| MAKERERE  P.O Box 7062  Kampala, Uganda  [info@cis.mak.ac.ug](mailto:info@cis.mak.ac.ug)  <https://cis.ac.ug> |  | UNIVERSITY  +256 41 540 628 |
| --- | --- | --- |

**COLLEGE OF COMPUTING AND INFORMATION SCIENCES**

Eliza- A virtual assistant for the Ubuntu platform

By

Group 38

DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF COMPUTING AND INFORMATICS TECHNOLOGY

A Project Report Submitted to the School of Computing and Informatics Technology

For the Study Leading to a Project Report in Partial Fulfillment of the

Requirements for the Award of the Degree of Bachelor of Science in Computer Science

Of Makerere University

Supervisor

Ms. Rose Nakibuule

School of Computing and Informatics Technology, Makerere University

rnakibuule@cis.mak.ac.ug, +256-414-540-628

November, 2018

GROUP MEMBERSHIP

| SNo | Names | Registration Number | Signature |
| --- | --- | --- | --- |
| 1 | Lincoln Karuhanga | 16/U/5465/PS |  |

# Table of Contents

[Table of Contents](#_22s39mdzz4ar) **3**

[Definitions](#_3icqq0p8sj52) **6**

[Virtual Assistants](#_dpzsqu3bx86i) 6

[Acoustic Model](#_uxwz2svyy0dn) 6

[Language Model](#_nhpt5f7kr6ks) 6

[**Executive Summary**](#_vvlxr92nl331) **6**

[Introduction](#_3ua4z47revfl) **8**

[Background](#_q3xd8syn8jul) 8

[Problem Statement](#_hhhgz5r24t3k) 8

[Objectives](#_n49c18hkkrwd) 9

[Main Objective](#_14o7s1pr46kh) 9

[Specific Objectives](#_cn8tgz5pqnny) 9

[Additional Objectives](#_g9x821s66y1m) 9

[Scope](#_n1irogvcdlrb) 9

[Operating System](#_1j1evrq7dzai) 9

[Primary Scope](#_m7hvsvh5jv8c) 9

[Secondary Scope](#_mwcumx8gp6cs) 9

[Languages](#_gfeq6nujvqvi) 10

[Primary Scope](#_d74pf18gp8oz) 10

[Secondary Scope](#_p312yhmgybpl) 10

[Supported Applications](#_qqwbywtcuvp4) 10

[Primary Scope](#_y0r2jq6wbuk7) 10

[Secondary Scope](#_68gwzk5wc2on) 10

[Method of control](#_nepvx9o9ari) 10

[Primary scope](#_f780kbsh4u99) 10

[Secondary scope](#_nawjveoc3de3) 10

[Significance](#_3we2ygaa5bm1) 10

[Literature Review](#_n4zrk4a367b) **11**

[Approaches to Speech Recognition on Embedded Systems](#_fcnp3xx9cnw) 11

[Previous projects](#_pa6wpvuskbu4) 12

[Methodology](#_65q247tf098d) **13**

[Understand speech recognition impediments for offline systems and find appropriate solutions to each](#_3emoch7dilep) 13

[Phoneme recognition based approach](#_8d6covlm7g3f) 13

[Model generation from training data](#_w654v6quchyq) 13

[Develop automated menu access methods and approaches](#_9uoourl2kxlt) 14

[Efficient action resolution and execution based on evaluated input](#_h73lkbsb0svu) 14

[Data Collection](#_ispupx7digyp) 15

[Using Publicly Available Data Sets](#_rb4xxgqr6mgh) 15

[Data Collection: Common Voice Uganda](#_rfzvi6e275wp) 15

[Speech Recognition](#_d6zl2zzglxfy) 16

[Models](#_lye0sz5gu5da) 16

[Approach](#_kkbazwkldsum) 16

[**System Study, Analysis and Design**](#_sikcjuwona3p) **17**

[Data Collection: Common Voice Uganda](#_7z9k32ngn5dl) 17

[Eliza](#_rrns3aw15nrd) 18

[**Presentation of Results**](#_6mlai4rnpt86) **19**

[Data Collection: Common Voice Uganda](#_93bwrk5leoew) 19

[Eliza](#_jsrglicbuu2a) 21

[**Limitations, Recommendations and Conclusion**](#_9a9puf7zr4m7) **22**

[Limitations](#_vxlj7rimxrl1) 22

[Training Data](#_9tdp7o7slsrh) 22

[Substandard Recognition Models](#_v1awi2euwjfs) 22

[Evolving Research into Speech Recognition](#_bu2t5oim7fa) 22

[Dynamic Computer Environments](#_sgjymfnubjom) 22

[Recommendations](#_xbjqhx68dbjb) 23

[Data Collection](#_jv80w6hboj8b) 23

[Action Resolution](#_ohpxayvv8pjp) 23

[References](#_z2xlqlr6x54a) **24**

[Appendix](#_rcbf77oo2b3) **25**

[Open Source](#_whjhngma1uil) 25

[Data Sets Used By The Project](#_4jaacxvbgcz) 25

[Project Dependencies[14]](#_nnws9zfzroyj) 25

[Project tools](#_9mx3crgvcp9o) 25

[Github Repositories](#_s7zwm4hr3zom) 25

[External Tools](#_ytpyabpips8l) 26

[Roadmap](#_4p5429lhmvg3) 26

[Phase 1](#_31oyi5vidq0q) 26

[Phase 2(In progress)](#_t87gpdi3zo8i) 27

# 

# Definitions

## Virtual Assistants

A virtual assistant or intelligent personal assistant is a software agent that can perform tasks or services for an individual.

## Acoustic Model

An acoustic model is used in automatic speech recognition to represent the relationship between an audio signal and the phonemes or other linguistic units that make up speech. The model is learned from a set of audio recordings and their corresponding transcripts. It is created by taking audio recordings of speech, and their text transcriptions, and using software to create statistical representations of the sounds that make up each word.

## Language Model

A statistical language model is a probability distribution over sequences of words that represents the probability that that particular sequence of words is likely to be the target transcription. The language model provides context to distinguish between words and phrases that sound similar.

# Executive Summary

This project set out to build an intelligent virtual assistant for the Ubuntu Platform. As stated in the objectives, we looked to;

1. Recognize speech(Speech to Text)
2. Understand what was said(Natural Language Processing)
3. Perform Actions Based on these commands

The bulk of this project was dominated by the first activity, given it was the starting point for the others, resulting in a spin off project, common voice Uganda which seeks to collect more of locale specific data to make the training more effective. The second objective was toned down and the project took a more action focussed approach.

The result is a working application, that is still not ready for mass consumption, but shows clear progress towards the overarching goal.

# 

# Introduction

## Background

Technology exists to make our lives easier and to this end, several tools are constantly being developed and experimented with. A key part of this ecosystem is variety. Variety gives consumers a chance to decide what works for them out of the range of available products. After decades of dominance on the personal computer scene, the Windows Operating System is increasingly seeing more competition from the previously little known Ubuntu Operating System, a Linux Operating System distribution.

A key factor in this growth has been the increasing availability of matching and often better and more open, decentralized tools that offer functionality at par with that offered by the Microsoft Operating System.

An increasingly useful tool on our smart devices is the Virtual Assistant which the Ubuntu Operating System conspicuously lacks.

The purpose of the project is therefore to start work on an open-source, community driven, open and transparent virtual assistant primarily targeting the Ubuntu Operating System, with the goal of further adding value to the platform and offering consumers variety.

## Problem Statement

Users of the increasingly popular Ubuntu Operating System find that the platform sometimes lacks convenient tools common with other, often proprietary Operating Systems. A key tool missing on the platform is an officially sanctioned Virtual Assistant, a version of which all competing proprietary operating systems have. It is important that an open source alternative be available for a tool that is gradually playing a bigger role in how people interact with their devices, to allow transparency, inclusion, variety and collaboration in the design decisions and data collected by such a tool.

## Objectives

### Main Objective

To develop a proper structure for and build a Virtual Assistant Primarily targeting the Ubuntu Operating System.

### Specific Objectives

* Efficient action resolution and execution based on evaluated input.
* Develop automated menu access methods and approaches generally applicable to all applications in the Gnome desktop environment.
* Understand speech recognition impediments for offline systems and find appropriate solutions to each, ultimately implementing a scalable approach.

### Additional Objectives

* Structure project in a way that allows multiple contributors to gradually add multiple application support
* Understand why previous projects have not achieved widespread use and remedy these issues

## Scope

### Operating System

#### Primary Scope

Ubuntu

#### Secondary Scope

Based on the success and applicability of the implementation approach chosen, this may expand to include other Linux Distributions.

### Languages

#### Primary Scope

English

#### Secondary Scope

Based on the approach to speech recognition chosen, other languages will subsequently be added, based on the community’s decision as well as ease of application of the knowledge and methods developed for English.

### Supported Applications

#### Primary Scope

Common system tasks

Popular Utility Applications

#### Secondary Scope

Additional applications will be added on demand and community contribution basis

### Method of control

#### Primary scope

Command based action triggers

#### Secondary scope

Natural language processing based techniques to action resolution and trigger

## Significance

This project explores existing research into natural language processing and speech recognition on embedded and primarily offline systems and attempt to consolidate and layout the best performing strategies currently available. Most speech recognition and language processing approaches are currently dependent on online networks of connected processing power that renders using the same approaches on small devices with limited computing power infeasible.

Exploration and publication of the possible performance benefits of using a restricted language or search set on the best performing and widely available speech recognition approaches.

Build for community contributions- the project will be structured in a modular manner, increasing complexity with time, in order to allow for significant contribution from the Ubuntu open source community, as well as constant evaluation of the chosen methods to ensure parity with the best methods currently available.

Phoneme dictionary generation and update. One of the approaches to speech recognition taken will require work with language phoneme dictionaries. Efficient generation of correct and dependable phonemes to required word sets will also be a major area of exploration and findings will be promptly published.

Evaluating and documenting performance, adaptability, ease of use and customization of present speech recognition tools and libraries, including but not necessarily limited to kaldi and pocket-sphinx.

# Literature Review

## Approaches to Speech Recognition on Embedded Systems

There are several approaches that have been experimented with when doing speech recognition under these conditions. The most promising though are Phoneme based approaches and Data collection and model training based approaches.

The most commonly used speech recognition engines currently use neural networks and one of the goals of the project will be to assess the feasibility of using this approach on an offline system, and whether enough data will be present to allow this approach.

The project takes an in depth look into two of the most commonly used open source speech recognition toolkits- CMU-sphinx and Kaldi. CMU-sphinx is a continuous-speech, speaker-independent recognition system making use of hidden Markov acoustic models (HMMs) and an n-gram statistical language model.

More recently, progress has been made with offline speech recognition at better than real-time performance levels to warrant to make the approach feasible for general application. In particular the a paper by Ian McGraw et all[1] demonstrates this progress, having achieved a 13.5% word error rate using a quantized Long Short-Term Memory (LSTM) acoustic model trained with connectionist temporal classification (CTC) to directly predict phoneme targets. Comparatively, google’s speech recognition engine has an error rate of 8% according to result from tests by Veton Këpuska and Gamal Bohouta at the Florida Institute of Technology[2].

## Previous projects

There have been attempts at implementing such a project before, each of which has failed to achieve widespread adoption. The most promising of these were;

* Gnome voice control[3]
* Simon[4]
* Jasper[5]
* Speech control[6]

None of these projects has achieved widespread use for multiple reasons, the most common being the limited capability for speech recognition from inaccuracy in speech recognition, taking a generic approach to application control, attempting to tackle several different issues concurrently, especially in regards to attempting the generic control of multiple applications in a similar manner, and a distinct lack of focus on a single platform.

# Methodology

## Understand speech recognition impediments for offline systems and find appropriate solutions to each

The primary data collection tool will be the ODK toolkit. Details of the server and application setup access are in the appendix. The project will also make use of publicly available datasets for use while training the models specifically the Mozilla Common Voice Dataset[11].

## Phoneme recognition based approach

The phoneme based approach takes a piece of sound, cleans it to remove unnecessary parts of it, for example, background noise as well as doing further preprocessing, like normalizing amplitude, before translating the recognizable speech into a sequence of phonemes which is subsequently compared to a previously built model based on a properly defined phoneme dictionary.

An implementation using this approach will have to find a scalable way of generating phonemes that accommodate the widely different accents and speech of different users, or adapt the dictionary to a particular user. Large scale phoneme dictionary generation is often done by experts and this will involve consulting one on the best approach to follow with generating this dictionary in an efficient scalable manner. This project seeks to build on top of progress made by the Kaldi speech recognition toolkit as well as the CMU-Sphinx toolkit- in particular pocket-sphinx from Carnegie Mellon University.

## Model generation from training data

The second approach is to simply collect speech data from multiple people, for example mentioning a set of words, cleaning the data by eliminating attributes of the data that will not be necessary for the model generation process, like volume levels, for example by minimizing background noise, normalizing amplitude and scaling the length of the sound and properly labeling each piece of input. Part of this data is then fed into algorithms that generate a model based on this data. The generated model is then tested with the remainder of the data to ensure its performance, recall and accuracy is within expected limits.

The approach used in the initial prototype experimented with speech data from a single user and used algorithms, out of the box from the sklearn toolkit to perform this training and validation. The initial results indicate an improving accuracy rate, but more data and better data cleaning, and a better understanding of the sound data is required to improve the model’s accuracy, as well as the need to use cross validation during training. These changes will determine suitability of these algorithms for this task, in particular, the Support Vector Machine, Decision Tree and Naive Bayes based classifiers.

## Develop automated menu access methods and approaches

The latest versions of the Ubuntu operating system use the Gnome desktop environment, which had previously been abandoned for the Unity desktop environment. The challenge here is most applications do not follow conventions in development, making accessing and manually parsing application menus in a single manner for all applications infeasible.

The current approach is to register specific actions for each application that will then map to a trigger word. This makes it easy to scale the project horizontally by adding more applications, while at the same time vertically, by adding functionality and actions to applications already added to the project.

## Efficient action resolution and execution based on evaluated input

Related to the previously mentioned approach, executing actions is also impeded by the same hurdles that complicate menu access.

The current approach is to trigger actions via key shortcuts for example triggering ‘ctrl+s’ when a save action is requested. This fits well into the scaling approach and is application specific, which handles the issue of these actions and commands being unique for each application.

Another approach to action resolution will be passive search and information structure build up. This will allow the application to return context specific results, for example, requesting a music track to be played will trigger a search in only areas known to have music.

## Data Collection

During the course of the project, multiple approaches were considered and experimented with in order to handle data collection.

### Using Publicly Available Data Sets

The first approach was to use publicly available data sets. In particular data sets from sources like the Mozilla Common Voice project were used and experimented with. A list of all open source data sets is available in the [appendix](#_4jaacxvbgcz). A key point and downside of these datasets is even though they try to access data from as diverse a source as possible they still inherently have a focus on a particular demographic of people, particularly Americans and Europeans, which is inherent in the internet usage numbers in these areas as compared to others.

### Data Collection: Common Voice Uganda

[*https://common-voice-uganda.github.io/frontend-web/dist/#/*](https://common-voice-uganda.github.io/frontend-web/dist/#/)

I found the Mozilla Open Voice Project an intriguing approach, and the idea was to replicate this with a focus on Uganda, expanding to East Africa. This led to a mini spin off open source project, which has unfortunately not gathered much steam yet. This platform focuses on collection and validation of this data, with the goal of building a rich and quality dataset, which will be used for this project, and also shared with the main Common Voice Project.﻿﻿

This project replicated the Mozilla Common Voice Model, including using their API to source sentences used for the recording and validation.

## Speech Recognition

### Models

This project experimented with multiple speech recognition models and/ or engines in order to strike a balance among speed, ease of setup, accuracy and compute intensity. The project experimented with two other offline models, CMUSphinx- dropped due to terrible accuracy and Kaldi- difficult setup, especially for multiple client machines before settling with Mozilla’s Deep Speech and the SnowBoy speech detection engine, whose attributes were acceptable on all fronts.

Of particular note is Snowboy’s approach to the problem, which is a hotword detection engine in particular. This means it depends on a personal(must have been pre generated by the individual) or a universal model, built through crowdsourcing keyword speech data and is particularly helpful as a trigger system.

| Model | Accuracy(WER)[2] | Ease of setup(1-3 being easiest) | Pricing |
| --- | --- | --- | --- |
| Google Cloud Speech Recognition | 8% | 3(API Based) | 0.006$/ 15s |
| IBM Watson Speech to Text | 24% | 3(API Based) | 0.02$/ 60s |
| Mozilla DeepSpeech | 16% | 2 | Open Source |
| Amazon Transcribe | 21% | 3(API Based) | 0.0004$/ 1s |
| Kaldi | 14% | 1 | Open Source |
| CMU Sphinx | 26% | 3 | Open Source |

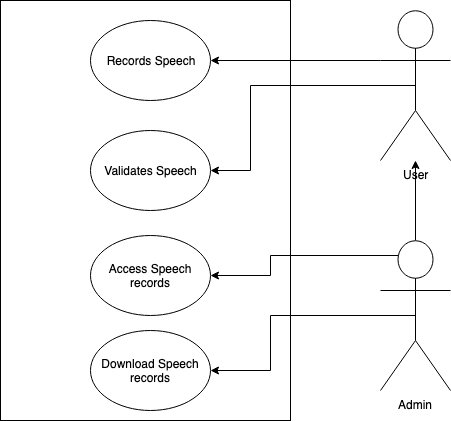
**Comparing the Top Speech Recognition Engines**

### Approach

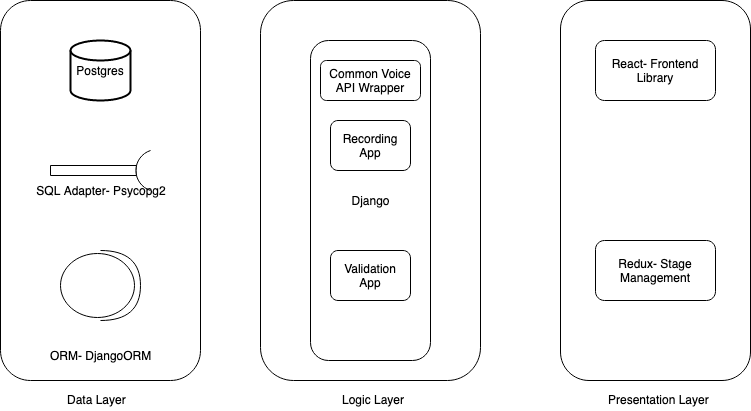
One of the approaches taken during speech recognition is a multi level recognition approach, in order to avoid issues with context. Particularly, when the software is run, it **waits for a particular wake word**, after which it enters a **restricted triggers** phase, during which is only accepts particular keywords for example, pause, save, search.Depending on the subsequent trigger, we then listen for speech that would apply to the particular context and then proceed to perform that action.

# System Study, Analysis and Design

## Data Collection: Common Voice Uganda

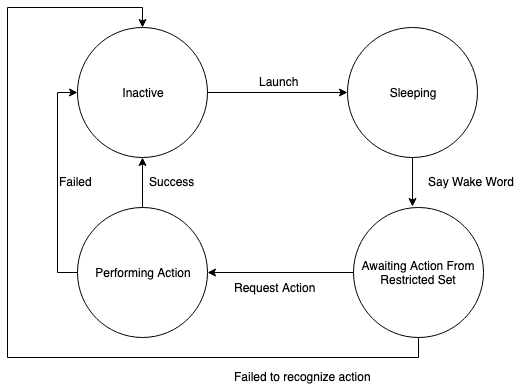
****

**Use Case Diagram**

****

**Component Diagram**

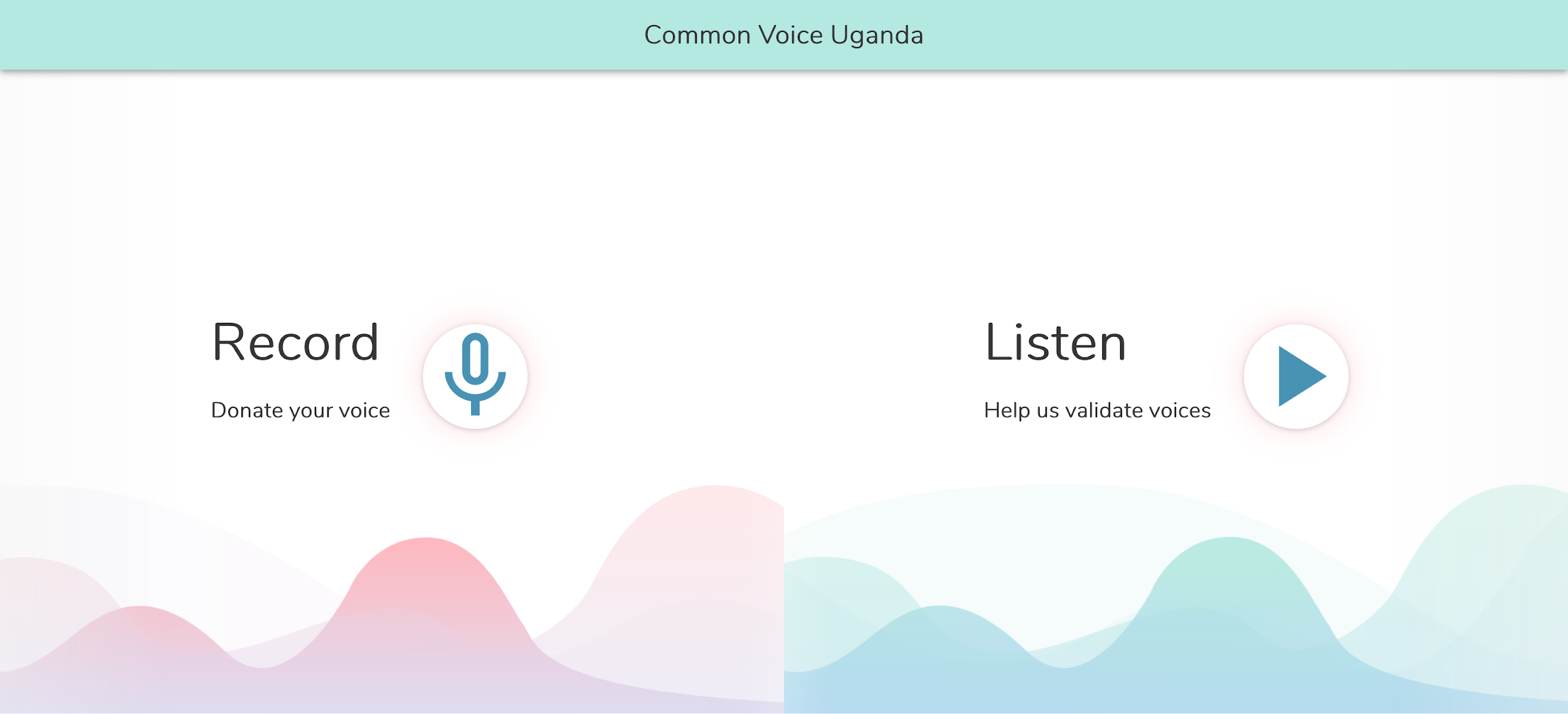
## Eliza



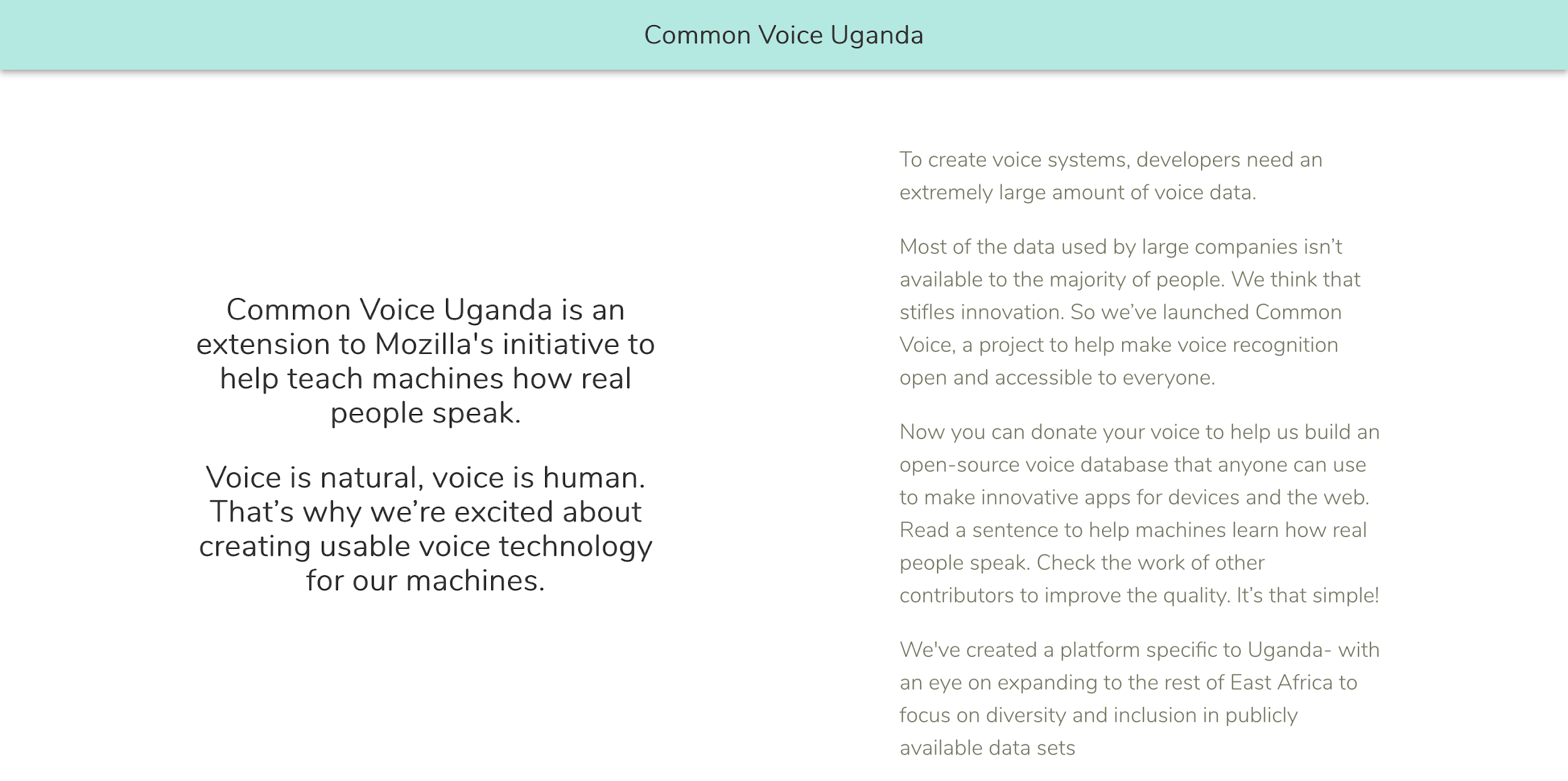
**State Diagram For Application**

# Presentation of Results

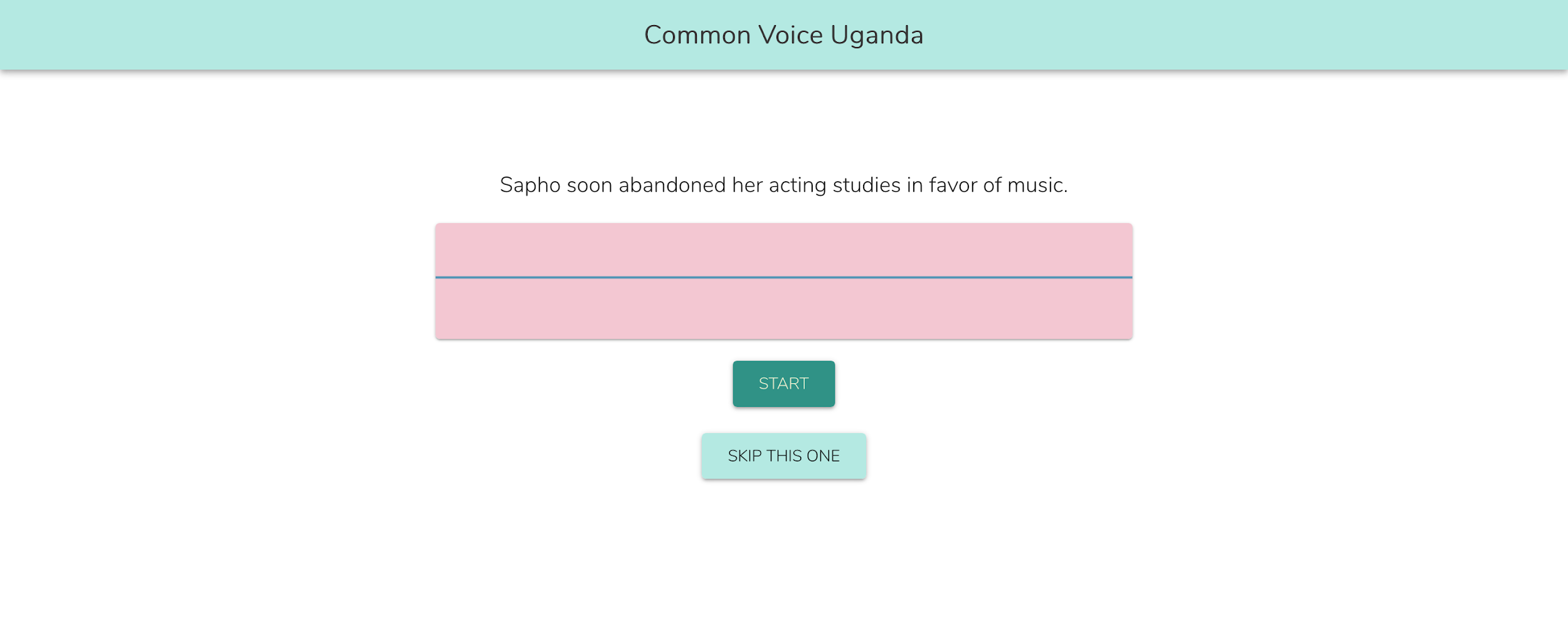
## Data Collection: Common Voice Uganda



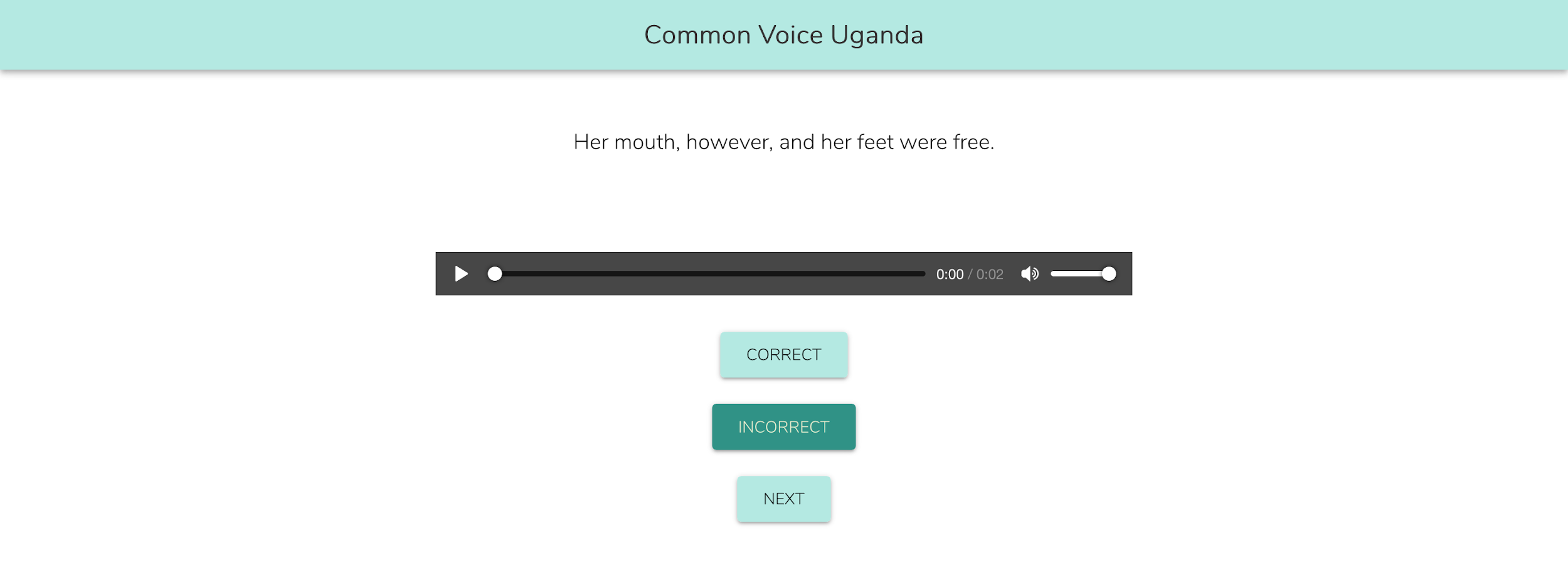
**Common Voice Uganda Home Screen**



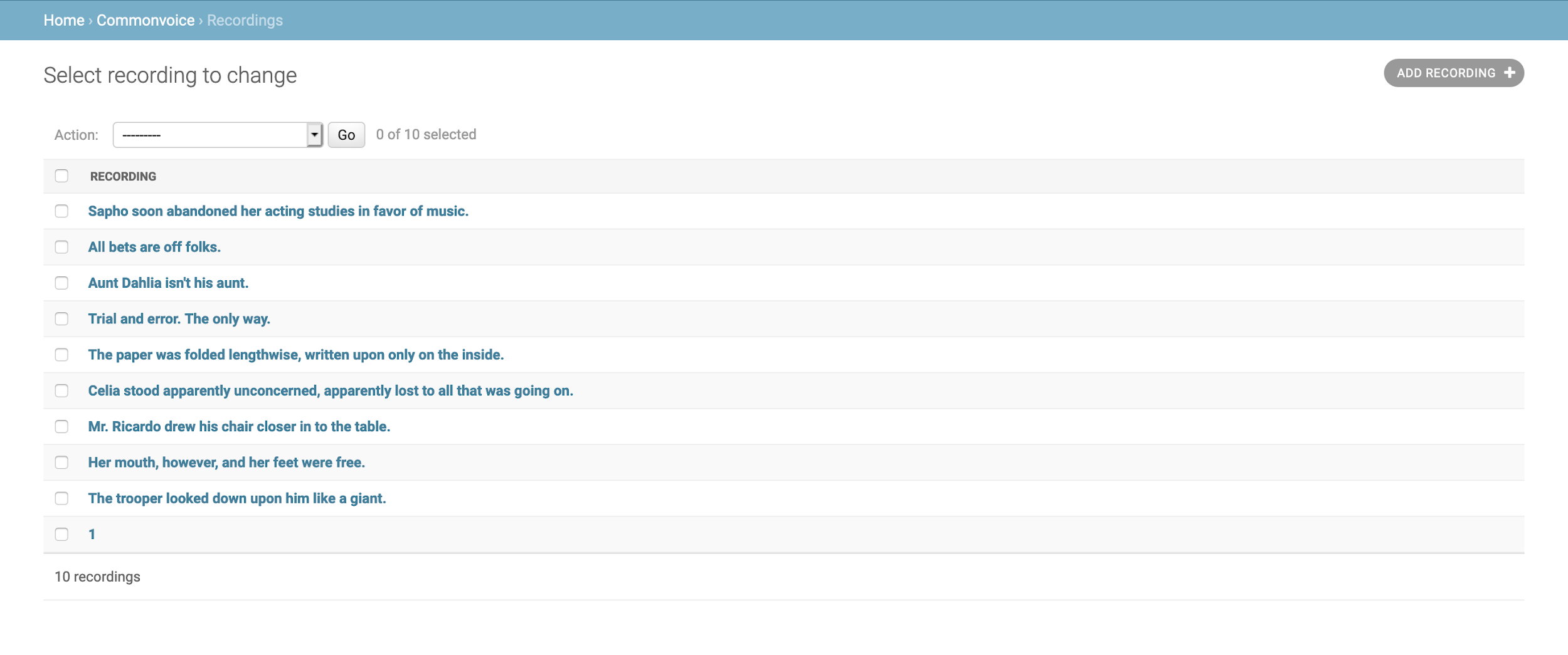
**Common Voice Uganda Home Screen**



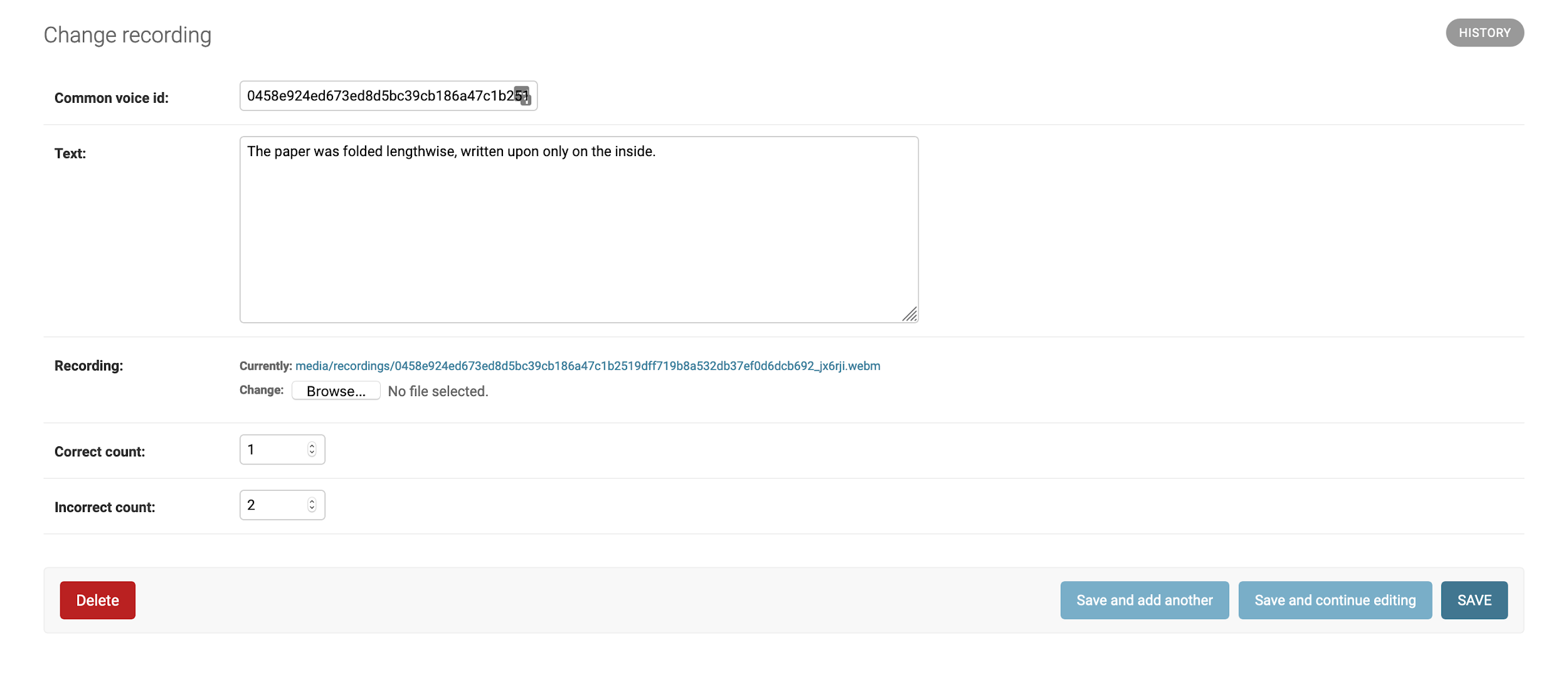
**Common Voice Uganda Speech Recording Collection Screen**



**Common Voice Uganda Speech Validation Screen**

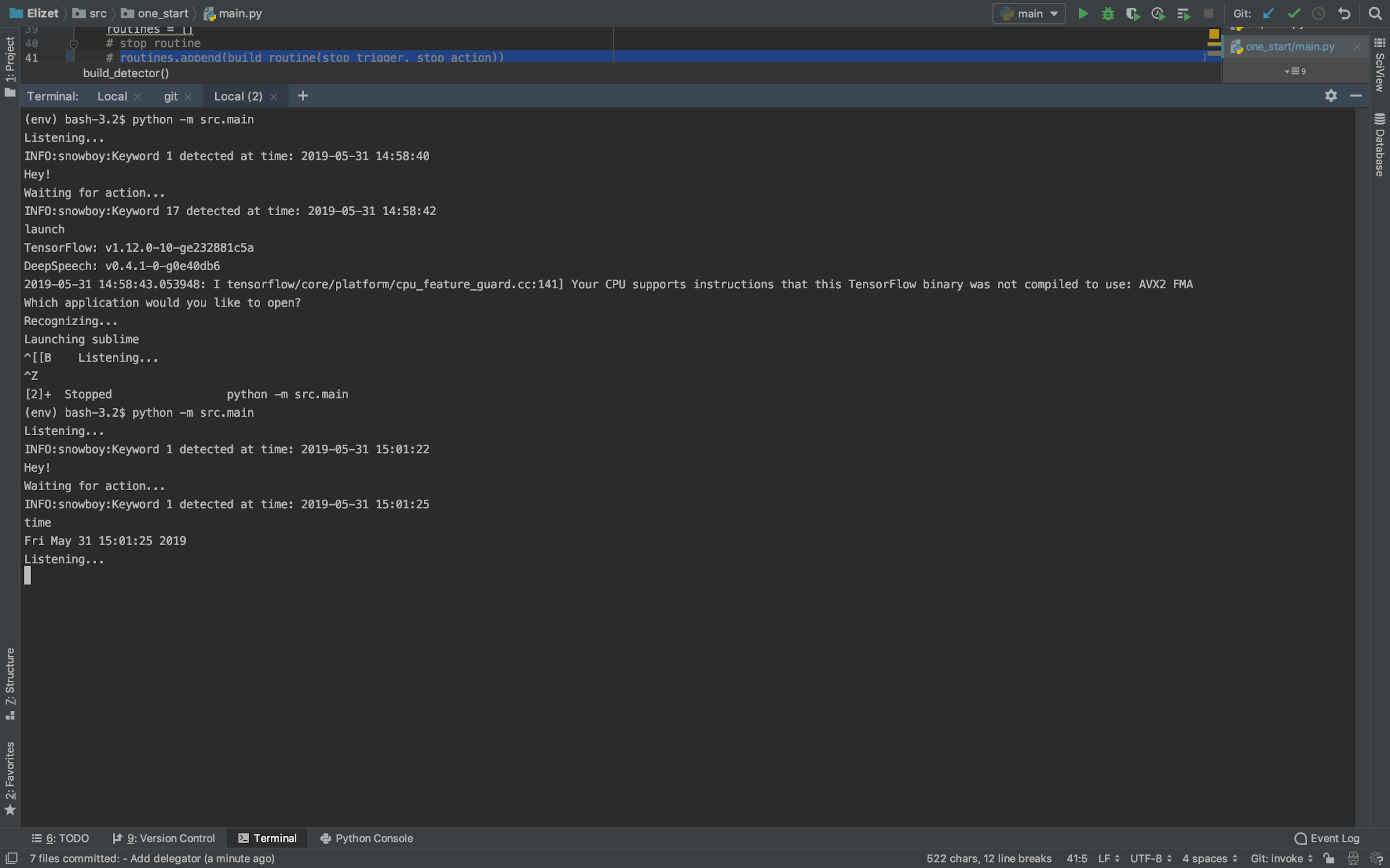
****

**List of collected data samples**

****

**One of the collected data samples**

## Eliza

****

**Version 1 of the application**

# Limitations, Recommendations and Conclusion

## Limitations

During the project implementation, a few challenges and hiccups were encountered that delayed, and often required a drastic shift in the intended implementation approach.

### Training Data

One of the factors that motivated this project was the lack of diversity in voice recognition data and subsequently, the models derived from this data. Acquiring this data, particularly that specific to Uganda, and by extension, East Africa was difficult.

### Substandard Recognition Models

A large part of this project involved identifying speech recognition engines/ models that are;

* Customisable
* Relatively easy and cheap to train
* Open source

### Evolving Research into Speech Recognition

Approaches to building these models is also under constantly evolving research, for example **todo** published research as I was right in the middle of compiling this report.

### Dynamic Computer Environments

In areas like file and application resolution, for example when opening a file and/ or application dynamic environment setups, particularly on a linux based system has been a challenge when testing the software on multiple systems, eg;

1. User and application permissions
2. Application types
3. Execution environment and software permission

## Recommendations

### Data Collection

As one of the largest parts of the project, several approaches were experimented with, both successfully and unsuccessfully. These included;

* Youtube data collection- Video and Captions. Of particular notes, is the VoxCeleb project used in this project, which although quite outdated, looked to use this approach. The goal here would be focus only on content from regions we are interested in optimising the models for.
* Collaboration with projects which by their nature involve large amounts of voice data. An example here is a project I recently got in touch with that is doing FM station monitoring. One of their approaches is continuously recording local radio stations and transcribing this data, which gives us valuable data with varying accents and model of speech in regions we are interested in.

### Action Resolution

The current approach to resolving actions after a user’s speech is detected and recognised is highly centralised and quite manual. This is obviously slightly remedied by the nature of this being a open source project, which lends itself to a relatively straightforward system of extensibility and relatively flat hierarchy that means the project can still be scalable. However, in the future, we would look to allow applications to automatically register actions they can handle and publish them to the software, to allow flexibility in extensibility, experimentation and modification. One approach would be to allow applications register something like an *actions.xml* or *actions.json* file listing what commands it can handle and what the possible speech patterns for invoking it would be.

# References

[1] Ian McGraw et all (10 Mar. 2016). Personalized Speech recognition on mobile devices [online], <https://arxiv.org/abs/1603.03185>. Accessed 22 Nov. 2018.

[2] Veton Këpuska and Gamal Bohouta (14 Mar. 2017). Comparing Speech Recognition Systems (Microsoft API, Google API and CMU Sphinx) [online], <https://www.researchgate.net/publication/314938892_Comparing_Speech_Recognition_Systems_Microsoft_API_Google_API_And_CMU_Sphinx>. Accessed 22 Nov. 2018.

[3] <https://wiki.gnome.org/Projects/GnomeVoiceControl>

[4] <https://simon.kde.org/>

[5] <https://jasperproject.github.io/>

[6] <https://wiki.ubuntu.com/SpeechControl>

[7] <http://kaldi-asr.org/doc/about.html>

[8] <http://publications.idiap.ch/downloads/papers/2012/Povey_ASRU2011_2011.pdf>

[9] <https://cmusphinx.github.io/wiki/>

[10] <https://cmusphinx.github.io/doc/pocketsphinx/>

[11] <https://voice.mozilla.org/>

[12] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition [online], <https://www.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/tutorial%20on%20hmm%20and%20applications.pdf>. Accessed 22 Nov. 2018.

[13] Deep Speech: Scaling up end-to-end speech recognition [online], <https://arxiv.org/abs/1412.5567>. Accessed 20 May 2019.

[14] Dependencies: [*https://github.com/Karuhanga/eliza/blob/development/requirements.txt*](https://github.com/Karuhanga/eliza/blob/development/requirements.txt)

# Appendix

## Open Source

### Data Sets Used By The Project

* Google Audio Set: [*https://research.google.com/audioset/dataset/speech.html*](https://research.google.com/audioset/dataset/speech.html)
* Mozilla Common Voice: [*https://voice.mozilla.org/en/datasets*](https://voice.mozilla.org/en/datasets)
* VoxCeleb: [*http://www.robots.ox.ac.uk/~vgg/data/voxceleb*](http://www.robots.ox.ac.uk/~vgg/data/voxceleb)
* LibriSpeech: [*http://www.openslr.org/12*](http://www.openslr.org/12)
* TED-Lium Dataset: [*https://www.openslr.org/51*](https://www.openslr.org/51)
* Vox Forge: [*http://voxforge.org/*](http://voxforge.org/)

### Project Dependencies[14]

* Mozilla Deep Speech: [*https://github.com/mozilla/DeepSpeech*](https://github.com/mozilla/DeepSpeech)
* Snowboy Hotword Detection: [*https://github.com/Kitt-AI/snowboy/*](https://github.com/Kitt-AI/snowboy/)

## Project tools

* ODK Server-<http://cs2-216003532-karuhanga-odk.appspot.com/>

### Github Repositories

* Virtual Assistant: <https://github.com/Karuhanga/eliza>
* Common Voice: [*https://github.com/common-voice-uganda/frontend-web*](https://github.com/common-voice-uganda/frontend-web)
* Common Voice: [*https://github.com/common-voice-uganda/backend*](https://github.com/common-voice-uganda/backend)

## External Tools

* Kaldi- speech recognition
* PocketSphinx- speech recognition
* pydub- audio processing
* pyaudio- audio device access
* sklearn- a bit of machine learning

## Roadmap

### Phase 1

* Setup ODK collection server and endpoints(Abandoned)
* Figure out a way of finding out which program is currently running(In progress)
* Evaluation of speech recognition alternatives (will make use of Kaldi for the more mature parts and PocketSphinx and sklearn for quick demos)(Done)
* Setup Kaldi(Done)
* Figure out a way of generating language phonemes that represent our accent(Abandoned)
* Demo keyboard control and automation(Done)
* Solve compatibility problems between the keyboard and pyaudio tools(Done)
* Demo sound based routine activation (this will be done by sound levels e.g clapping at this point) on select applications(Done)
* Collect keyword data based on single user (Using odk)(Abandoned)
* Perform data cleaning and normalisation(Partially Done, Abandoned)
  + Normalise amplitude(related to volume)
  + Remove leading and trailing silences
  + Normalise length
* Build speech recognition model for hotwords based on one user(Done)
* Demo hotword recognition, based on single user(Done)
* Fully automate keyboard input to allow us use hotkeys and shortcuts(Done)
* Integrate hotkey activation to hotword triggers(Done)
* Complete v1 build with three application hotword controls(Done)

### Phase 2(In progress)

* Expand range of words that can be recognised by model
* Remove single user dependency- start collecting hotword data from multiple individuals
* Improve data collection methods- make these more professional and increase scale
* Improve data cleaning methodology(Abandoned)
* Identify new model that is easier to set up than Kaldi(Done)
* Train model on new acquired data set(Done)
* Expand targeted applications(No done)
* Evaluate possibility of shifting from a control based system to a more intelligent assistant(Moved to v3)
* Passive search and trigger readiness- look into possibility of expanding scope to continuously scan the pc and build a structure that allows efficient search based on voice input/ grep it(Moved to v3)