Singing Data Labeling Tool
Milestone 5 Progress Evaluation Report

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

Advisor:
Dr. William Shoaff wds@fit.edu

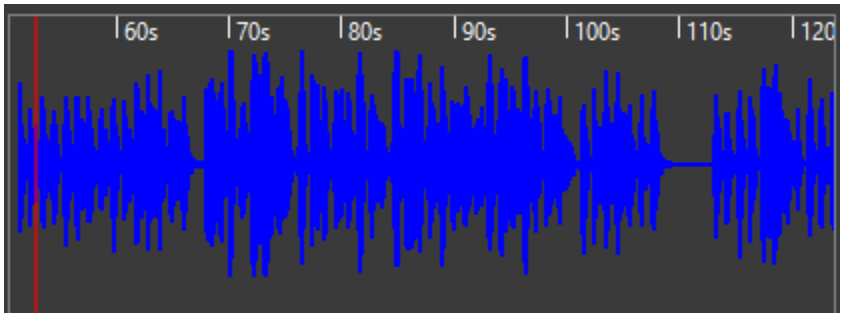
Client:
Caleb Matthew Long, Appalachian State University

Task	Completion %	Nandith Narayan	Avinash Persaud	Carlos Cepeda	To Do
Project Save/Load	100%		100%		Will be continuously updated as project data evolves
Wave draw Time Ticks	100%	100%			Spacing of ticks can be adjusted if needed
Timeline view	80%	40%	40%		Text needs to be displayed on each label. Need more keyboard shortcuts for actions such as duplicate, insert, delete, etc.
Audio Playback	10%		10%		Needs to be implemented
Showcase Poster	100%	100%			Poster will get updated with feedback from advisor & instructor.

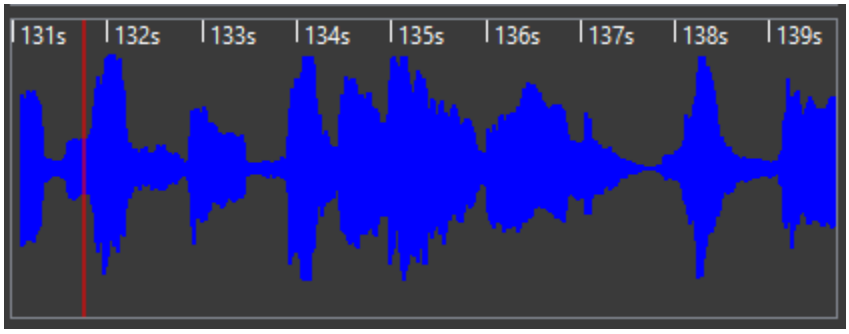
Create note view UI element	15%			15%	Implement labels representing notes in the song and their respective waveforms and phonemes

Task Discussion:
Task 1: Using RapidJSON, the project's intermediate data structure is saved into a json file. This file is read back in RapidJSON's full precision mode so that loss of accuracy does not occur. The choice to use a text format such as JSON instead of a binary file was so that the data can be easily manipulated externally and so version control can be more readable.

Task 2 To add ticks to the wave draw widget, we first computed the range of time that the current view window covered. Since we store the left and right (start and end) positions of the window, we can calculate the time range by multiplying the difference of the two positions with the sample rate. Using the time range, we then decide if the time ticks should be in increments of decaseconds, seconds, deciseconds, centiseconds, or milliseconds. Using this time increment, we then draw vertical lines or 'ticks' to visualize the time at any given point. We additionally compute the time in seconds at each tick and display it as text.
Tick Example at 10s of seconds (decaseconds):



Tick Example at seconds:



The time ticks dynamically adjust according to the current zoom amount.

Task 3: The commonly used portaudio library was selected. First implementation revision will play back visible portions. A future update will playback user selected portions as well.

Task 4: The timeline widget required a lot of various pieces. First of all, Avinash designed and implemented the data structure to store the labels associated with their start time. He also wrote functions to insert, remove, move, and set these labels. Then, Nandith used these functions to create the GUI. To do this, a graphical object was created to represent each label. Then, the positions of the labels are calculated and only the visible labels are drawn to the screen. Then, the wave draw widget was made to add a label to the timeline on a left click. Next, movement of labels was added. To move labels, first the label that the user clicked on was calculated and grayed out. Next, a display of what the new label's position would be is displayed right above the old label. This display updates with the user's cursor position. Then the user can place the label by left clicking. The user can abort the move by right clicking.

Original labels on the timeline:

Label seleted to be moved:

Label successfully moved:

Task 6: The noteview UI element is supposed to display the notes present in the song along with the corresponding waveforms and phonemes. This part of the milestone is still being worked on, but a demo is planned for the day of the presentation. This feature is behind schedule due to lack of knowledge on the matter, so considerable time is going to learning how to implement the piano roll using the Qt GUI framework. The struggle is figuring out how to implement the area where the notes will be displayed and how to sync the scrolling feature with the virtual piano keys.

Milestone 6 task matrix:

Task	Nandith Narayan	Avinash Persaud	Carlos Cepeda
Audio Playback	Sync playback cursor to audio playback	Use portaudio to properly start and stop audio playback	
Polish Timeline	Add text to the timeline	Fix a potential bug with removing the last label	
Integrate Machine Learning models	Integrate classification model	Integrate Boundary detection model	
Implement Remaining Track Types	Create “Add Track” dialogue with custom and preset options	Write data structure logic and update save logic.	
Complete implementation of noteview UI element	Help create data structure for notes being displayed	Help create data structure for notes being displayed	Implementing the GUI that will be presented to the user along with linking it to the data structure

Milestone 6 Task Discussion:

Task 1: When playing audio, it would be useful if the user could see which part of the audio was being played back. In addition, keyboard shortcuts to play/pause/stop the playback would be needed. We plan to finish this before our presentation on the 30th.

Task 2: Polishing the timeline requires adding text annotations to the labels on the timeline as well as more keyboard shortcuts for common actions.

Task 3: Integrating the machine learning models poses a challenge for us. We plan to either bundle the python script with the tool, or try to use the C++ tensorflow library since it’s a relatively simple CNN. The preprocessing poses a challenge though. This is because the preprocessing step requires the use of multiple python libraries. If we aren’t able to get the C++ implementation working in time, we’ll just bundle the tool with our python scripts.

Task 4: To fully encompass all the data needed for the HTS format several other track types other than text (allows spaces) and word (doesn’t allow spaces) are needed. These include autonumber (linearly numbers each entry), integer, real numbers, and most importantly the note track.

Task 5: As mentioned earlier, the goal of M5 was to have a basic demo that allowed the user to interact with the piano roll; the demo will only allow the user to Once that is taken care of, the data structure of the noteview will need to contain information about the individual notes, and the specific properties of the notes are decided based on the HTS label format. For M6 the noteview will display the lyric, phoneme, and pitch of each notes of the song, and the user will be able to change how each musical measure will be divided.

Client Feedback

Discussed with the client over discord occasionally.
Meeting Date: **3/20/2022**
Try to make the Spectrogram clearer, visually it's harder to follow the formants than other programs like wavesurfer. Add features from wavesurfer such as a pitch estimation view and the ability to add a label on typing.

Date of meeting with Faculty Advisor: **3/21/22**

Faculty Advisor feedback on milestone tasks:

- Evaluation by Faculty Advisor
- Faculty Advisor: detach and return this page to Dr. Chan (HC 214) or email the scores to pkc@cs.fit.edu
- Score (0-10) for each member: circle a score (or circle two adjacent scores for .25 or write down a real number between 0 and 10)

Carlos Cepeda	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Avinash Persaud	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Nandith Narayan	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10

- Faculty Advisor Signature: _____ Date: _____