JAPANESE PITCH ACCENT TRAINER

DESIGN DOCUMENT

HAFZA ABDULLAHI
4TH YEAR SOFTWARE DEV STUDENT
SUBMISSION DATE: 01/12/2025

CONTENTS

SYSTEM ARCHITECTURE OVERVIEW

The Japanese Pitch Accent trainer is built using Flutter and Python. Comprising of both a mobile app and web.

The system is made up of three primary layers

- **Presentation Layer (Client)**: A Flutter based application handling the user interaction, audio recording, and visual feedback display.
- **Application Layer (Server)**: A Python Flask API hosted on Render, responsible for pitch extraction and graph generation using Praat-based algorithms.
- **Data Layer**: Firebase for authentication and cloud storage, combined with local asset management for Anki deck integration.

PRESENTATION LAYER (CLIENT)

Responsible for

- User authentication
- Audio recording and playback
- Flashcard rendering
- Graph visualization from backend responses

Flutter compiles to both **Web** and **Android**, a single codebase maintains consistent behaviour across platforms.

APPLICATION LAYER (PYTHON + RENDER)

This layer performs all digital Signal Processing tasks that would be too heavy for mobile browsers on Render.

Render[1] is a modern cloud hosting platform that lets you deploy web apps, APIs, databases, and background workers **without managing your own servers.** The python render server handles:

- Pitch extraction
- Contour smoothing
- Graph generation

Parselmouth (Praat for Python) allows industry standard acoustic analysis without implementing raw DSP manually.

Why is Render used?

This project uses **Parselmouth, SciPy, Matplotlib**, and other low-level audio libraries. These require:

- system-level dependencies (like libsndfile1)
- Linux audio libraries
- a controlled environment

Render supports Docker, meaning you can ship your entire backend (all Python + C++ library requirements) inside a pre-built container.
This guarantees it runs the same way on the cloud as it does on your local machine.

Other platforms (like Firebase Hosting or Vercel) **cannot run Parselmouth** because they don't support such heavy DSP libraries. So Render was the best option

DATA LAYER (FIREBASE + LOCAL JSON ASSETS)

The project integrates two data sources:

- **Firebase Authentication**: Manages secure user sessions.
- **Local JSON Decks**: Pre-processed Anki flashcards containing vocabulary, readings, example sentences, and audio assets.

ANKI CONNECT API

To integrate vocabulary from the a deck into flutter, the project uses **AnkiConnect[2]**, a local REST API plugin that exposes the user's Anki data (notes, fields, and media files including audio) through HTTP requests. This retrieves the card info from a premade flashcard deck "***Kaishi 1.5k - Basic Japanese Vocabulary[3]***"

Since Flutter Web and mobile apps **cannot directly read .apkg deck files** (due to SQLite parsing limitations and browser sandboxes), I used preprocessing script in Dart to extract all relevant flashcard information **before packaging the app**, later this will be reworked so the deck is uploaded to the firebase database.

USER INTERFACE ARCHITECTURE

WIDGET HIERARCHY

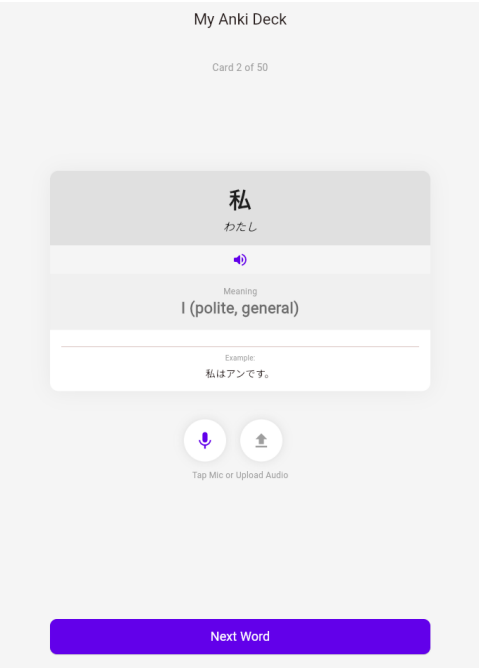
The UI is decomposed into modular, testable widgets, each following the Single Responsibility Principle:

- **WelcomeScreen**
  Determines navigation flow based on authentication state.
- **LoginScreen**
  Implements Google Sign-In using FirebaseAuth.
- **HomeScreen**
  Displays flashcards, handles recording, transmits audio to the backend, and renders pitch graphs.
- **WordCard Widget**
  Presents:
    - Kanji
    - Furigana
    - English meaning
    - Example sentence
    - Native speaker audio button

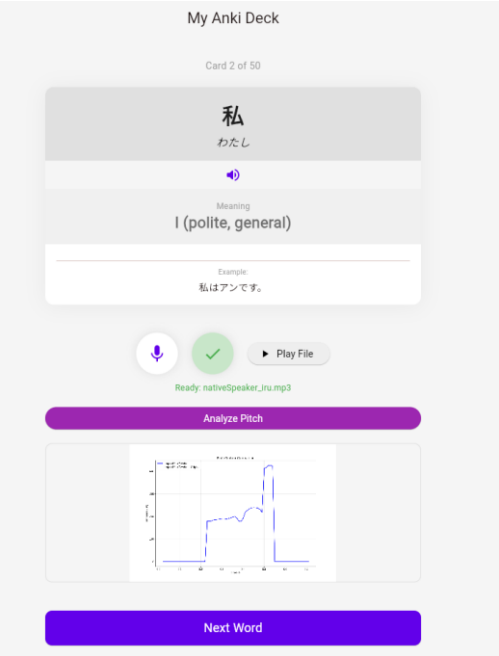
This mirrors standard Japanese learning tools (e.g., Anki), maintaining user familiarity.

UI SCREENSHOTS

App flashcard format as presented to the user before audio has been uploaded



App flashcard after using has uploaded audio from a file and pressed analyse pitch.



REFERENCES

LINKS

- **[1] Render** https://render.com/

- **[2] Anki connect**, https://ankiweb.net/shared/info/2055492159

- **[3] current anki deck used by project**
  https://ankiweb.net/shared/info/1196762551