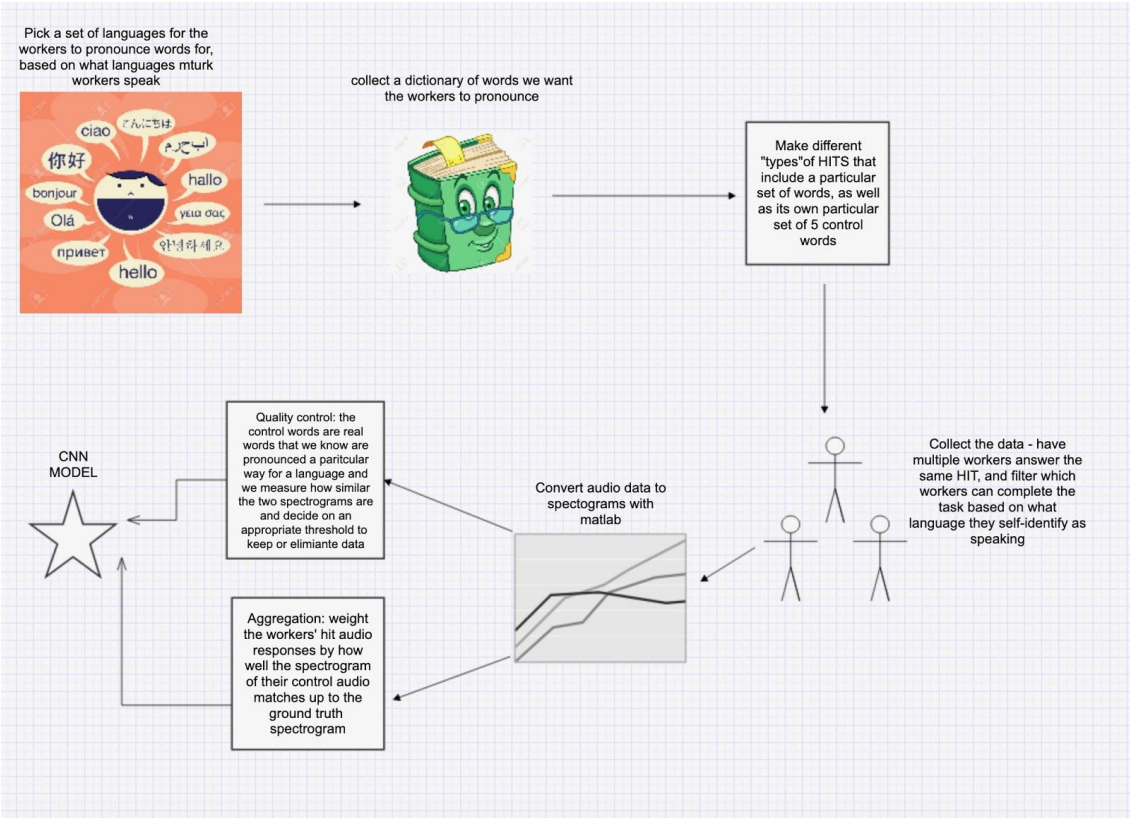
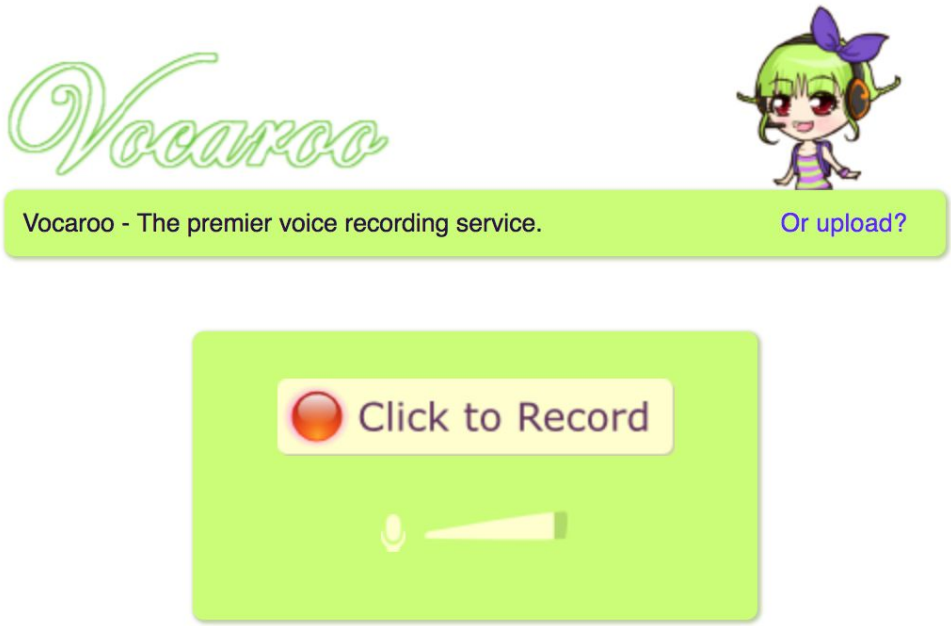
Mia Chiquier, Dan Hayes, Kolade Lawal, Connor McSwain, Richard Wu
4/4/19

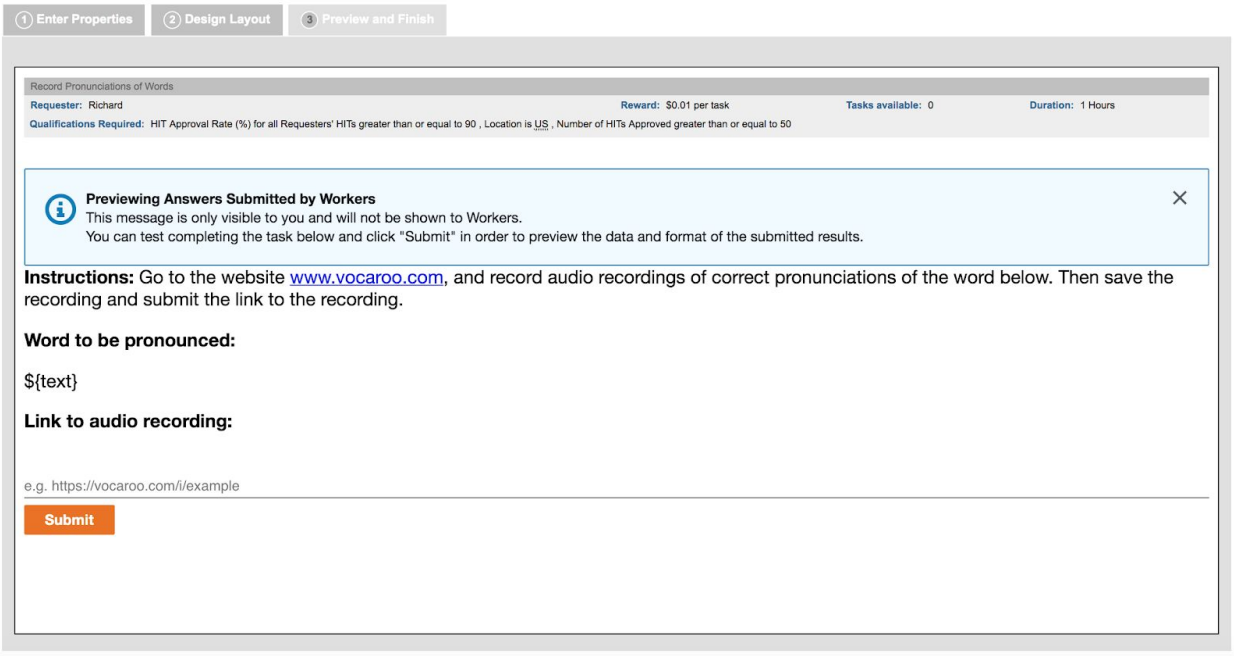
Final Project Deliverable 1

Flow diagram of major system components



Mockups of any user-facing interfaces (crowdworkers and end-users)





README:

1.) We are going to begin by deciding the specific set of languages that the workers will be asked to pronounce. This decision is going to be based on multiple factors. First, we need to determine what languages are feasible for MTurk workers, based on the languages that the MTurk workers speak. We are also going to make sure that we have a wide range of origins for the languages we choose, because if all of the languages are too similar the CNN model will have more difficulty differentiating them. (1 point)

2.) Collect a dictionary of words we want the workers to pronounce. This dictionary will be approximately 1000 words per language (this will depend on how much funding can be secured). The words are going to be provided by a dictionary of multiple languages. We will aim to pick the most common words in each language if we can find resources to do that for each language. (1 point)

3.) Make different "types"of HITS that include a particular set of words, as well as its own particular set of 5 control words. The control words come from annotated audio data (ex. We have french audio data for which we know what word it is and that it is in French). We will specify on the HIT that there should be no background noise when recording. (2 points)

4.) Collect the data - have multiple workers answer the same HIT, and filter which workers can complete the task based on what language they self-identify as speaking. We assume that we want a large baseline of hits for each word so that the classifier can learn about the pronunciation of each word from many recordings. *(2 points)*

5.) Convert audio data to spectrograms with matlab through the function plt.specgram. The audio recordings can be downloaded as mp3 or wav files, and converting them to spectrograms will be a relatively easy step. This is necessary though because it is easier for classifiers to look at spectrograms than classify audio. *(1 point)*

6.) Quality control: the control words are real words that we know are pronounced a particular way for a language (we'll have the audio files of the correct pronunciation, which we will convert to spectrograms) and we measure how similar the two spectrograms are and decide on an appropriate threshold to keep or eliminate data. (3 points)

7.) In the aggregation step, we are going to use the quality rankings of each worker to weight the hit audio responses by how well the spectrogram of their control audio matches up to the ground truth spectrogram. Transforming the spectrogram into the Fourier domain and gathering the top 5 spatio-temporal frequencies and finding the difference of these values for two spectrograms. Intuitively, the word "food" would have a very stable spectrogram whereas "caterpillar" would have a very wavey one and so they would have different frequencies in the spatio-temporal domain. We can also use sound-frequency measure from the audio-file as an additional difference estimator. If a file/image has low difference with the mean then we know that it is a good data point. (4 points)

8.) CNN model in Pytorch. This model will be composed of grey-scale images. In order to keep the size of the input grey-scale images constant, we will take the audio of all the words of the same language and concatenate them together, make one giant long spectrogram, and divide that spectrogram evenly such that all the images are equal in size. After a few convolutional layers, there will be a fully connected layer whose activation function will classify which of the languages this language belongs to. (4 points)

Total points: 18