 1	wish	wrong	0	No	12/1/25 19:58:21
Student 1	wish	wish	100	Yes	12/1/25 19:59:09
Student 1	a	a	100	Yes	12/1/25 20:02:05
Student 1	and	andy	59	No	12/1/25 20:03:17
Student 1	big	biggie	69.64	No	12/1/25 20:04:39

Enabling Audio Retention

Auto Retention:

- Defaulted OFF during class creation
- Can be Toggle through Checkbox located in Class Details > Enable Audio Retention
- When enabled
  - Audio files are stored in Firebase > Storage > [User] > [Word] >
- When disabled
  - Audio files are no longer stored
  - All previous recordings are stored until manually deleted from Firebase
- Latest attempt for a student can be replayed by selecting an attempted word in the Student Details > Word Progress

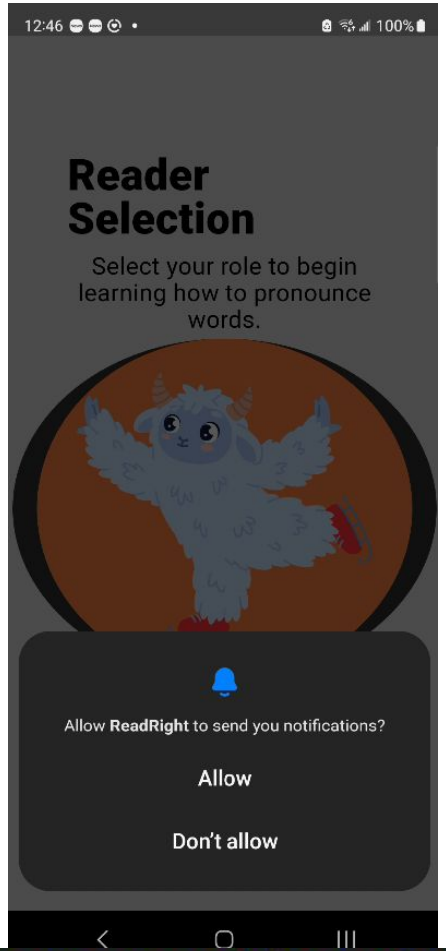




Additional Functionality

Push Notifications

- Firebase allows the functionality to enable push notifications by using the firebase_messaging package
- In order to get the functionality working the following needed to be implemented:
  - Each device has to have notifications allowed
  - A unique Firebase Cloud Token (FCM) for each device must be generated for the device to be registered




```
I/flutter (31636): FCM Token: eVlbQuXJRQCBKYkyfMKBwD:APA91bFOKf82ra6_HGcaxABqCBlNb-PyIYigVHofOR
ty70pegZGNNrNysgeejjgH0dctZoPh0su_U5Kf40zYF3hYI7Yy4V7h0YTMrTOHnIn8b0KlFiVytiY
```

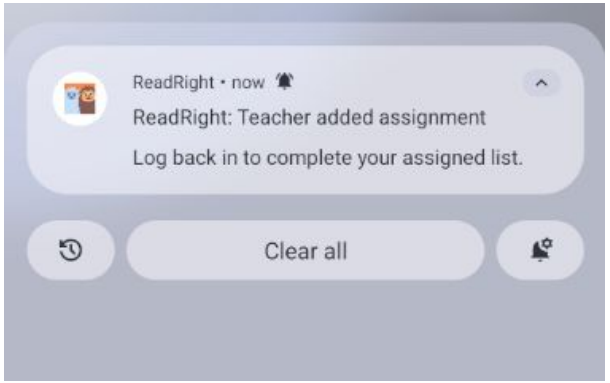
Creating a Push Notification

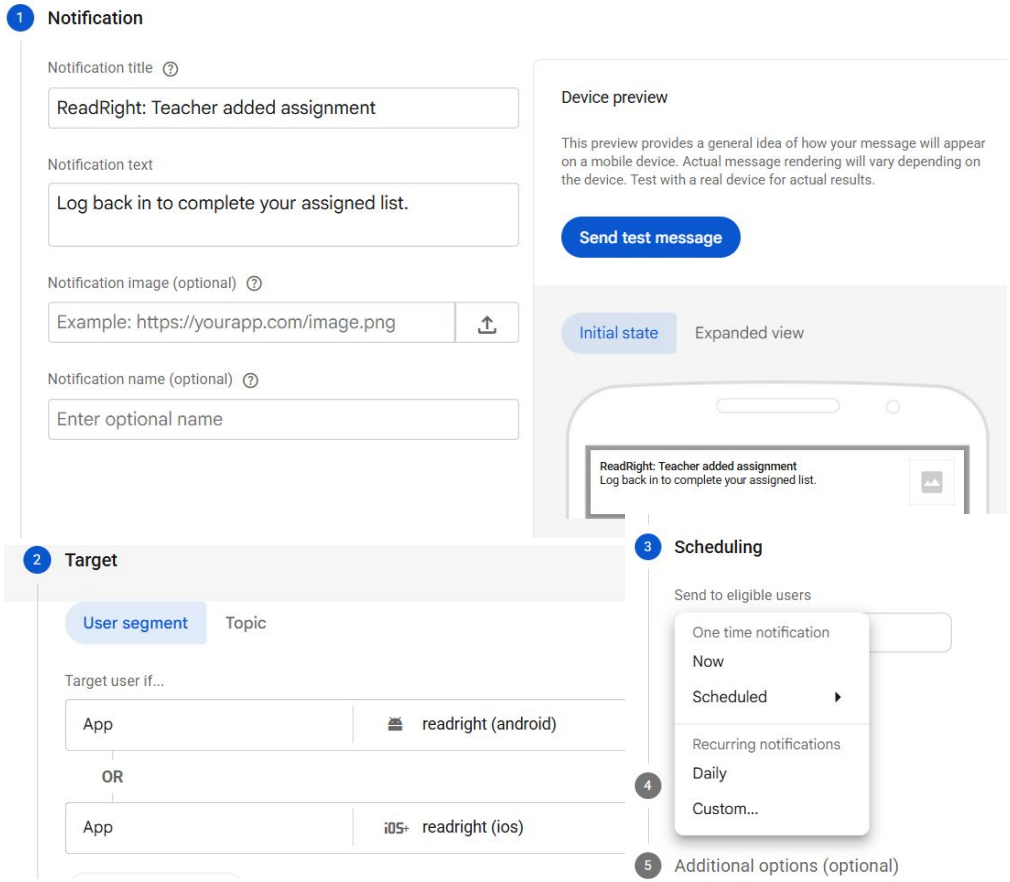
From Firebase > Messages, a push notification can be created by adding the following:

- Title + Text
- Notification Parameters
- Schedule for release

Once these are added, a notification can be pushed to users based on the Notification Parameters.

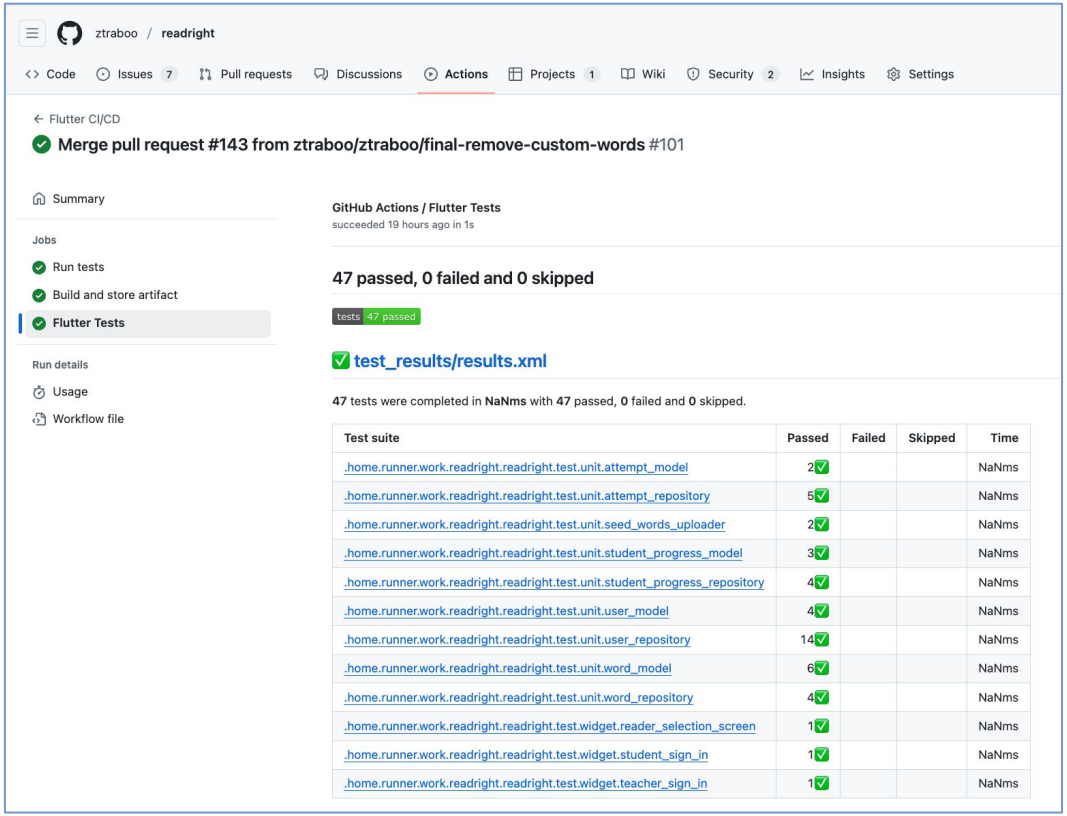
Note Testing does not work on IOS simulator due to needing a Apple Developer ID



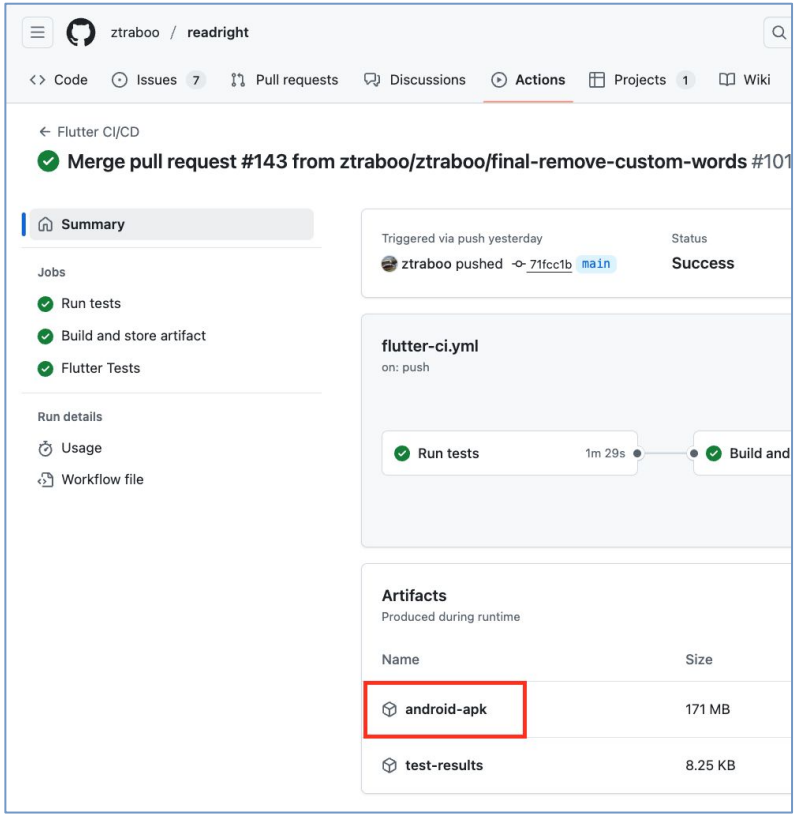


Continuous Integration and Delivery (CI/CD)

Flutter Tests (CI)

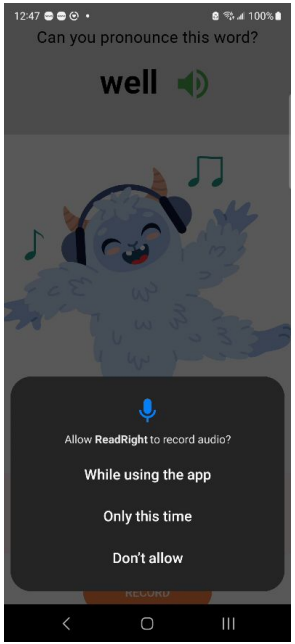


Android APK Build (CD)



Error Handling and Fallback

Microphone Permissions



Network Disconnections

- No error message
- Recognizes disconnection and reverts audio flow to use PicoVoice's Cheetah STT provider over the default Deepgram

Required Dependencies

Cheetah STT:

`cheetah_flutter: ^2.3.0` - https://pub.dev/packages/cheetah_flutter

Deepgram STT:

`http: ^1.1.0` - https://pub.dev/packages/http

`path: ^1.8.3` - https://pub.dev/packages/path

Transcription Assessment:

`string_similarity: ^2.0.0` - https://pub.dev/packages/string_similarity

CSV Export:

`csv: ^6.0.0` - https://pub.dev/packages/csv

`share_plus: ^12.0.0` - https://pub.dev/packages/share_plus

Push Notifications:

`firebase_messaging: ^16.0.4` - https://pub.dev/packages/firebase_messaging

Audio:

`flutter_sound: ^9.30.0` - https://pub.dev/packages/flutter_sound

`path_provider: ^2.1.2` - https://pub.dev/packages/path_provider

`permission_handler: ^12.0.1` - https://pub.dev/packages/permission_handler

`audioplayers: ^6.0.0` - https://pub.dev/packages/firebase_storage

Lessons Learned As A Team

What did your team **learn technically**?

- How to implement frontend and backend support for Mobile Applications - JG
- Structure the app to use a Model-View-ViewModel design pattern for better testability and maintenance - ZT
- Communicate with backend services (e.g. Firebase) - ZT
- Use Dart audio stream to write PCM bytes and then subscribe to stream with multiple instances for creating real-time Speech-To-Text (STT) transcripts and audio codec files - ZT
- How to allow permission status for microphone and notifications - ZT
- How to setup a testing pipeline on GitHub - NC

What was the **hardest part**?

- Learning how to incorporate packages correctly - JG
- Figuring out how to record to a Dart audio stream to allow multiple subscribers to listen for the audio bytes to output Speech-To-Text (STT) transcript file or audio codec file - ZT
- Implementing local assessment providers for transcript accuracy - NC
- Convert audio code from PCM to AAC or WAV formats (used native code for Android and iOS) - ZT
- Restructure the code to implement Model-View-ViewModel (MVVM) architecture after having already implemented a lot of code in an individual screen - ZT

What workflows or habits really **helped** the project?

- Weekly meetings and Task Assignments - JG
- Using Github Project to put in sprint tasking and details to view remaining tasks - ZT
- Prioritization of tasks - NC
- Communicating regularly over Teams - ZT

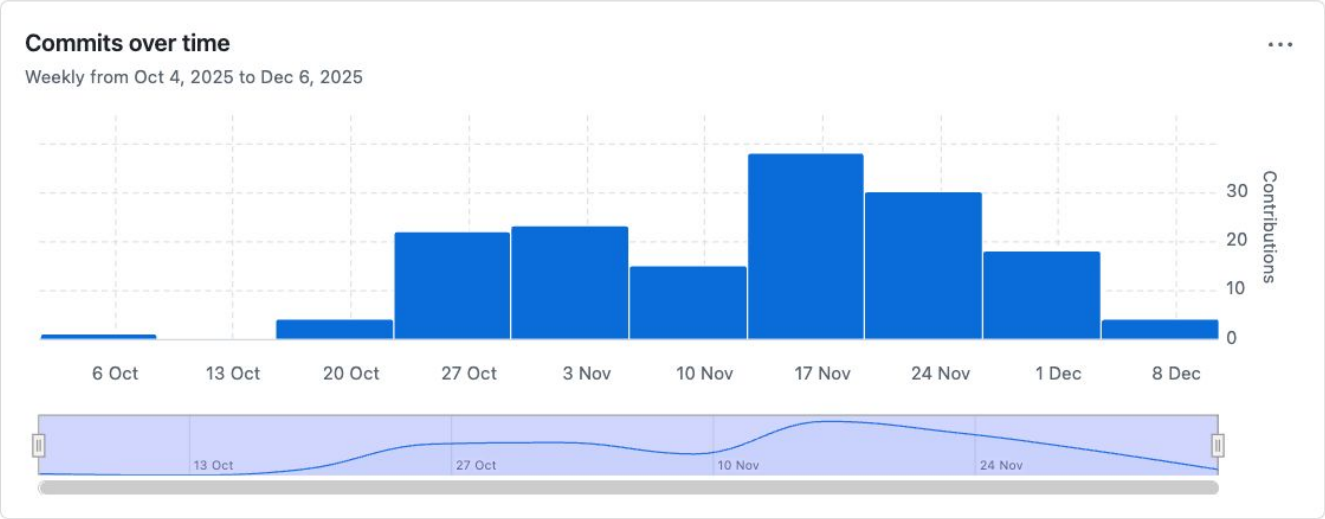
What would your team **do differently** next time?

- Limit scope creep- JG
- Break up the tasking into smaller manageable items - ZT
- Plan to use design pattern Model-View-ViewModel (MVVM) and Provider state management at the start - ZT
- Plan audio and STT providers prior to implementation - NC
- Implement model / services right away and make sure that everyone adheres to this standard through the ViewModel classes - ZT

What are you **proud of** as a team and would absolutely repeat?

- Proud of how well we worked together to push this level of completion through - JG
- Teammates completed their assigned work and check-code changes in often - ZT
- Talked about feature implementation prior to writing any code - ZT
- Identified unrealistic feature requests and found work arounds and/or pushed back - ZT
- Organization and incremental progress - NC

ReadRight - Github Analytics



Source: https://github.com/ztraboo/readright/graphs/contributors?selectedMetric=commits

What features would your team like to add to ReadRight if you had **more time**?

- Add support for multiple classes and additional class support options - JG
- Adding functionality to edit the word list & sentences - JG
- Add in Firebase caching for data to ensure the student flow works in offline-first mode - ZT
- Add in a student word-level dashboard showing progress. This was taken out because the customer thought that it would be too much for the student - ZT
- Use Rive.app to create state-driven animations. Use this for the mascot and star ratings in the student flow initially - ZT
- Add social authentication/registration to the student and teacher login with Firebase Authentication - ZT
- Refactor the code base by implementing Model-View-ViewModel throughout - ZT
- Deploy *release-1.0* of ReadRight app to Apple App and Google Play stores for delivery - ZT