Completed:

I created a listener class that takes in audio input for a given duration and outputs an audio buffer periodically. This buffer is passed to a calculator class that I created which performs calculations on the audio. The calculations include determining the note being played, the accuracy of each note (cents), creating an array of onsets (error-prone but explained below), and the root mean square graph. These calculations are put into a pandas data frame that we use for comparison to the original musical score. I also found a much faster way of calculating which notes were being played, bringing calculation times from about 8 seconds to about 0.1 seconds on a recording of about 10 seconds. With this, we will be able to perform these calculations in real time and generate comparison outputs in real time as well.

The code for these classes can be found here in Listener.py and Calculator.py
The code can be found in main here in AudioThreadTest.py, which is just most of the code I wrote before organizing it into two separate classes.

We ran into an issue of calculating onsets (where a new note begins) for string instruments. Before, I was using the Librosa onset_detect() function to determine where new notes started. However, we found that this onset detection method does not work well with string instruments due to the less varied RMS when playing. In other words, the energy of the instrument does not change as much between new notes on string instruments as it would on an instrument like piano. Currently, we are doing comparisons based on the start time of each note in a given score. Whenever the provided score expects a new note, we simply check if the player is playing that note. This is a temporary solution that has some obvious drawbacks. (Such as playing one note continuously when you should be playing it repeatedly E.g. one long C vs 10 short Cs)

We did some initial research on potential alternatives. We found some papers that claim to be able to detect onsets for string instruments with high accuracy, such as: here and here. However, we could not find any actual code to test out.

We also briefly looked into libraries such as BeatNet (paper and Github) and autocorrelation (explanation and Librosa function) for potential alternatives to finding the onsets. For example, if we can effectively determine the beats of someone's playing using BeatNet, we can potentially compare that to the actual beats of the piece and determine where mistakes are made.

In Progress / For future:

The first thing that needs to be done in the future is to find an alternative method of determining onsets for string instruments or a different approach altogether. While the current code works well for parsing and comparing music from instruments like piano, it struggles due to the difficulties in calculating onsets for string instruments. A good place to start would be the papers that we have begun looking into.

Then, we can aim to make the calculations and comparisons happen in real time. Currently, the main reason that we are doing all calculations and comparisons after the recording is due to the high latency of determining what notes were played. However, we can avoid this latency with the new function that I have implemented (just set the fast parameter to True in the Calculator). To make things in real time, some modifications will need to be made to the Listener class so that it can call calculate periodically rather than just output a buffer at the end for us to manually pass.