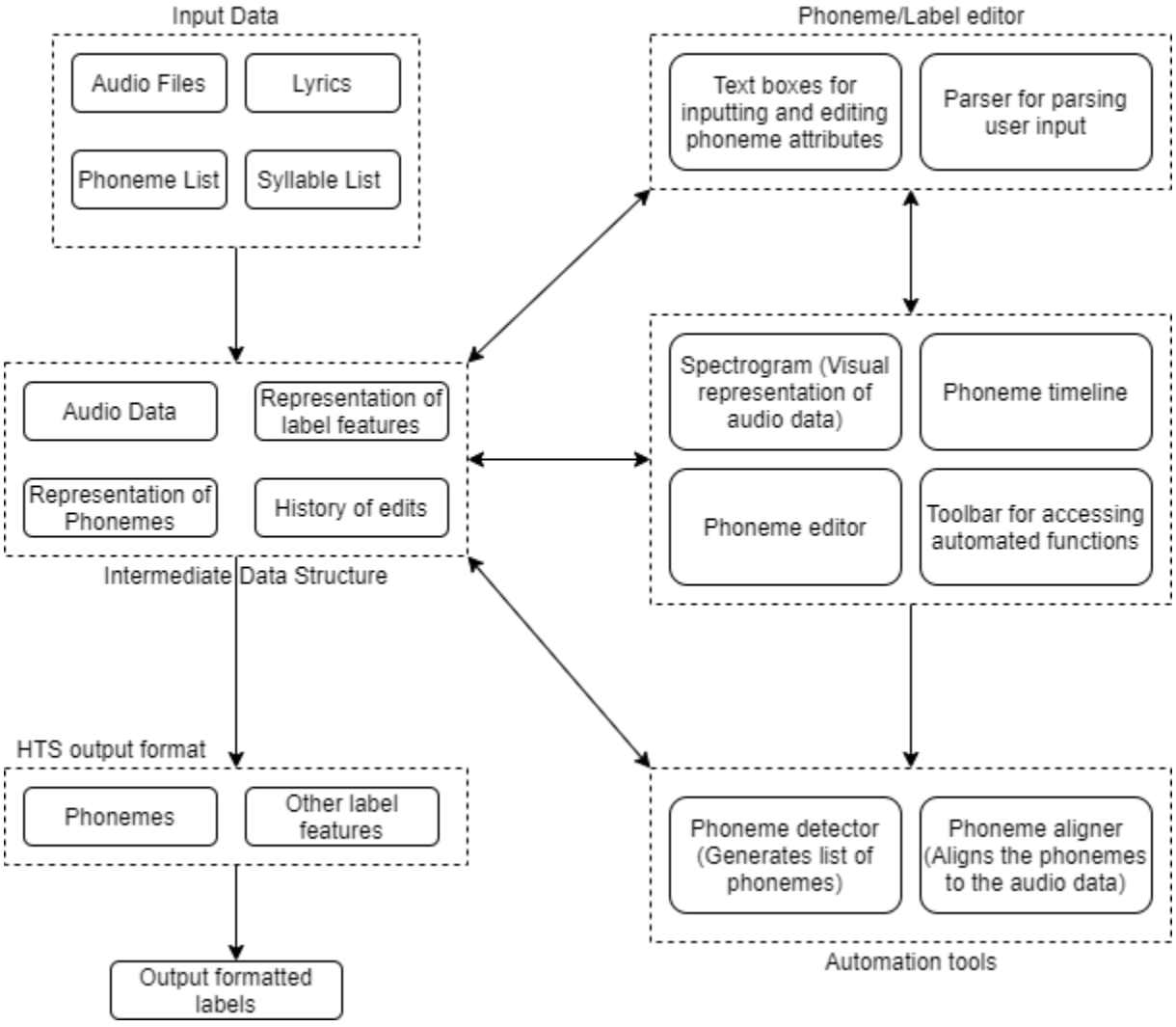
Singing Labeling Data Tool
Design Document

Group members:
Nandith Narayan nnarayan2018@my.fit.edu
Avinash Persaud apersaud2018@my.fit.edu
Carlos Cepeda ccepeda2018@my.fit.edu

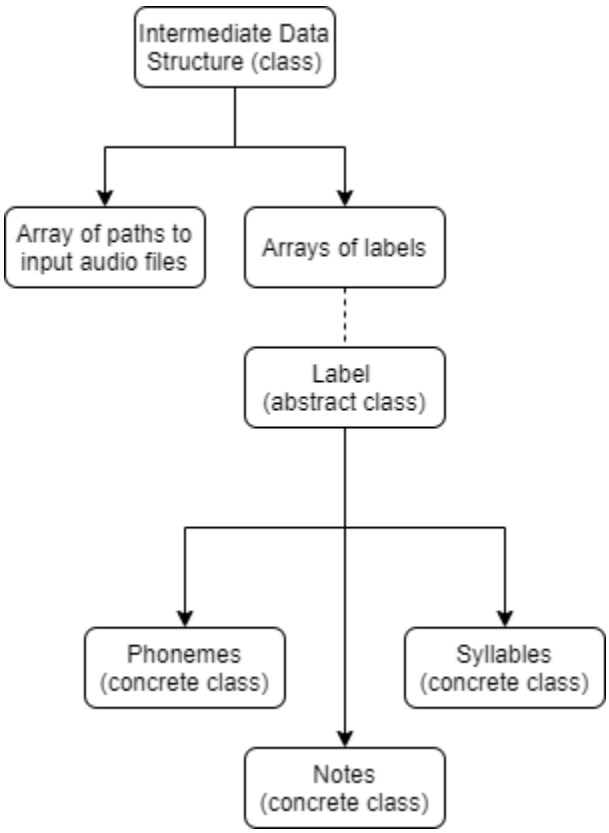
Advisor:
Dr. William Shoaff

System Architecture Diagram:



Input Data:
The user can import data into the project at any time. This data can be audio data, a list of phonemes, a list of notes, a list of syllables, and the lyrics of the song(s). The user will be presented with a Graphical User Interface to select which files to import into the project when they press the import data button.

Intermediate Data Structure:
The intermediate data structure will store all the data required by the tool and the data inputted by the user. This will be used to let the user save a project to load it later and resume labeling. It will be a class which contains arrays of instances of label classes. These label classes will all inherit from an abstract label class. So, every different type of label will have its own class. The audio files will be represented by their path.



Upon clicking the save button, the contents of the intermediate data structure will be serialized and written to files within a project folder. Upon loading a project, all the data from the saved files will be read and parsed into the intermediate data structure. The data will be stored in a text format and the audio files will be stored as a path to the file rather than the file itself.

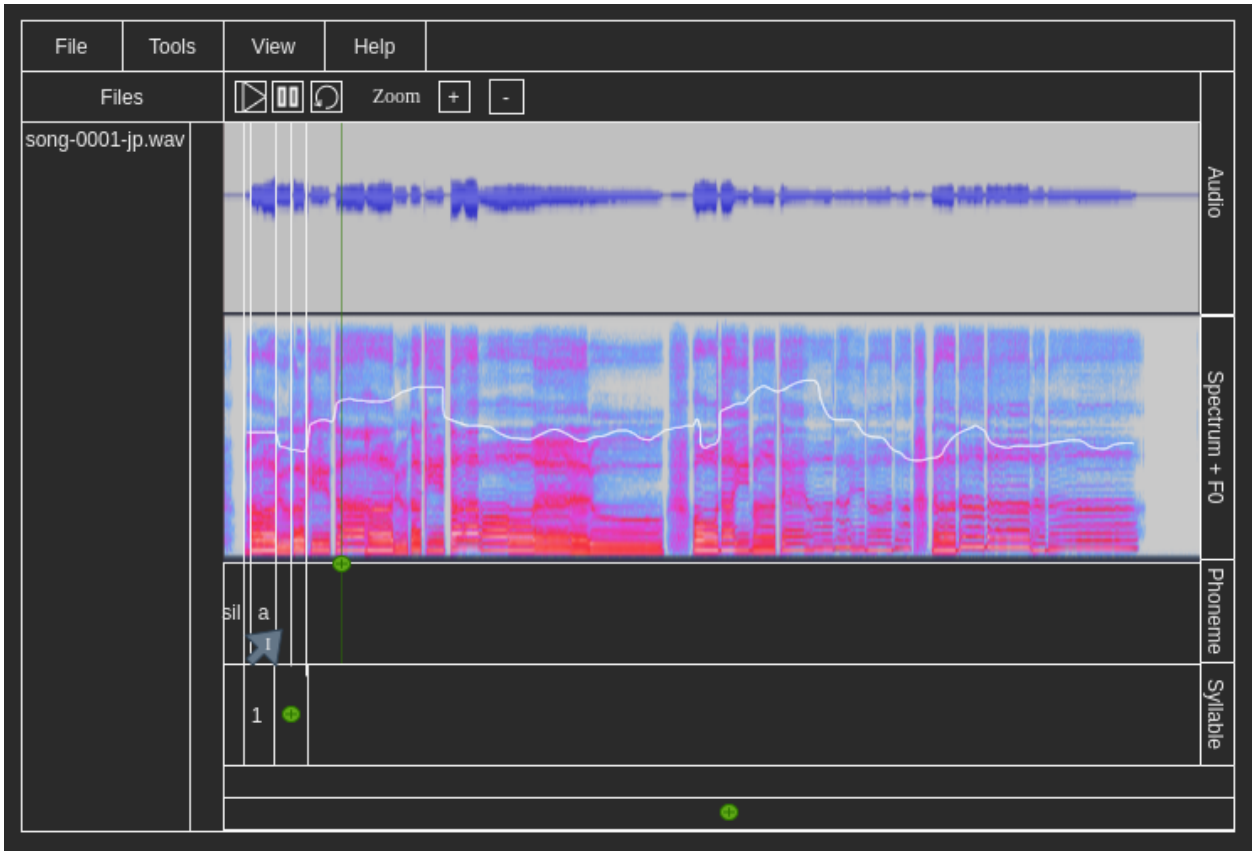
Phoneme/Label Editor
This editor will contain text boxes for the user to input phoneme attributes. The Graphical User Interface for this editor will contain buttons to add/remove attributes as well as edit them. All the text from every text box will be concatenated and parsed as a whole. Parsing will use Antlr v4 as a parser generator.

Example grammar:

S -> statement*
statement -> assign | operation
assign -> label (dot attribute)? = value
label -> type openSqrBracket index closeSqrBracket
attribute -> identifier
operation -> label (dot attribute)? binaryOp value
binaryOp -> '*' | '-' | '+' | '/' | '^' | '%'
value -> integer
type -> identifier
index -> integer
dot -> '.'
openSqrBracket -> '['
closeSqrBracket -> ']'
identifier -> [a-z|A-Z|_]*
integer -> ‘-’? [0-9]*

Automation tools
The user will be able to click on the toolbar and select from a list of tools for automation. These tools consist of automatic phoneme detection, and automatic phoneme alignment. Automatic phoneme detection will generate a list of phonemes. Automatic phoneme alignment will align those phonemes to the audio data.

Graphical User Interface
The graphical user interface will contain multiple scenes. These scenes are listed as follows, the phoneme editor, the primary labeling scene, the import data scene, and the export scene. The primary labeling scene will display a graphical representation of the raw audio data as well as a spectrogram of the frequencies present in the audio data. It will also contain a timeline for adding, removing, editing, and duplicating phonemes. Other feature layers can be added and edited in a similar matter, including user defined layers based on premade layer types.



The import data scene will open up a window that will let the user select which files they desire to import into the project. The export data scene will open up a window that will let the user select where they want the output to be written to, as well as the format to output the data as. The user will be able to use shortcuts like ctrl+c, ctrl+v, and ctrl+d to copy, paste, duplicate phonemes on the timeline. The user will be able to use the scroll wheel to change the scaling of the timeline and audio data.

HTS output format

The HTS output format represents the labels as regular expressions. The specification for this format is as follows:

$p_1\hat{}p_2-p_3+p_4=p_5@p_6\_p_7$
/A:$a_1\_a_2\_a_3$ /B:$b_1-b_2-b_3@b_4-b_5\&b_6-b_7\#b_8-b_9\$b_{10}-b_{11}!b_{12}-b_{13};b_{14}-b_{15}|b_{16}$ /C:$c_1+c_2+c_3$
/D:$d_1\_d_2$ /E:$e_1+e_2@e_3+e_4\&e_5+e_6\#e_7+e_8$ /F:$f_1\_f_2$
/G:$g_1\_g_2$ /H:$h_1=h_2\hat{}h_3=h_4|h_5$ /I:$i_1=i_2$
/J:$j_1+j_2-j_3$

p_1	the phoneme identity before the previous phoneme
p_2	the previous phoneme identity
p_3	the current phoneme identity
p_4	the next phoneme identity
p_5	the phoneme after the next phoneme identity
p_6	position of the current phoneme identity in the current syllable (forward)
p_7	position of the current phoneme identity in the current syllable (backward)
a_1	whether the previous syllable stressed or not (0: not stressed, 1: stressed)
a_2	whether the previous syllable accented or not (0: not accented, 1: accented)
a_3	the number of phonemes in the previous syllable
b_1	whether the current syllable stressed or not (0: not stressed, 1: stressed)
b_2	whether the current syllable accented or not (0: not accented, 1: accented)
b_3	the number of phonemes in the current syllable
b_4	position of the current syllable in the current word (forward)
b_5	position of the current syllable in the current word (backward)
b_6	position of the current syllable in the current phrase (forward)
b_7	position of the current syllable in the current phrase (backward)
b_8	the number of stressed syllables before the current syllable in the current phrase
b_9	the number of stressed syllables after the current syllable in the current phrase
b_{10}	the number of accented syllables before the current syllable in the current phrase
b_{11}	the number of accented syllables after the current syllable in the current phrase
b_{12}	the distance per syllable from the previous stressed syllable to the current syllable
b_{13}	the distance per syllable from the current syllable to the next stressed syllable
b_{14}	the distance per syllable from the previous accented syllable to the current syllable
b_{15}	the distance per syllable from the current syllable to the next accented syllable
b_{16}	name of the vowel of the current syllable
c_1	whether the next syllable stressed or not (0: not stressed, 1: stressed)
c_2	whether the next syllable accented or not (0: not accented, 1: accented)
c_3	the number of phonemes in the next syllable
d_1	gpos (guess part-of-speech) of the previous word
d_2	the number of syllables in the previous word
e_1	gpos (guess part-of-speech) of the current word
e_2	the number of syllables in the current word
e_3	position of the current word in the current phrase (forward)
e_4	position of the current word in the current phrase (backward)
e_5	the number of content words before the current word in the current phrase
e_6	the number of content words after the current word in the current phrase
e_7	the distance per word from the previous content word to the current word
e_8	the distance per word from the current word to the next content word
f_1	gpos (guess part-of-speech) of the next word
f_2	the number of syllables in the next word
g_1	the number of syllables in the previous phrase
g_2	the number of words in the previous phrase
h_1	the number of syllables in the current phrase
h_2	the number of words in the current phrase
h_3	position of the current phrase in this utterance (forward)
h_4	position of the current phrase in this utterance (backward)
h_5	TOBI endtone of the current phrase
i_1	the number of syllables in the next phrase
i_2	the number of words in the next phrase
j_1	the number of syllables in this utterance
j_2	the number of words in this utterance
j_3	the number of phrases in this utterance

Time Level User Interface

The main project level User Interface will display the spectrogram and waveform of the audio data as well as feature layers in a timeline format. By scrolling with the mouse wheel, the time scale can be changed. The spectrogram and waveform display will be generated dynamically. Feature layers will be overlaid on top of the spectrogram. The time dependent parts of the UI will be drawn using a time_scale value that will be modified when the user scrolls the mouse wheel or manually sets it in a dropdown menu.