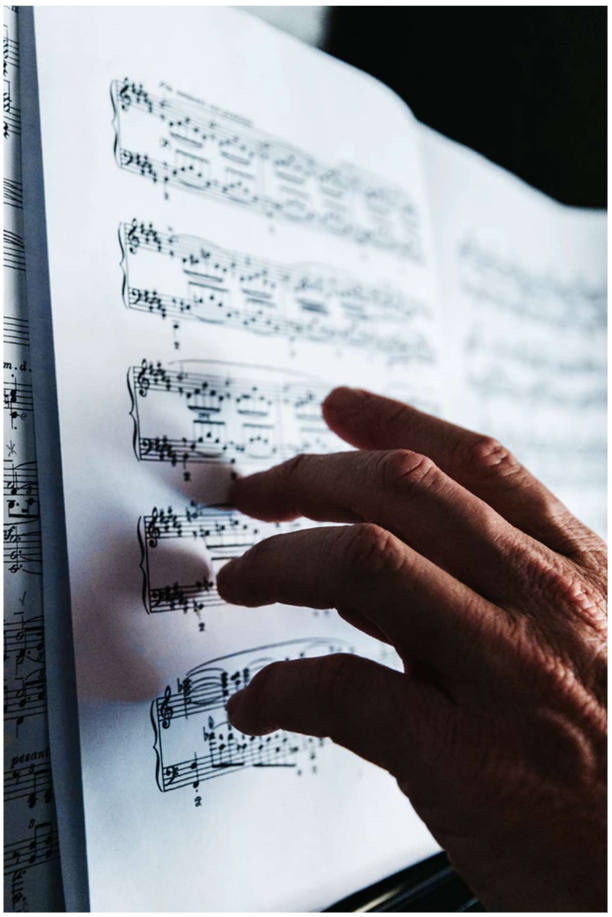
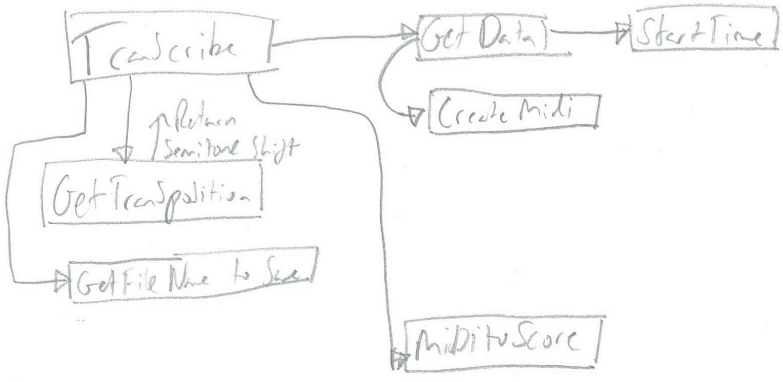
4097 Oscar Lindenbaum

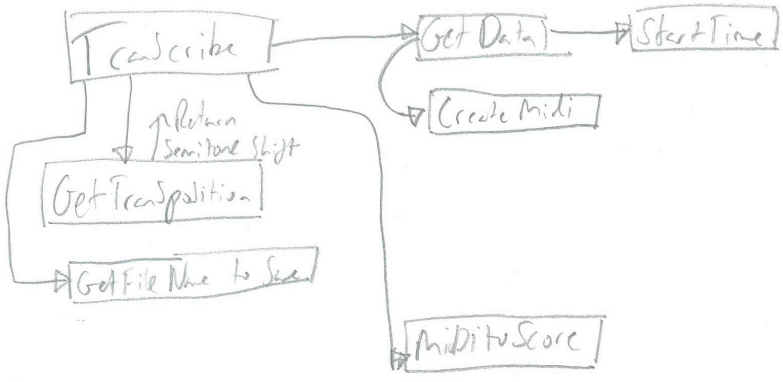
NoteSwitch

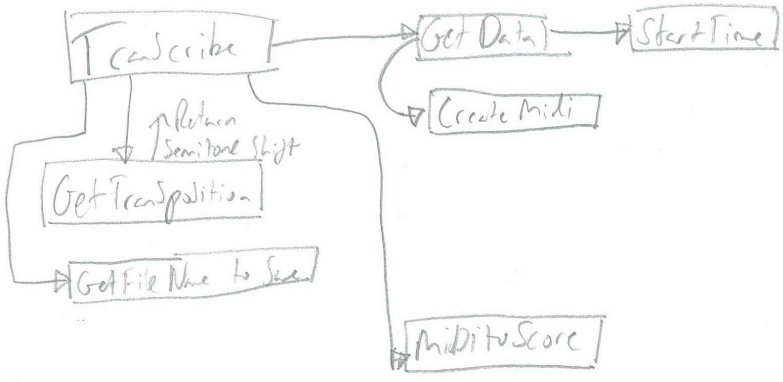


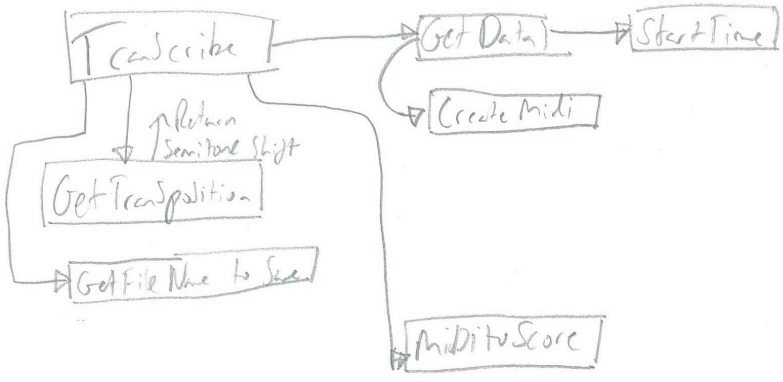
Contents

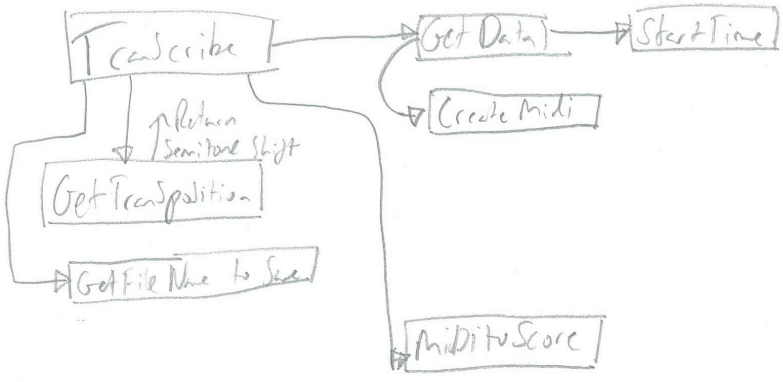


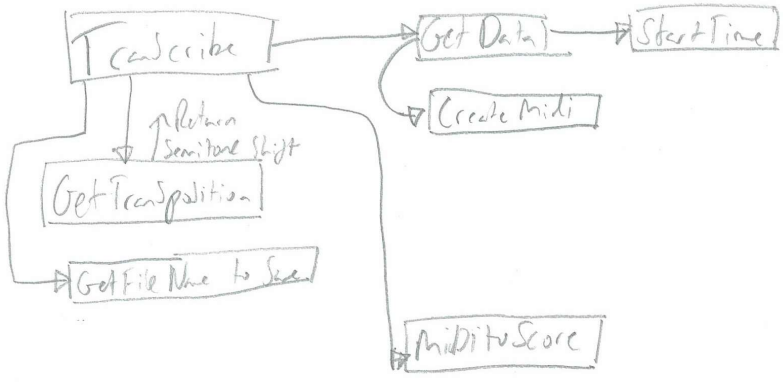
Computer Science NEA - NoteSwitch

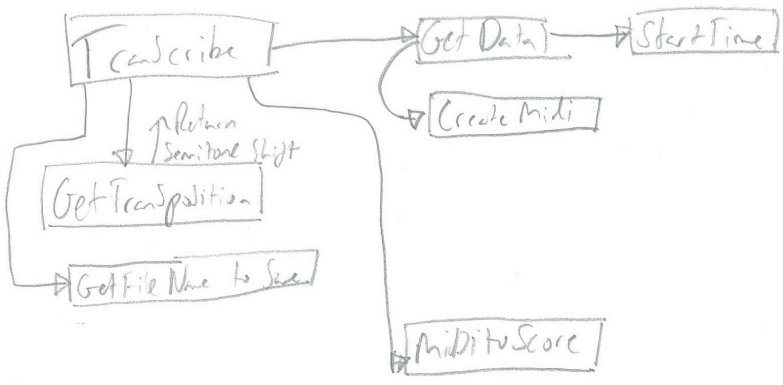












Analysis

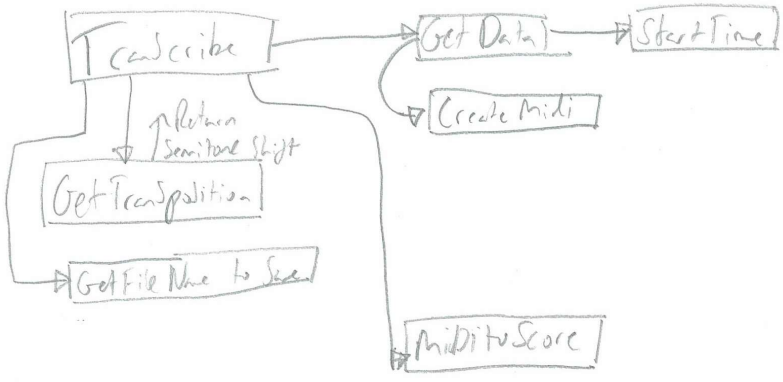
Background of the Project

Currently, it is complicated to either transpose or identify the notes being played on any musical instrument without years of learning by ear and perfect pitch. As a musician myself I have found on multiple occasions that I create a melody for my main instrument and I'd like to play it on a different one. This is when the problem arises as I would have to transpose the melody to the other key in order for it to be played correctly, I would then have to write it down and repeat the process if anyone else needed to play. If it was the case where I was writing a composition for a band I would have to manually notate and transcribe the tune for each key because most instruments are in a different key. What the app will do is make it easier to switch between instruments with either written music or single notes without a lot of thinking and working out the notes one by one.

The Project

The separate functions that the app would do to solve the problem would be:

- ➔ Identify Single Notes being played by the given starting instrument and displaying their transposed counterpart to the user. The output could either be a text output or a graphical output such as a score.
- ➔ Identify multiple notes being played into the application and notate the rhythms and the different transposed notes onto a score. This file would then be outputted for the user.
    - ◆ If possible an option the app could have is that it would output both the midi file used to create the score file and the score file itself,what this would allow the user to put the recorded piece into another 3rd party software such as



MuseScore which would then allow corrections or alterations to be made to the sheet music. What this would additionally do would be to record multiple tracks for example for an arrangement produced for a larger band.

- ➔ One additional feature which would not be a part of my essential solution would be to be able to tune your instrument with it. This would be done by analysing the pitch and deciding the 'level' of accuracy for the tuning which is needed to be attained.

What Research Do I Need To Do?

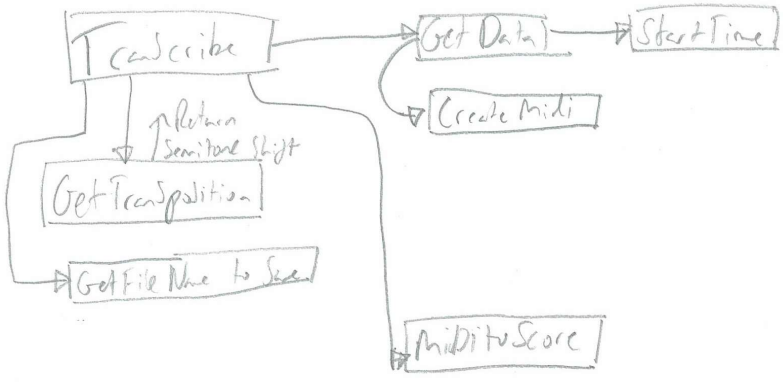
In order for the project to be successful I will need to research about how computers interpret audio and how it is stored, in addition to this I will need to understand how I can extract frequency values from live audio with the process being fast enough for the real time audio stream. I will do this by analysing and selecting the most suitable algorithm for the task at hand. Another area I will need to look into is MIDI files, what data they contain and how they are created. This is so I can manually create the MIDI file and make sure I am collecting all the necessary data.

Audio Processing

One of the main challenges for me is going to be extracting the data required from either a live audio source or from an audio file, the reason this is so difficult is that the audio I will be using will be analogue which means the data is continuous. Another aspect I will have to consider is if the program is processing a live feed the algorithms and operations carried out will have to be as efficient as possible in order to keep up with the feed - some possible factors that would affect this is the Chunk size (The buffer of data we will be processing at each instance) and the sampling rate, by reducing the sampling rate we would have less complex data to process (This rate should still obey Nyquist's theorem).

In order to maximise processing speed, I could use parallel signal processing, this would be one of the final objectives in my solution as it might require a restructuring of the whole program. One way I could do this was to implement threading into my program. (Documentation - http://www.wseas.us/e-library/transactions/computers/2010/89-308.pdf)

One restriction I have found is that the audio coming into the program will have to be monophonic or polyphonic with one main lead instrument which is dramatically louder and



clearer than the rest of the audio - this is because of the difficulty in separating the instruments and identifying whose voice is, as the data is continuous. One method I could use to reduce this factor is to have a certain level of 'noise' which I am going to disregard and then process the rest of the audio as one instrument.

Fourier Transforms

A Fourier transform provides the means to break up a complicated signal, like a musical tone, into its constituent sinusoids. - https://ac.els-cdn.com/S1877050915020281/1-s2.0-S1877050915020281-main.pdf?_tid=a8cc4b48-4b3f-4ca0-b878-4ea1b8e99b9c&acdnat=1530539823_51f11584e05adc331dd611fa26bd8b79 Page 878.

Commented [1]: https://ac.els-cdn.com/S1877050915020281/1-s2.0-S1877050915020281-main.pdf?_tid=a8cc4b48-4b3f-4ca0-b878-4ea1b8e99b9c&acdnat=1530539823_51f11584e05adc331dd611fa26bd8b79

Has a lot of good info

Commented [2]: http://mac.citi.sinica.edu.tw/~yang/teaching/lecture02_stft.pdf

Commented [3]: describe what it would be used for

In order to process the live audio feed with the fastest processing time, the number of comparisons or calculations must be minimised.

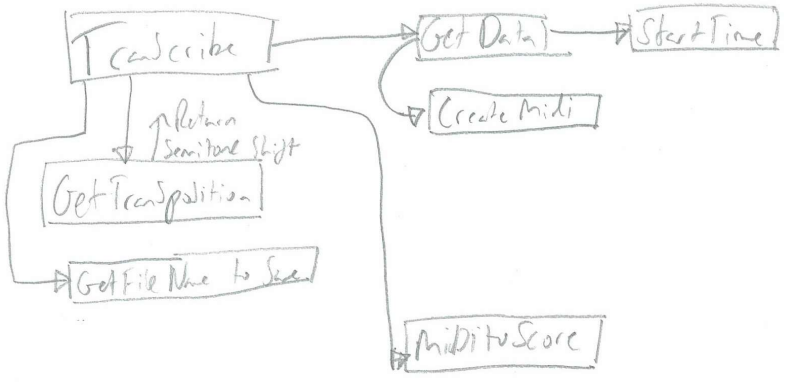
7.3.2 Computational speed of FFT

The DFT requires N^2 complex multiplications. At each stage of the FFT (i.e. each halving) $\frac{N}{2}$ complex multiplications are required to combine the results of the previous stage. Since there are $(log_2 N)$ stages, the number of complex multiplications required to evaluate an N-point DFT with the FFT is approximately $N/2log_2 N$ (approximately because multiplications by factors such as W_N^0, $W_N^{\frac{N}{2}}$, $W_N^{\frac{N}{4}}$ and $W_N^{\frac{3N}{4}}$ are really just complex additions and subtractions).

N	N^2 (DFT)	$\frac{N}{2}log_2 N$ (FFT)	saving
32	1,024	80	92%
256	65,536	1,024	98%
1,024	1,048,576	5,120	99.5%

Source http://www.robots.ox.ac.uk/~sjrob/Teaching/SP/l7.pdf page 95.

As described in the documentation above the discrete fourier transform has a complexity of

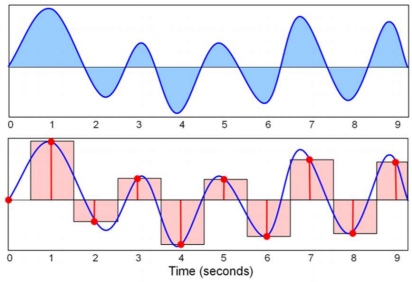


N^2 multiplications, therefore this algorithm has the necessary speed for the program.

Discrete Fourier Transform

The discrete Fourier transform approximates the continuous Fourier transform (Riemann sum approximation)

$$\begin{aligned}\hat{x}(\omega) &= \sum_{n\in\mathbb{Z}} x(n)e^{-j\omega n} \\ &= \sum_{n\in\mathbb{Z}} f(nT)e^{-j\omega n} \\ &\approx \int_{t\in\mathbb{R}} f(nT)e^{-j\omega t}dt \\ &= \frac{1}{T}\int_{t\in\mathbb{R}} f(nT)e^{-\frac{j\omega t}{T}}dt \\ &= \frac{1}{T}F\left(\frac{\omega}{T}\right) \qquad (21)\end{aligned}$$



From: M. Mueller, *Fundamentals of Music Processing*, Chapter 2, Springer 2015

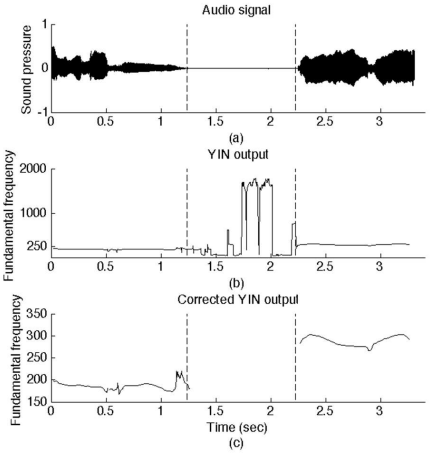
Source http://mac.citi.sinica.edu.tw/~yang/teaching/lecture02_stft.pdf *Slide 26*

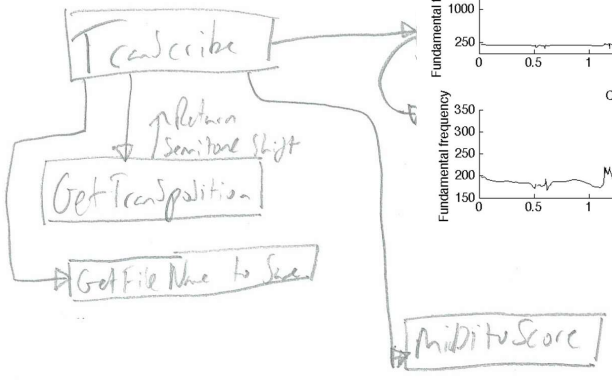
Continuous Fourier Transform (Discrete-Time Fourier Transform)

DTFT is another method that achieves similar results but this time on continuous data, because my program would run on a lightweight computer DTFT might not be the most practical for analysing the audio. This would be because we can't use a computer to calculate with data which is continually increasing in length as the processing power required would be large.

The Yin Algorithm

An algorithm is presented for the estimation of the fundamental frequency (F0) of speech or musical sounds. It is based on the well-known autocorrelation method with a number of





modifications that combine to prevent errors. The algorithm has several desirable features. Error rates are about three times lower than the best competing methods. There is no upper limit on the frequency search range, so the algorithm is suited for high-pitched instruments. The algorithm is also relatively simple and low latency.

The Algorithm can identify the note onsets and the pitch in five steps.

1. Difference Function
   If we model the signal as a difference function, an increase in signal amplitude with time causes the ACF peak amplitudes to grow with lag rather than remain constant. This encourages the algorithm to choose a higher-order peak creating an amplitude bias.

$$d(\tau) = \sum_{j=1}^{j=W} (x_j - x_{j+\tau})^2$$

$$d(\tau) = \sum_{j=1}^{j=W} (r_t(0) + r_{t+\tau}(0) - 2r_t(\tau)$$

2. Cumulative Mean Normalization
   This new function is obtained by dividing each value of the old by its average over shorter-lag values. It differs from d(τ) in that it starts at 1 rather than 0, tends to remain large at low lags, and drops below 1 only where d(τ) falls below average.

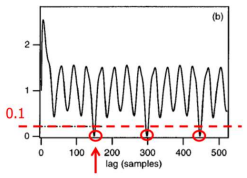
$$d_t(\tau') = \frac{d_t(\tau)}{(1/\tau)\sum_{j=1}^{\tau} d_t(j)}$$

3. Absolute Threshold
   Identifies a limit and a 'note' is the first dip to surpass the threshold.

4. Parabolic Interpolation
   Find the exact position of the dip.

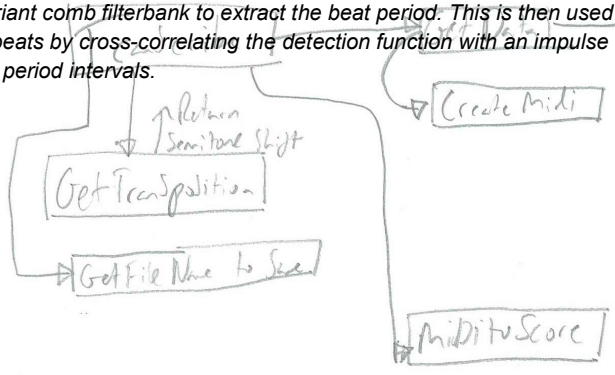
5. Best Local Pitch Estimate
   Identify Pitch at the dip.

- Set threshold to say 0.1
- Pick the first dip that exceeds the threshold



Spectral Analysis

The first stage in the beat tracking algorithm is the transformation of an input audio signal into a mid-level representation from which beat times can be robustly identified. We use the complex spectral difference onset detection function. A sequence of beat times ym is recovered by passing the autocorrelation function of the detection function through a shift-invariant comb filterbank to extract the beat period. This is then used to identify the phase of the beats by cross-correlating the detection function with an impulse train with impulses at beat period intervals.



4097 Oscar Lindenbaum 62113 Wheatley Park School

Computer Science NEA - NoteSwitch

In order to accurately notate the rhythm I will need to know the bpm of the audio signal, I will do this by using Spectral Difference D(m).

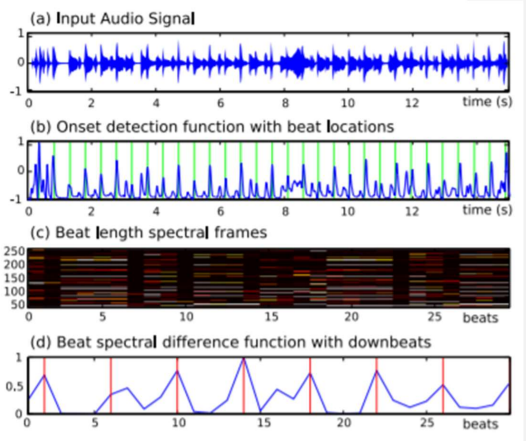
$$D(m) = \sum_{\omega=1}^{N/2} \hat{X}_m(\omega) \ln \frac{\hat{X}_m(\omega)}{\hat{X}_{m+1}(\omega)}$$

Given the number of beats per bar τ, we calculate a signal η(φ) as the measure of spectral change at each downbeat candidate φ = 1,...,τ

$$\eta(\varphi) = \sum_{m=1}^{M} D(\tau(m-1)+\varphi)$$

where M is the number of complete bar length segments. We then extract the beat leading to most spectral change φ_d as the index of the maximum value in η(φ)

$$\varphi_d = arg(\underbrace{max(\eta(\varphi)}_{\varphi})$$



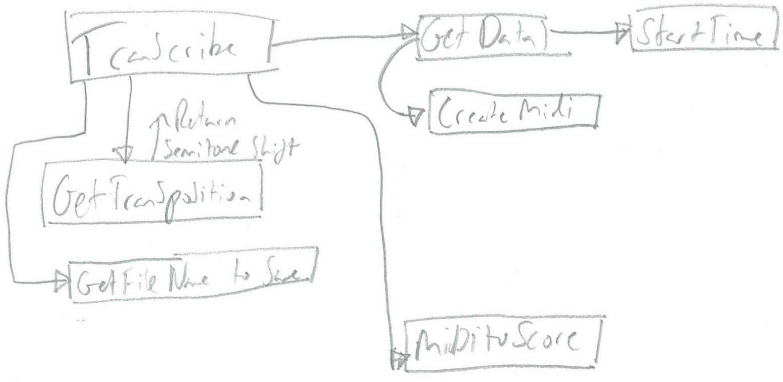
Assuming a steady tempo, we may then extract the downbeats γ_d from the beat indices γ_m, by setting

$$d = (m-1)\tau + \varphi_d$$

Source: https://www.eecs.qmul.ac.uk/~markp/2006/DaviesPlumbley06-eusipco.pdf

Creating MIDI Files

Taken from



http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.588.6600&rep=rep1&type=pdf

Standard MIDI Files

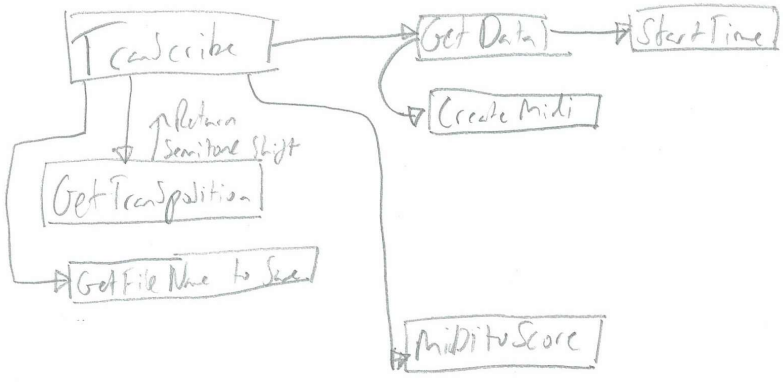
The specification of MIDI originally covered only the real-time connection between musical instruments. Later it was clear that the computer could control the instrument through a MIDI interface through programs called sequencers (they produce sequences of MIDI commands). The next step was a method of storing and exchanging MIDI sequences. The Standard MIDI File format [3] was developed to encapsulate MIDI sequences, and to include all the information concerning the performance of the sequence that was previously only implicit in the real-time nature of the MIDI stream.

The SMF format includes the following components:

- The track - similar to a track on an analog tape. There can be several tracks in the sequence that are intended to be played simultaneously.
- MIDI events - any command from the MIDI specification can be included in a standard MIDI file, along with a timestamp that indicates the time at which the command is to be issued.
- Meta-events, which include:
- Tempo, time and key signature changes.
- Lyrics and arbitrary text such as instrument names.
- Marker and cue points for rehearsal, and synchronization to other media, such as video.

In order to create a simple MIDI file I will need note values, note lengths, note onset times and note velocity. To create the file I can use a module created by Mark Conway Wir called MIDIUtil - https://midiutil.readthedocs.io/en/latest/index.html

MIDIUtil is a pure Python library that allows one to write multi-track Musical Instrument Digital Interface (MIDI) files from within Python programs (both format 1 and format 2 files are now supported). It is object-oriented and allows one to create and write these files with a minimum of fuss.



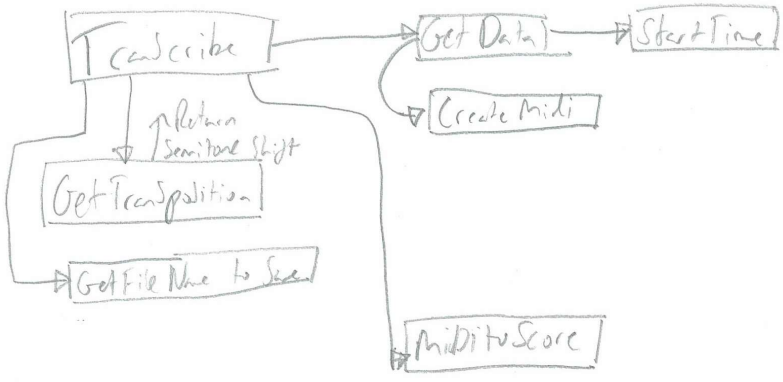
Scenarios

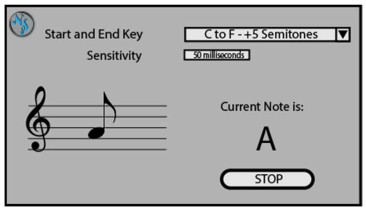
- The user has composed a song on their instrument and has been asked to write it down, by using the 'Transcribe' functionality of the program they can simply play their composition piece into the program and the program will notate it for them and give them a PDF score.

- A composer is writing a piece for a Saxophone quartet but is creating it on an Alto Saxophone (Key of E flat). They want to quickly create the parts for each other type of saxophone e.g Baritone, Tenor (Key of B flat) they could do this by playing single notes into the program and the counterpart in the other key will be displayed on the screen.

- The same scenario as above but the composer wants to create harmonies within the parts, they could do this by selecting the transposition of (+3 semitones for Major triad or otherwise for different triads) and the displayed note would be a part of this harmony.

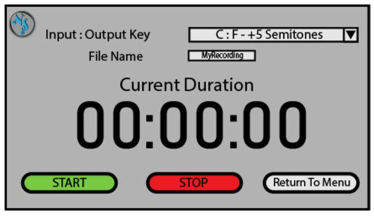
- A person finds a solo for a piece written for the Mellophone but wants to play it on their Flute they would go to the transcribe option and select the starting key as the key for the Flute (C) and the ending key as the number of semitones difference between the Mellophone (F) to the Flute (C) which is -7 Semitones. Then when they play the piece as written for the Mellophone into the program they will receive a transposed part in the key of C for the Flute.





Transcribe

- We are transcribing from C to F (Flute to Mellophone)
- Sensitivity isn't too long so the note refreshes often.
- We play an E on the Flute and the output should be an A as E shifted +5 Semitones is A. This is the note for the Mellophone in F.
- When we are done we can press the stop button, the program stops transposing and we can change any settings needed.



Transpose

- We select our transposition, in this case from C to F or Flute to Mellophone.
- We input our file name for our saved score file. In this case 'MyRecording'
- We press start and then start playing, in this case we are playing happy birthday. Score in C below (Fig 1)
- After pressing stop a file is saved as MyRecording.pdf (see Fig 2)

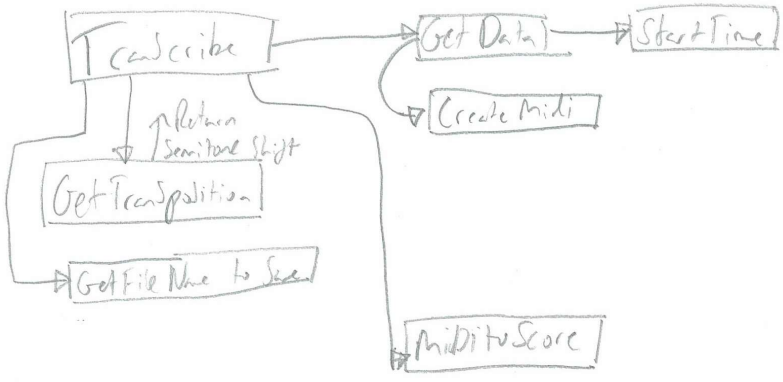


Fig 1.

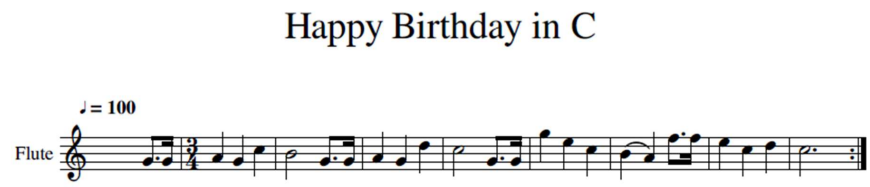
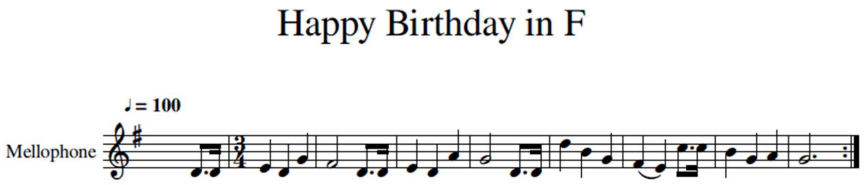
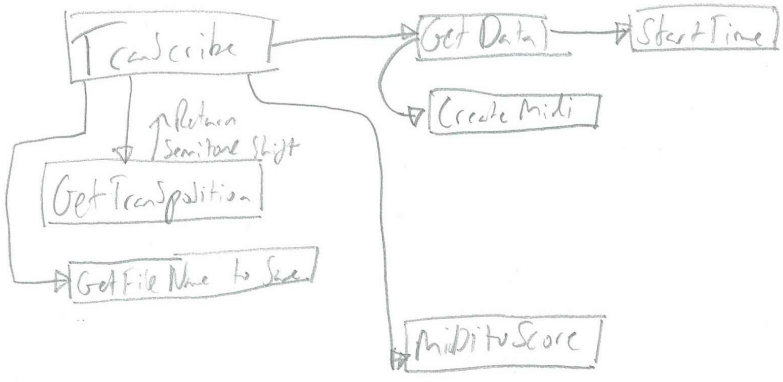


Fig 2.



End User Research

In order to find out in more detail about what music students would use the application for and what they feel it is important to get right I took a survey from a local Band of students.
See appendix i.
Additionally, I sent out a Google form to a multitude of students to see how useful the application would be to a typical student.
See appendix viii



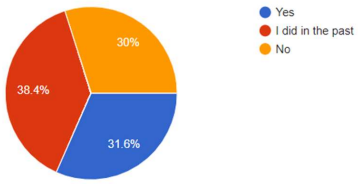
4097 Oscar Lindenbaum 62113 Wheatley Park School

Computer Science NEA - NoteSwitch

General Student Research

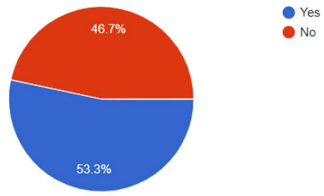
Do you play an instrument?

237 responses



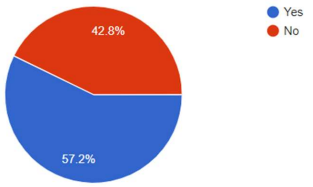
Do you play more than one instrument?

75 responses



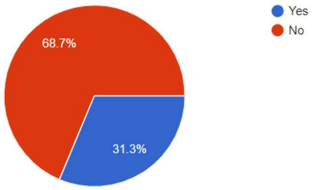
Can you read sheet music?

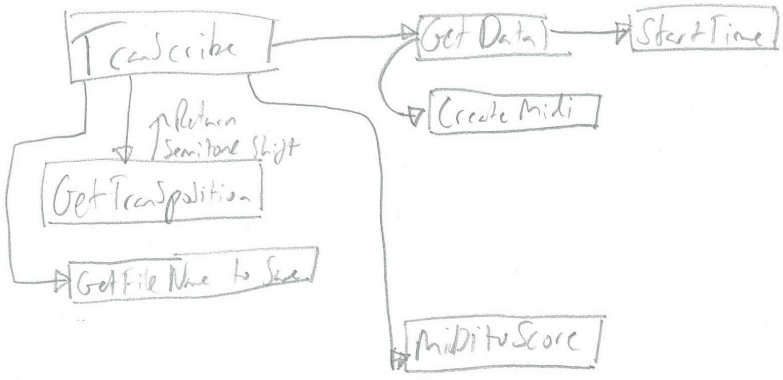
166 responses



Have you ever had to notate a musical rhythm by hand?

166 responses



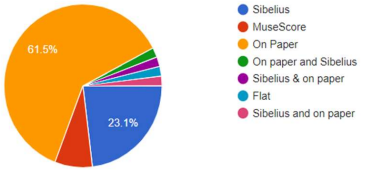


Computer Science NEA - NoteSwitch

Overall the results from the google form are surprising with 70% of students having played or currently play an instrument, and of those over 50% play an additional instrument. However only 31.3% of these have ever transcribed music themselves, possibly because it was too hard to, this shows a use for the application. From the students who have notated a rhythm they all seemed to have used either sibelius or on manually wrote it out, which I found was confusing to use from my band student interviews.

How did you notate it?

52 responses



Band Student Research

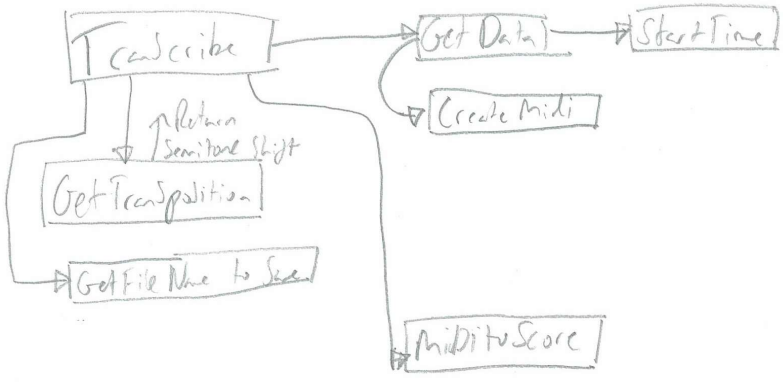
COMMENTS IN BOLD

- *<u>If applies:</u> What do you find are the difficulties of playing multiple instruments?*

- Different fingerings between similar instruments
- Switching between clefs
- Differences in keys (A on trumpet is not an A on Violin)

There is a need for a program to solve switching between instruments, an additional objective could be to provide fingering charts for each note depending on the instrument, this would only require a database of the fingering and could be easily implemented.

- *What do you feel are the most important things to get correct when notating music?*



- Rhythms and Pitch accuracy
- Articulation
- Key
- Time Signature

The program will be able to notate the rhythms and pitch accurately and selecting the output key will be a part of my solution. Given more time I could notate the articulation as well but this will be much harder to specific as the whole track would have to be considered to make a decision on what is staccato or forte ect. This would also take much more processing power to attain good, real time results.

- *If you have had to notate music in the past, what did you use and how easy was it?*

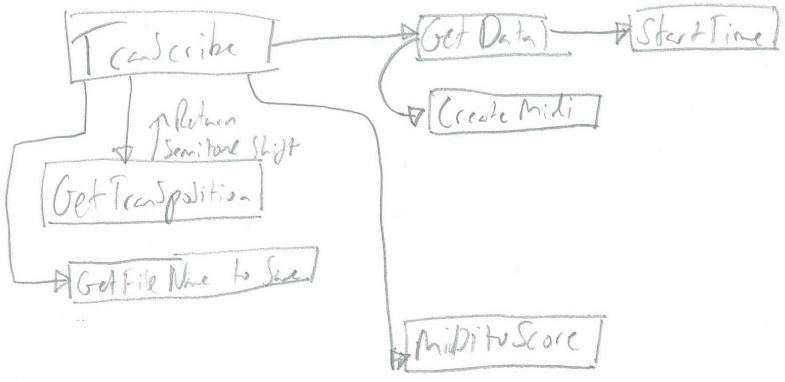
- Sibelius
  - Horrific
  - Fairly easy
- PhotoScore
  - Confusing
- Logic Pro
  - Writing the piece in the correct key was time-consuming

Most people use Sibelius and have mixed opinions with the software, I will look into these programs and see what makes them hard to use.

- *What additional functionality or features, if possible, would improve the proposed application?*

- Scan scores into the program
  - Possibly Recognise the piece
- Play the score file with MIDI instrument
  - Change speed of the piece
- Select the desired octave for the notes

Users would benefit from all these features, given the time period the feasible features to add would be to limit the octaves for the notes and also have playback for the scores natively these would be an additional objective. For scanning the scores into the program this could be possible just to upload to the database for future use but a MIDI file would need to be created from the score file which could be implemented if I had more time.



- *What are the most important settings that would need to be changed?*

- How sensitive the pitch and rhythm detection is
- Desired Output key

Changing the output key is a part of my solution so it shows a need for the program. Adjusting the pitch and rhythm detection settings could be added later but as a low priority objective but it is feasible.

- *The proposed style for the GUI is minimal with very few buttons, does this style fit with the application? If not what could be changed?*

- All feedback was positive and supported the UI
- Looks simple and easy to use
- Simple to use for musicians who may not be up to date with the latest technology

The Design of the GUI is minimal enough to make sense to a large audience but explicit enough for the more advanced user to understand.

- *Do you know the key of your instrument? If so would you be able to identify the key of others when given a name? e.g What Key is the Flute?*

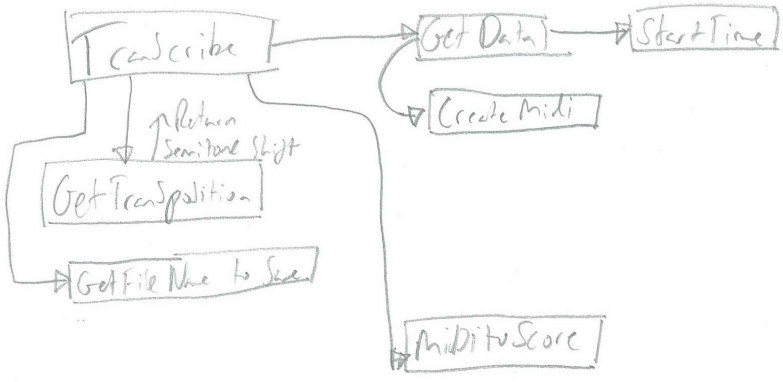
- All responses showed knowledge of instruments being in different keys, one response said the flute was in A which is also correct as there is an A and a C flute.

This would make it a good idea to list the Keys rather than instrument name.

- *Can you easily read sheet music in Treble Clef? If not what sheet music can you read?*

- All responses said they could read Treble.

It isn't necessary for the application to notate the score in a specified clef as all research shows that understanding of Treble Clef is consistent across musicians. Maybe add implement this feature as a low priority objective.



4097 Oscar Lindenbaum 62113 Wheatley Park School

Computer Science NEA - NoteSwitch

Research

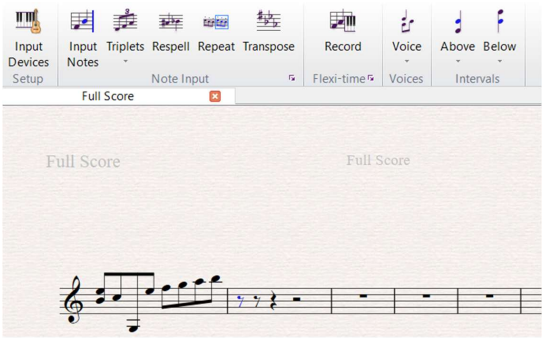
What do people currently use for music notation?

The Current Market

Currently there are applications that can tune your instrument, show how to use the circle of fifths (To aid transposing) and programs such as Musescore an application that lets you notate music by hand, but there are no apps which will take your live audio feed and notate that for you or transpose a piece of music into the desired key automatically - in order to do this you would have to use multiple apps and would require previous knowledge additionally to a lot of time.

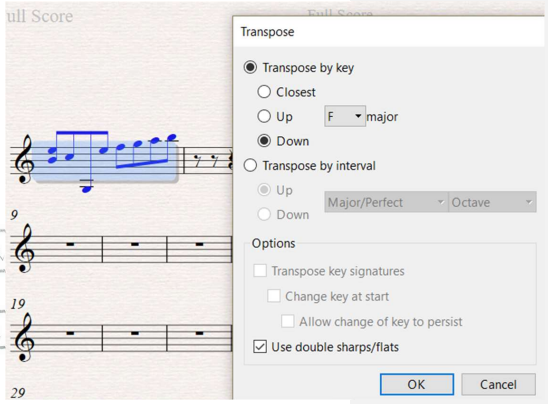
Sibelius

In order to understand the pros and cons of Sibelius I downloaded the free version *Sibelius | First.* Sibelius has 3 versions: Sibelius | First, Sibelius and Sibelius | Ultimate. Each version has a different capability of what it can do with the First edition being the most limited, as it is free. Sibelius costs £99 and £499-£750 for the base version and the Ultimate version respectively.

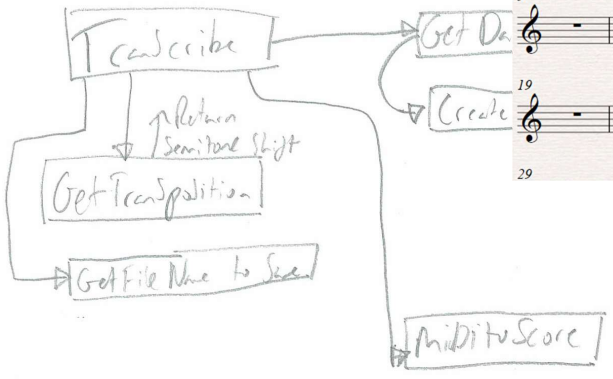


Initially the interface is very clean and simple, upon notation I am able to do this with ease because of my music knowledge, for someone with less knowledge this could be quite difficult knowing what each symbol means.

When I needed to transpose the music into another key this was easily done by selecting the transpose prompt and the destination key.



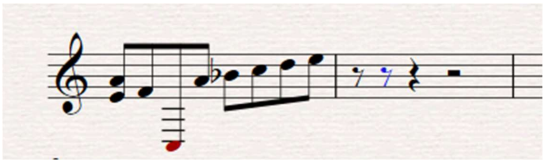
By using the Free edition of Sibelius this is about all the program can do for me without spending a lot of money for the additional features such as PhotoScore or AudioScore.



4097 Oscar Lindenbaum 62113 Wheatley Park School

Computer Science NEA - NoteSwitch

Overall I think that the software is easy to use for experienced musicians but not for anyone else, the price point of Sibelius is also a factor as it is very steep, reducing its possible market. What I think it got right is how simple the interface needs to be so that the composer doesn't get lost while notating. Sibelius also has a feature where you can listen to the playback of the score, this is a feature I think would be beneficial to the user and would allow for any mistakes to be heard as well as spotted.



PhotoScore

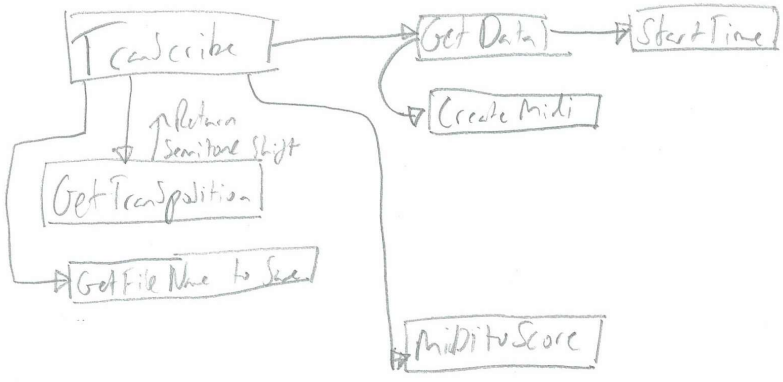
Photoscore is an extension to Sibelius Ultimate edition and is made for *converting printed and handwritten music into professional scores.*

Upon research there is a free version available but likewise most of the features will be locked behind payment. Whilst I think this program would aid the notation of pieces from old scores it doesn't notate from live music like AudioScore. What is a feature that I could add to my solution is the scanning in of sheet music for the server and then the new PDF generated from it.

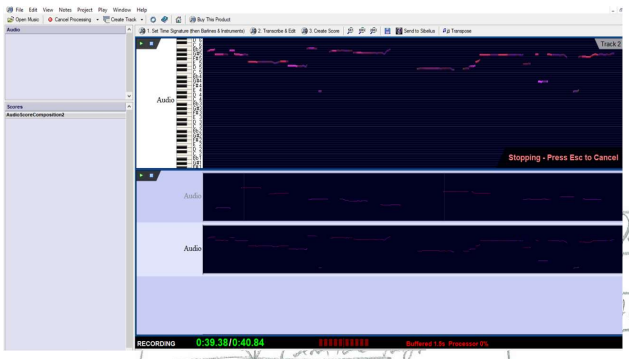
AudioScore

AudioScore is another extension to the ultimate edition and it enables you *to turn recorded audio or a MIDI or live mic performance into transcribed music notation*. Unfortunately AudioScore standalone costs £209 but they do have a free trial to use which I am going to base my research on.

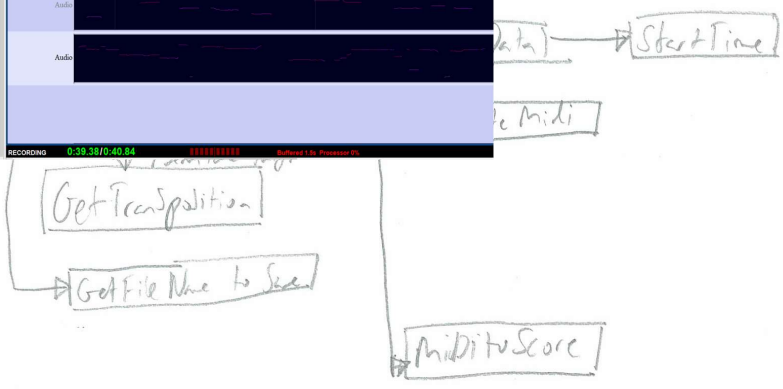
	PhotoScore & NotateMe Lite	PhotoScore & NotateMe Ultimate		PhotoScore & NotateMe Ultimate
Shortest note value	16th note	128th note	**Automatic scanning and reading after scanning**	+
Accidental types	3	7	**Printing**	+
Clef types	2	8	**Transposes in PhotoScore itself**	+



Maximum voices per staff	2	4	**Reads slurs and ties**	+
Maximum staves per page	12	64	**Reads hairpins**	+
Maximum pages per score	20	400	**Reads text (including bold, italic, etc.)**	+
Maximum dots per note/rest	1	3	**Reads articulation marks**	+
Handwritten music recognition		+	**Reads triplets and other tuplets**	+
Bad timing navigator		+	**Reads grace notes and cue notes**	+
Rescore feature		+	**Reads cross-staff notes and beams**	+
Find and Replace feature		+	**Reads guitar chord diagrams**	+
Reads the length of irregular bars		+	**Reads 4-string guitar tab**	+
Saves WAV/AIFF audio files		+	**Reads 1-, 2- and 3- line percussion staves**	+
Saves MIDI files		+	**Reads double and repeat barlines**	+
Reads repeat endings, Coda, and Segno		+		
Reads ornament and pedal markings		+		

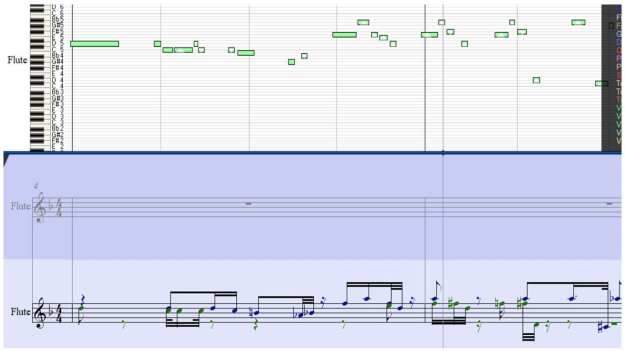


Upon first use of the AudioScore Trial the interface sees minimal once again but this time less interesting and obvious what to do.

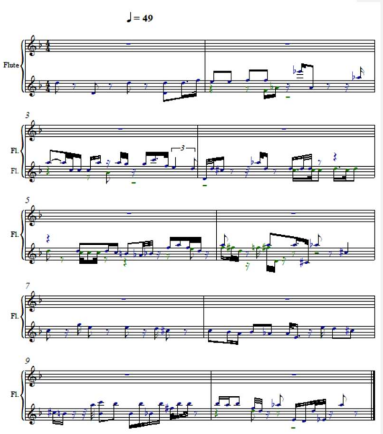


Computer Science NEA - NoteSwitch

The program then allowed me to manually view and edit the MIDI Piano Roll for any corrections. This was very useful because as you can see it created a very precise, nearly too precise score.

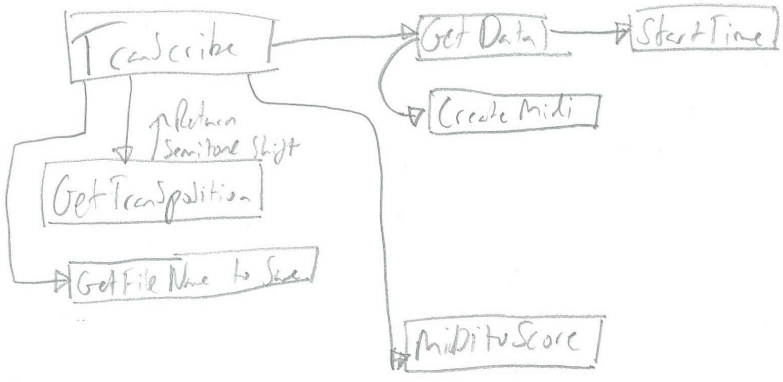


Afterwards it created a score, what I wish it did was allow me to



edit the sensitivity so it didn't pick up any accidental background noise and ruin the notation. What the program failed to do was notate any of the dynamics such as staccato or crescendos, this is a feature only in the premium version which is a very important part in music. Overall this is a very good solution to the problem but the price point is very high once again which means certain features will be hidden until you pay or the trial expires.

Logic Pro

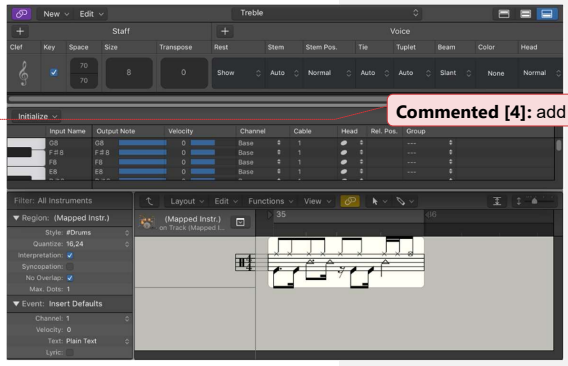


4097 Oscar Lindenbaum 62113 Wheatley Park School

Computer Science NEA - NoteSwitch

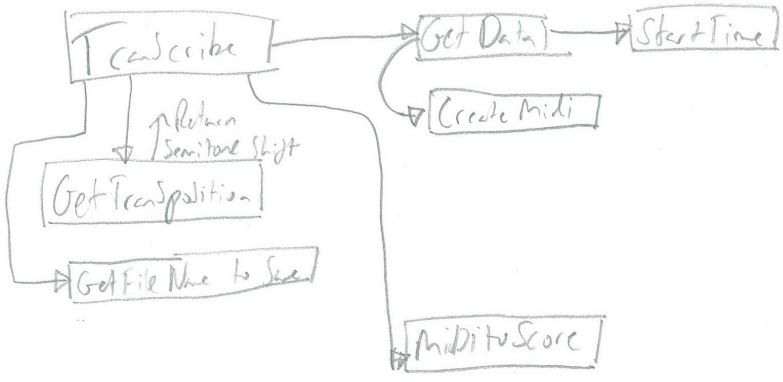
Logic Pro is *a digital audio workstation and MIDI sequencer software application* for macOS, immediately an issue with it is that it is only for macOS not windows which means I cannot download and test the program myself but I have previously used it for creating orchestral compositions. Logic Pro is very powerful but hard to use because all there is a lot going on and specific functions are hidden away, it is also not made for notation but more for creating brand new music.

Because it is not made for notating music like Sibelius adding in dynamics or particular rhythms is not easy. As pointed out by Carmen Powell in my user research, it is very time consuming to change the key which would make transcribing for different instruments very cumbersome.



Logic Pro is also steeply priced but less so coming in at £199.

Commented [4]: add more



What data am I expecting and What do I need to get from it?

The main data that my program will take is the raw audio for creating the score files. The program will create a WAV file with a Sample Rate of 44100Hz, One Channel and a resolution of 32 bits. From this WAV file I need to extract the information of each notes location, duration, velocity and pitch I also need to detect the tempo of the audio. I will do this by using the YIN algorithm to obtain the frequency and its onset times. Using this data I can create a MIDI file, which can then be converted into a PDF Score file.

When I need to upload the file to the server I need to get parameters to uniquely identify these files, the database contains two tables:

ANDROID3.mainDB - dbo.info

Column Name	Data Type	Allow Nulls
[User]	ntext	☐
UploadDate	ntext	☐
UploadName	ntext	☐
uploadID	int	☐
dropboxID	int	☐
		☐

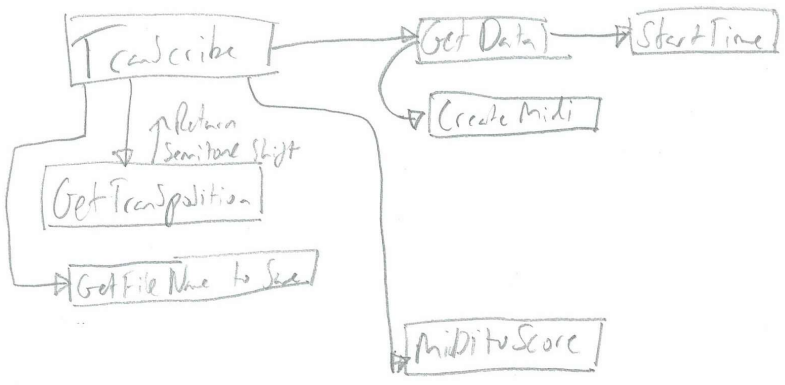
Solution1 - ANDROID3.mainDB - dbo.download

ANDROID3.mainDB - dbo.download

Column Name	Data Type	Allow Nulls
dropboxID	int	☐
MIDI_URL	ntext	☐
Score_URL	ntext	☐
		☐

The table called download is used to store the different urls to access the MIDI and PDF files, the other table is for the information about that file such as the name and date of when it was created. This server allows the user to search through past records by any search type and then to download the files associated with that entry.

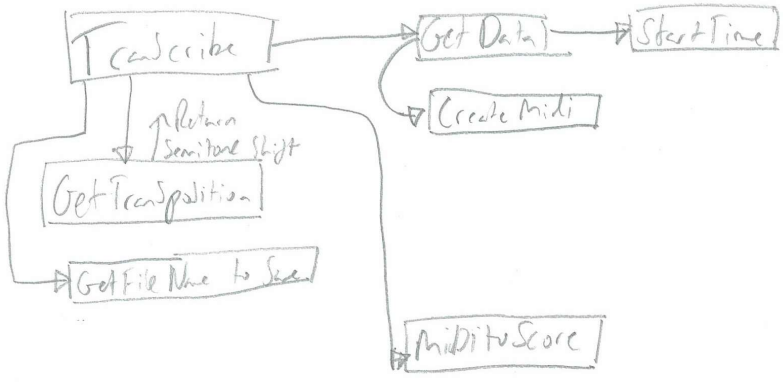
In the case the user is transposing they will only see the transposed note and its musical notation. In the other scenario when the user is transcribing they will only be able to enter a name and then after they are finished they can find and view the outputted score file. When the user wants to download or search the database they will see the output of the SQL search in a table determined by their parameters. Then if they want to download them they will receive a prompt saying the success or failure of the download. Then they can view the files.



Why is my solution better?

My solution is better because it will be aimed at students using it, rather than professionals. This is because the user doesn't need to know anything apart from what they want to do e.g "I want to transcribe this solo for clarinet" and after that they only need to press start and play the solo into the program. Also if they then want to edit it at a later stage the capability is already implemented. My solution also contains no fancy settings to edit or unnecessary windows like in Logic Pro or Sibelius. The application will also allow students to share their work with each other and their teacher - allowing them to come back to them at a later stage or for marking which would be beneficial for the teacher. The end user and the client would want this program as it would speed up and improve their learning, allowing them to have more freedom with their compositions (Being able to notate anything not only the simple rhythms which they can physically write). Additionally if they were a more advanced student it would speed up working out harmonies and other parts when writing music for a quartet or small band.

.



Client Requirements

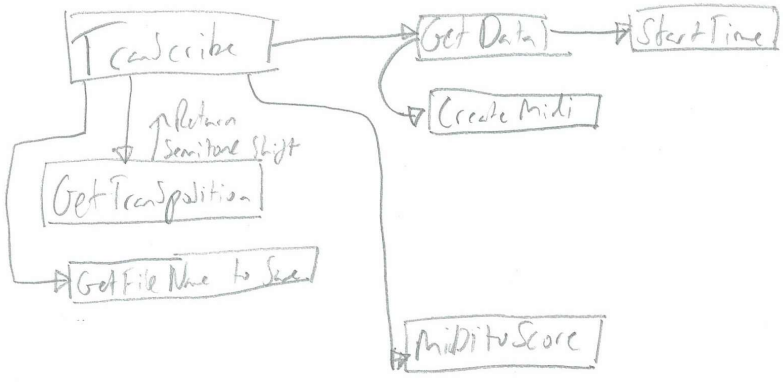
Name: Michael Ahmad - Music Teacher

- *On average how many students per class play multiple instruments?*

At KS3 i would say 2-3 out of 30 (<10%) At KS4 the proportion is higher (~20 - 30%) At KS5 the numbers are too small to have significance

As there are little amount of students that would benefit from this application at schools, the application would be suited for more advanced musicians as the numbers increase as they get older.

- *What do you feel are the most important things to get correct when notating music?*



To a certain extent the answer to this question is dependent on the style of music being notated. The system of notation developed to cater for western classical music. Rhythmically this is fairly straightforward to notate (although expressive changes of tempo may complicate things). However, all kinds of popular music are far harder to notate having being influenced by musical traditions that do not have systems of notation - the most important being the influence of - West african music. Having accurate note durations would be a really good feature of any notation software!

What this tells me is that for music to be appropriate and the best it can be it needs to be written for a certain style for example how to notate sounds which might not be set to a note value. Given the time period of this project this is going to a very low priority project but could be introduced through AI by showing the program different styles of music and more common rhythms. Another point that the client makes is that the rhythms being notated are very important so this will become a key testing point.

- *If your students have had to notate music in the past, what did they use and how did they find it?*

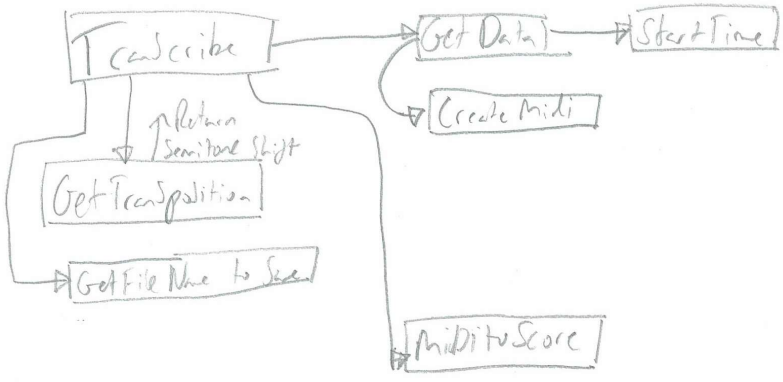
Most often they use Sibelius and input notes in step-time. Some however chose to play in to a sequencer such as Logic or pro tools and then export MIDI information to Sibelius or other notation software such as Musescore. Such exports are dependent on note durations being quantized accurately against a click so that the note on and note off info corresponds to regular musical note values. This was tricky for many students.

My application will remove the necessity to play the rhythms against a click and on the MIDI input device, it would allow the student to play whatever instrument is their preference is straight into the program. This would make it easier for all students regardless of their level, showing a need for my program.

- *What additional functionality or features, if possible, would improve the proposed application?*

Easier, more flexible quantize features
Enharmonic accidental recognition - i.e tonality and Key recognition

Within the timescale it will be difficult to implement key recognition but would be a good feature to add in later. In order to make sure the program gets the correct key to



write the music in the users can enter the starting and destination keys manually.

- *What are the most important settings that would need to be changed?*

If you are going to notate rhythms it is necessary to have a *1) 'click' function (metronome) and then there needs to be a* *2) 'metre' function (time- signature, 4/4 ¾ 2/4 6/8 9/8 etc)* *3) 'Tempo' function with BPM counter - changing the tempo to play in time with the click*

The program does not require any click or thinking from the user but adding in an optional click could be another feature to add in at a later stage. Tempo and Metre detection will also be implemented.

- *Is the Score a useful output?*

Yes

- *What other forms of data would be useful?*

MIDI data

- Should the students be able to upload and share their work with each other and yourself?

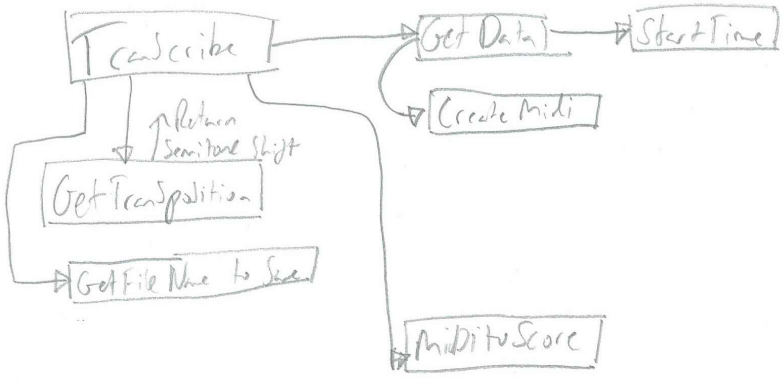
Yes

All of these answers confirm that the output of the program is useful, the users can access all file types on the server and download them for other use.

- Can the students easily read complex rhythms?

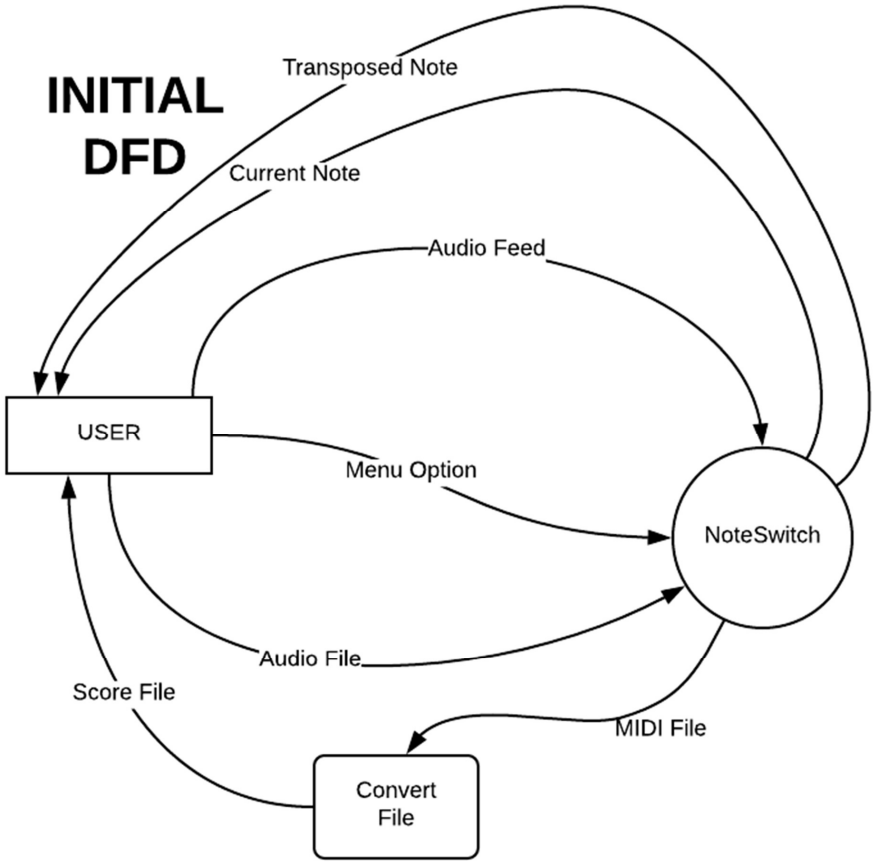
Quite often not

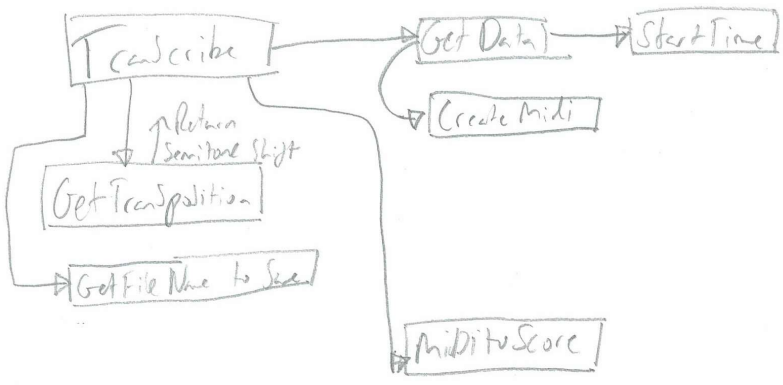
This answer raises questions about how important the accuracy of the rhythm notation is, if it is too accurate the users won't be able to read the rhythmic output easily or if the accuracy isn't high enough does the program serve its purpose? In order to settle this I will only allow the program to notate notes up to 1/32nd of a beat long so there is enough accuracy but not too much.



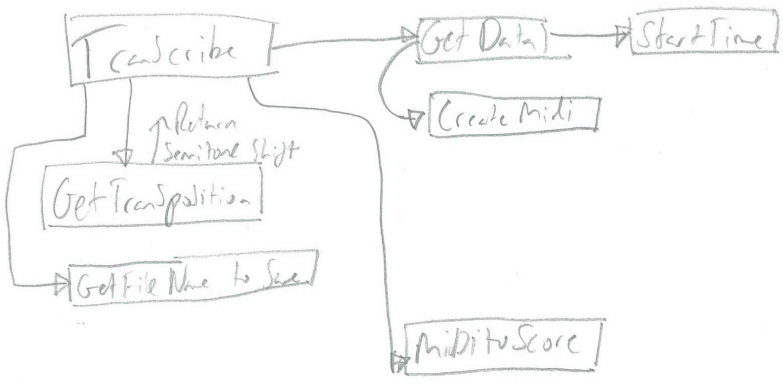
Initial DFD

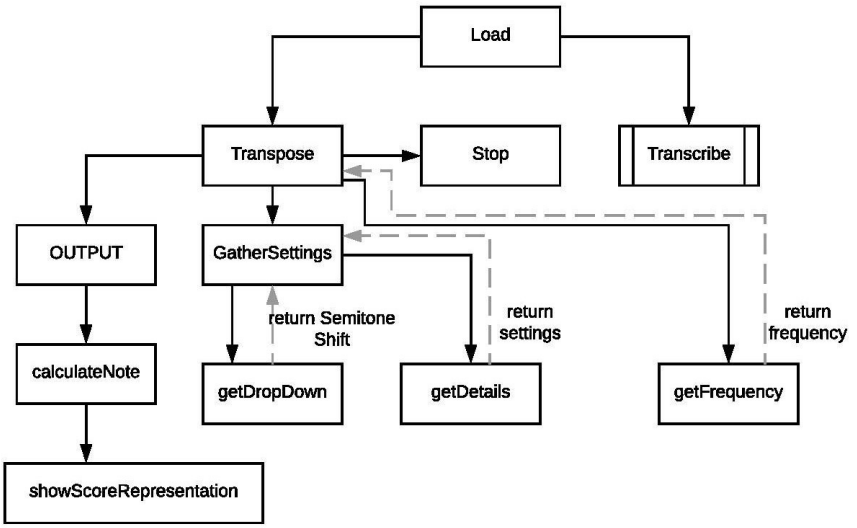
Commented [5]: make white text so can be in contents

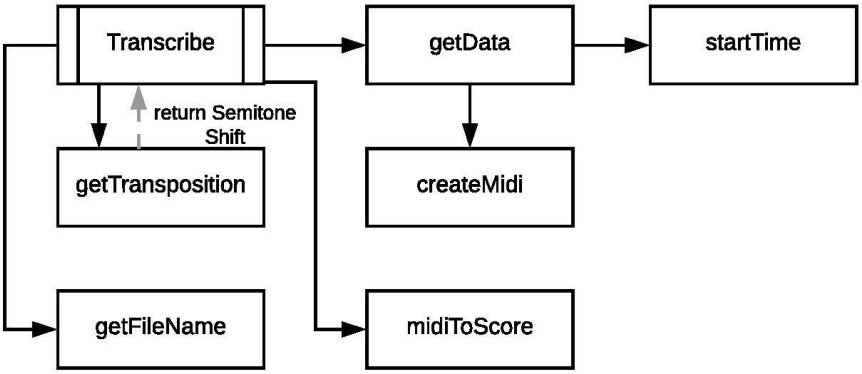


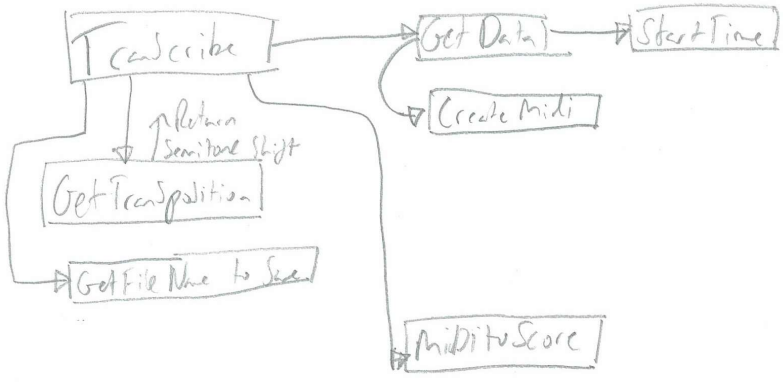


Initial Hierarchy Chart



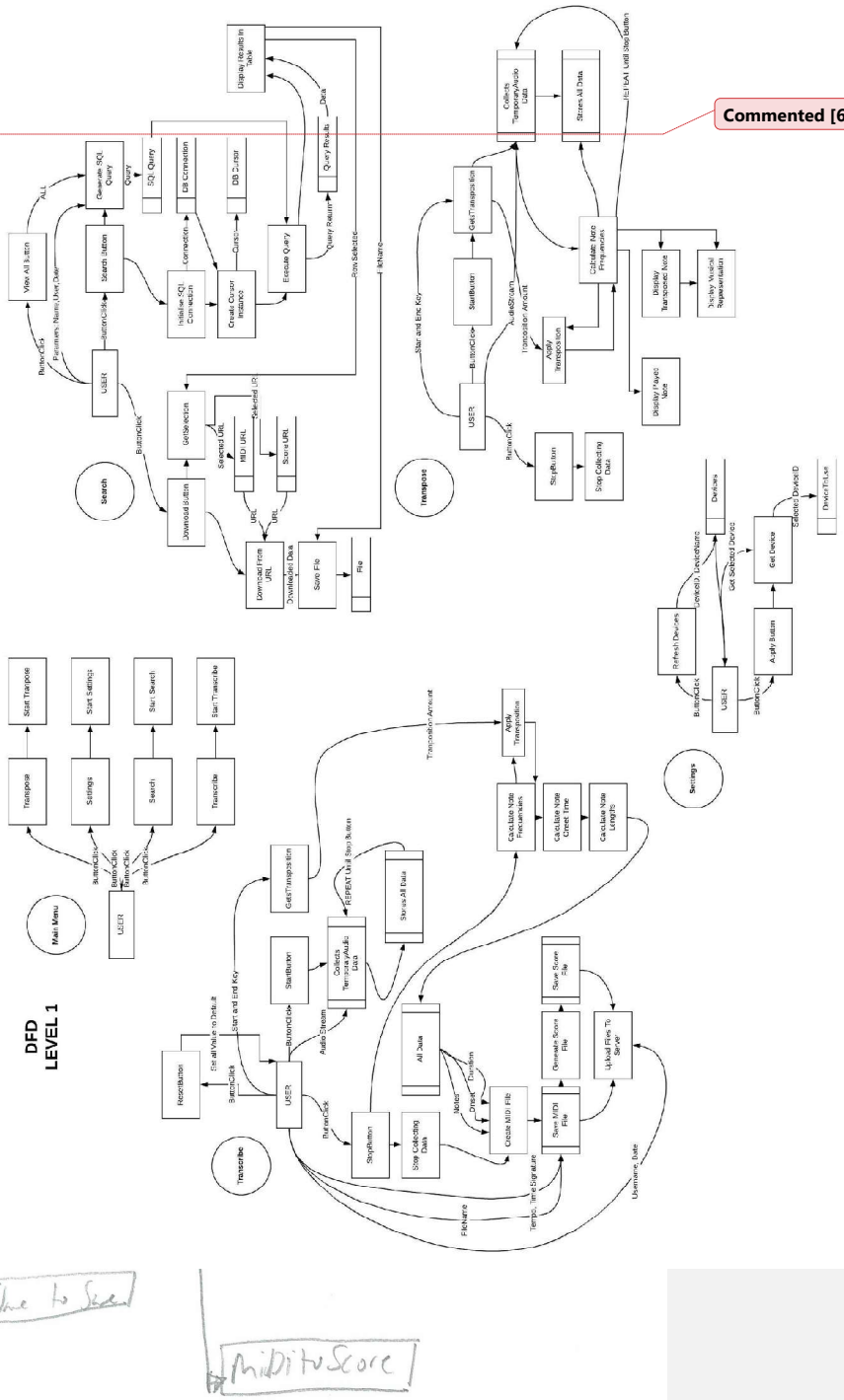

Load
Transpose
Stop
Transcribe
OUTPUT
GatherSettings
return Semitone Shift
return settings
return frequency
calculateNote
getDropDown
getDetails
getFrequency
showScoreRepresentation


Transcribe
getData
startTime
return Semitone Shift
getTransposition
createMidi
getFileName
midiToScore

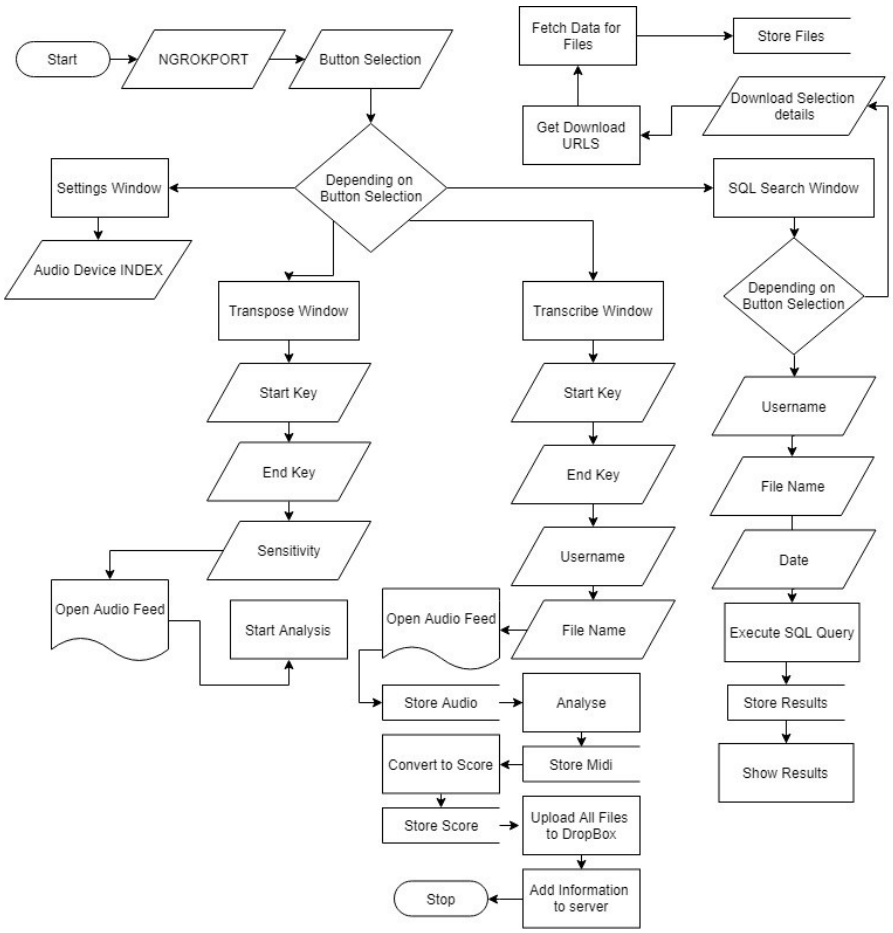

Transcribe
Get Data
Start Time
Return Semitone Shift
Create Midi
Get Transposition
Get File Name to Scan
MidiToScore

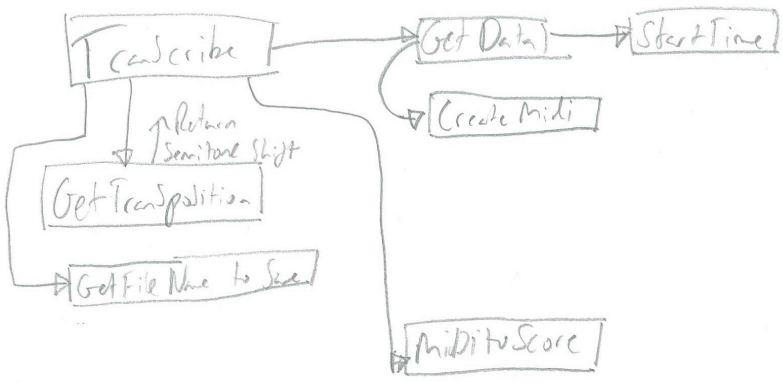
DFD LEVEL 1

Commented [6]: Make white



System Data FlowChart



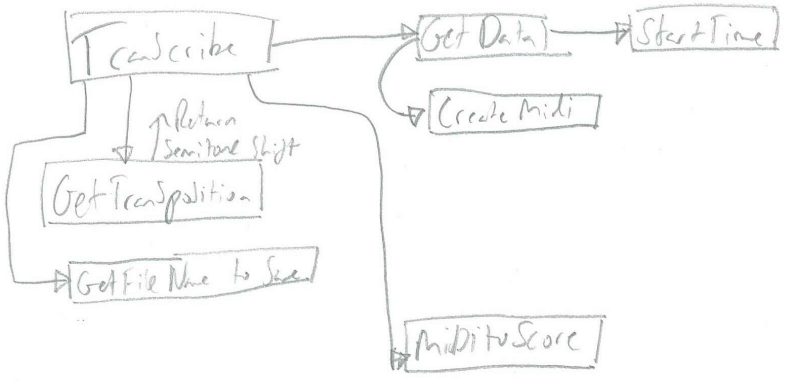


Objectives

Critical Objectives are highlighted in bold

No.	Main Window
-	ESSENTIAL
1.1.1	**The user is presented with a selection of buttons, “Transcribe” and ”Transpose” which opens the correct window.**
-	DESIRABLE
1.2.1	The user can select a “Search” button which opens the server and allows them to search through the database for past scores and recordings.
-	OPTIMAL
1.3.1	The user can open the “Settings” window, this would allow the user to select the audio devices manually.
1.3.2	The program has an additional function for tuning instruments.

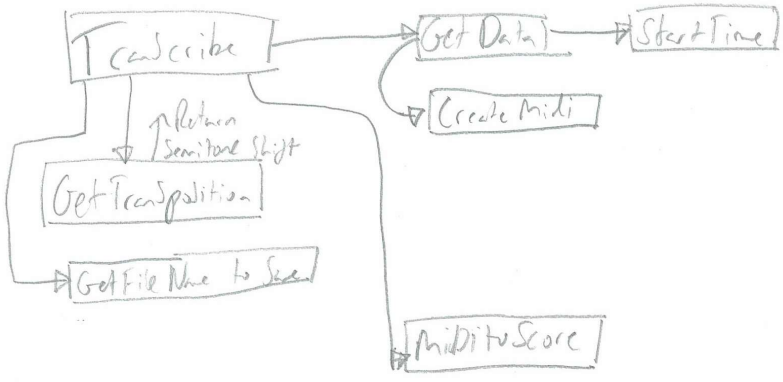
No.	Transpose Window
-	ESSENTIAL
2.1.1	**The program can successfully identify the frequency and the correct note so it can display the correct note.**
2.1.2	**The user can select the transposition value and this changes the transposition output of the program.**



2.1.3	**The correct transposed note is displayed for the user as text.**
-	DESIRABLE
2.2.1	The correct transposed note is displayed for the user as a musical representation.
2.2.2	The user can change the sensitivity of the note detection (The margin of *cents* determining which note is being played) *One semitone is equal to 100 cents.*
-	OPTIMAL
2.3.1	The user can pick from a list of instruments which has the key already paired with them so that they don't need to find out the key of their instruments.

No.	Search Window
-	ESSENTIAL
3.1.1	**The user can view past scores and MIDI files.**
3.1.2	**The user can download the score and MIDI files for future use.**
-	DESIRABLE
3.2.1	The user can search through all of the scores with different parameters.
-	OPTIMAL
3.3.1	The user can select an entry and 'preview' the files without downloading them.

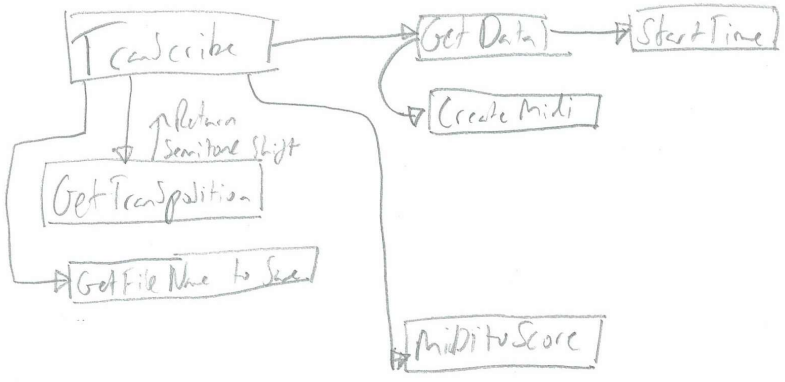
No.	Transcribe Window
-	ESSENTIAL
4.1.1	**It can create a midi representation of an audio file or live audio stream.**
4.1.2	**It can convert the generated midi file into a musical notation such as a score which is outputted for the user.**



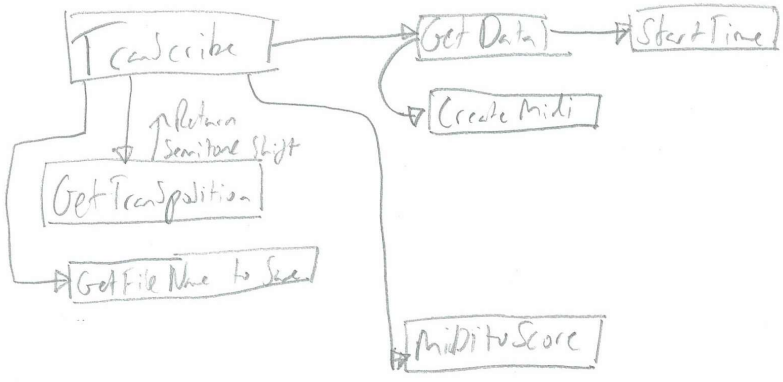
-	DESIRABLE
4.2.1	The score has the correct quantized note lengths.
4.2.2	The program can be changed so that the user enters the instruments starting key and the desired end key which then changes the transposition of the notes.
4.2.3	The score has the best 'spellings' of notes displayed i.e replacing double sharps with a flat so it is easier to read.
4.2.4	The program outputs both the score file and the midi file so it can then be edited in other programs such as MuseScore for corrections.
4.2.5	The program can detect the Tempo automatically.
4.2.6	The program can detect the time signature automatically.
-	OPTIMAL
4.3.1	The user can pick from a list of instruments which has the key already paired with them so that they don't need to find out the key of their instruments.
4.3.2	The program can record multiple parts for a larger composition and merge all of the scores together for a conductor's score.

Data Dictionary

Data	Source	Type	Validation	Sample Data
Live Audio Feed	Microphone	Audio	None	
Button	User	Input	None	'ButtonClicked'
Note Frequency	Live Audio Feed	Integer	In the correct format?	440
Transposition(semitones)	Drop Down Box (Start and End Key)	Integer	-12<=Value <=12	+4

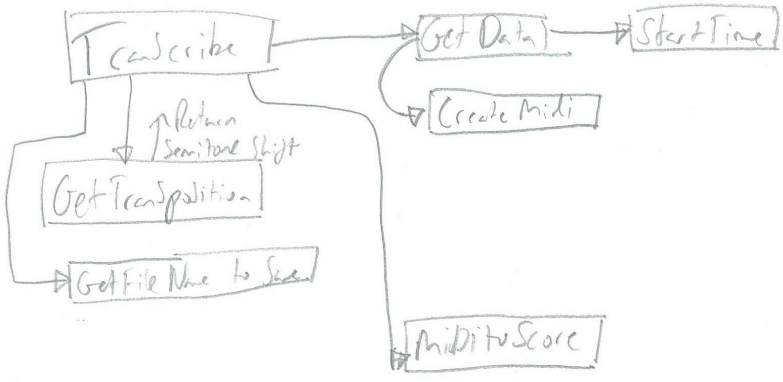


Note Onset Times	Live Audio Feed	Float	Time is greater than the start time, correct format	5.4623466
Note Off Times	Live Audio Feed	Float	Time is greater than onset time, correct format	6.231156
Note Duration	Live Audio Feed	Float	Value is rounded to 2dp and is positive.	1.51
Note Amplitude (Volume/Velocity)	Live Audio Feed	Float	Greater than ‘noise’ level	1
MIDI File	getData()	File Type	File Structure is correct so it can be converted to Score.	Standard MIDI Structure
Score File	MidiToScore()	PDF/JPEG	The file has size >0kb	abc.pdf
Drop Down Box (Start and End Key)	User	Input (Returns Transposition Semitone)	An option has to be chosen before Audio is listened to.	Dropbox Selected Item: IDX 1
Sensitivity	User	Integer	Greater than 0 and non-negative	43
Overall Duration	Live Audio Feed	Float	Rounded to 3dp and is in the same format as onset times	27.224



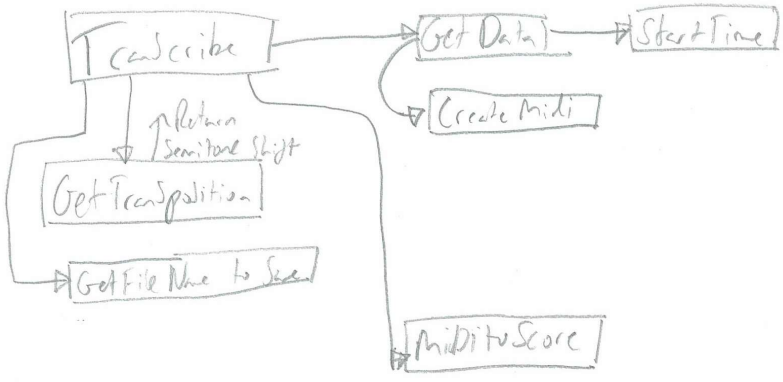
File Name	User	String	File Doesn't exist and doesn't contain "NUL, \, /, :, *, ", <, >, |"	'testFile'
File Location	User/Directory of Program	String	Location exists and is writable	C:\\DIR\\SUBDIR
Note Table	Note File	Database	PRESET	A,A#,B,B#,C#,D...
Frequency Table	Frequency File	Database	PRESET	16.35 17.32 18.35 19.45 20.60 21.83 23.12 24.50 25.96
'Noise' Level	PREDEFINED	Float	PRESET	-50
CHUNK Size	PREDEFINED	Integer	PRESET	1024
Sample Rate	PREDEFINED	Integer	PRESET	44100

Data	**Type**
Live Audio Feed	Audio
Button	Input
Note Frequency	Integer
Transposition(semitones)	Integer
Note Onset Times	String
Note Off Times	String
Note Duration	Float



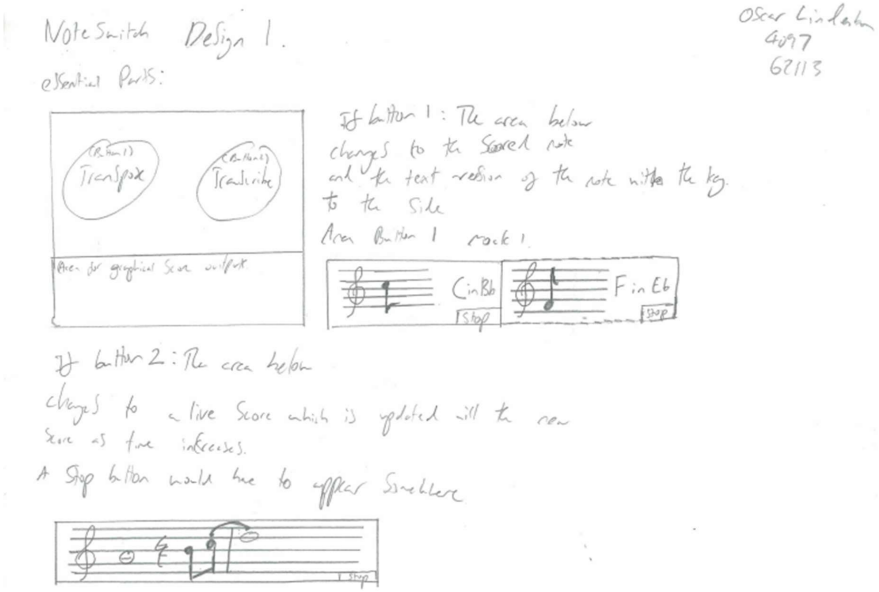
Computer Science NEA - NoteSwitch

Note Amplitude (Volume/Velocity)	Float
MIDI File	File Type
Score File	PDF/JPEG
Drop Down Box (Start and End Key)	Input (Returns Transposition Semitone)
Sensitivity	Integer
Overall Duration	Float
File Name	String
File Location	String
Note Table	Database
Frequency Table	Database
‘Noise’ Level	Float
CHUNK Size	Integer
Sample Rate	Integer

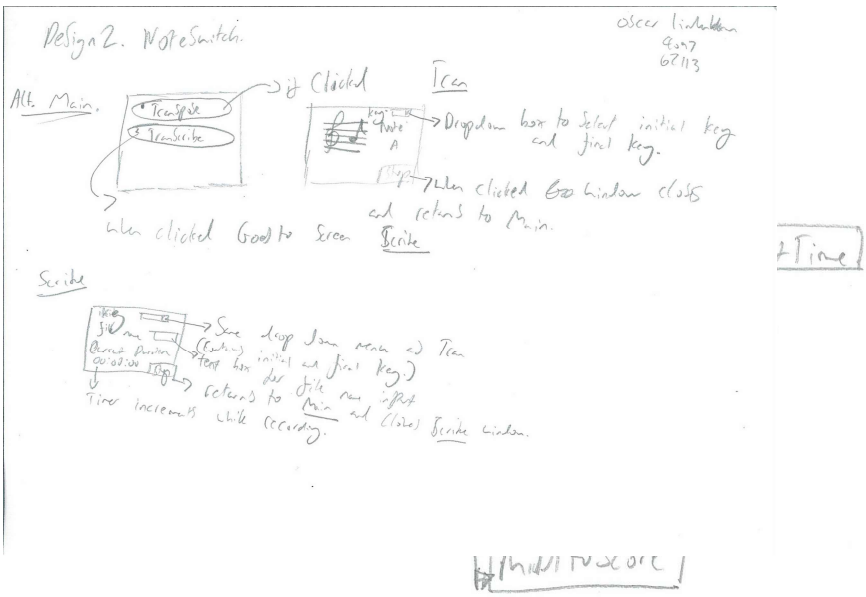


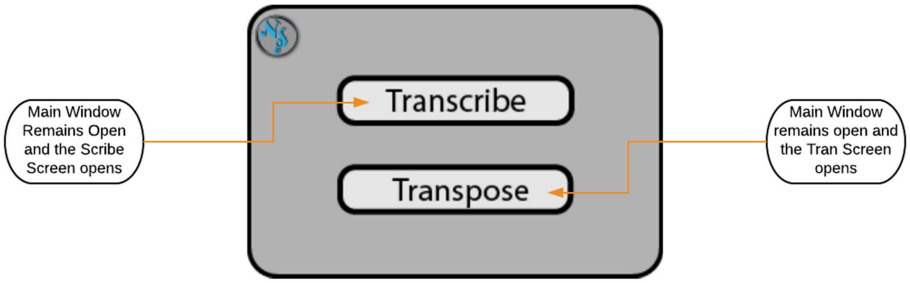
Design

GUI Design

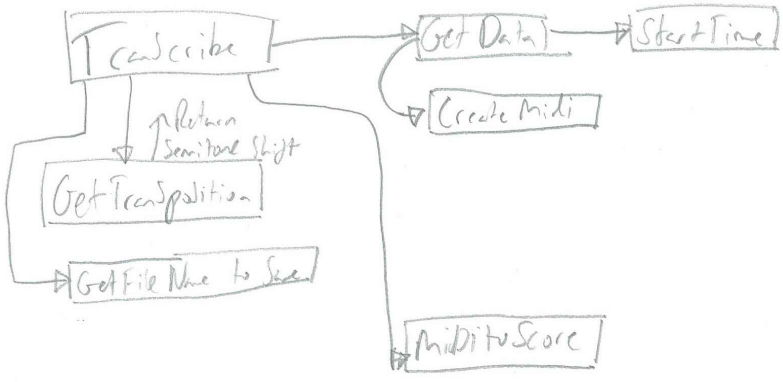


Main Screen

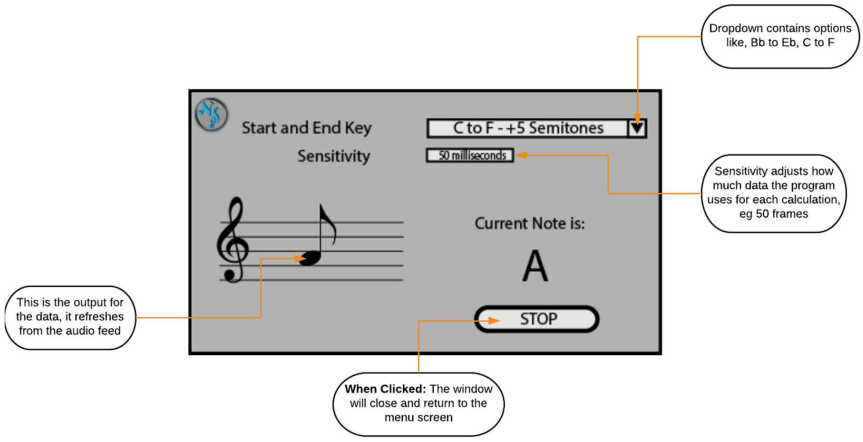



Main Window Remains Open and the Scribe Screen opens
Transcribe
Transpose
Main Window remains open and the Tran Screen opens

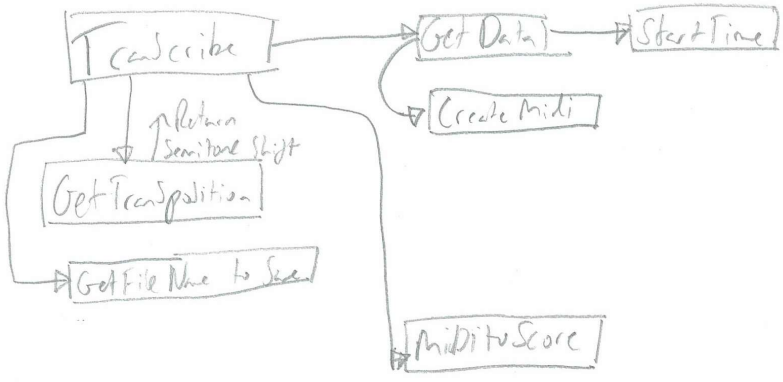
Tran Screen


Transcribe
Get Data
Start Time
Create Midi
Return Semitone Shift
Get Transposition
Get File Name to Save
MidiToScore

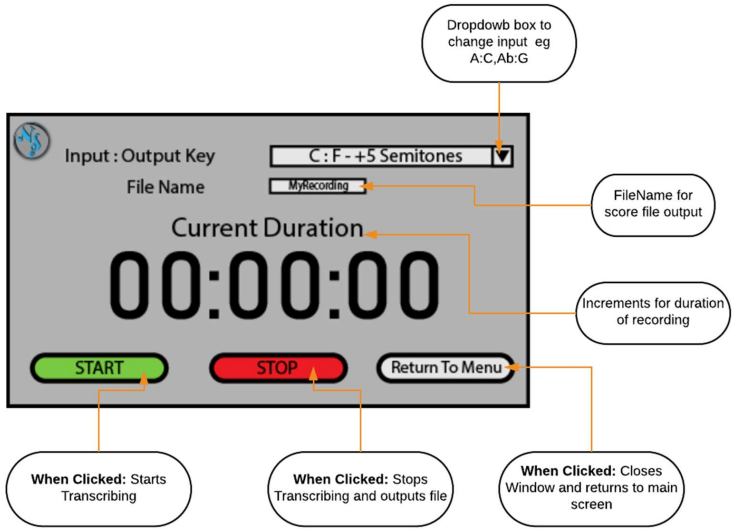
Computer Science NEA - NoteSwitch

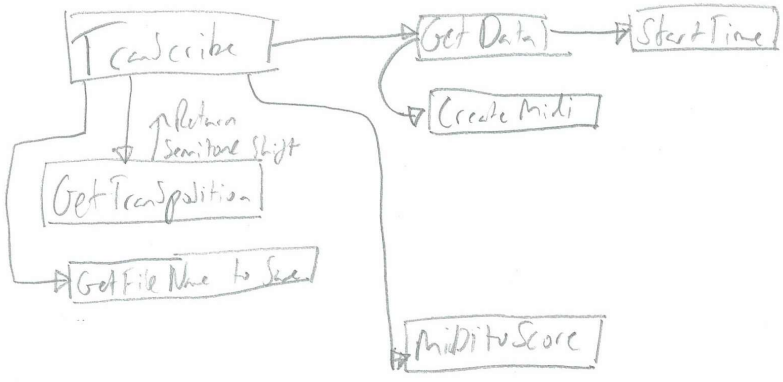


Scribe Screen



Computer Science NEA - NoteSwitch

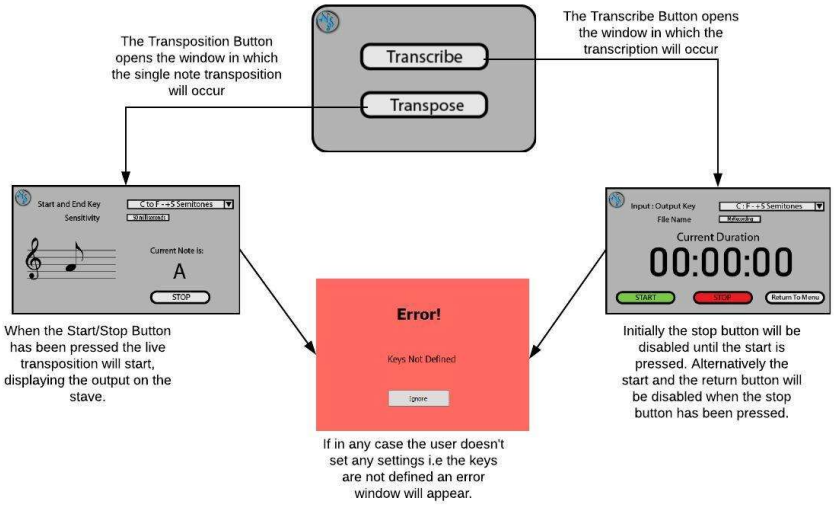




4097 Oscar Lindenbaum 62113 Wheatley Park School

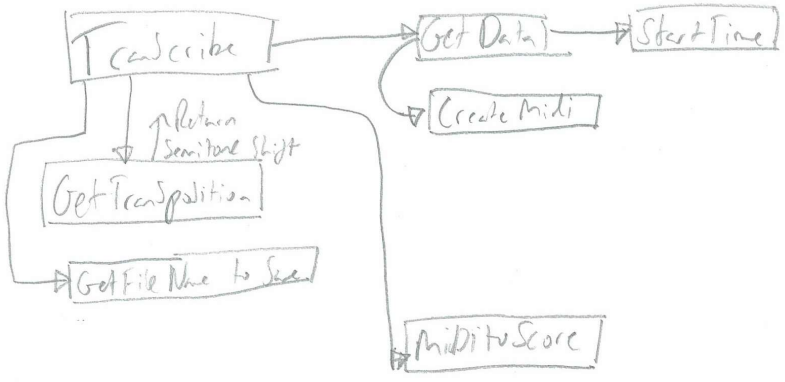
Computer Science NEA - NoteSwitch

Simple GUI Flowchart

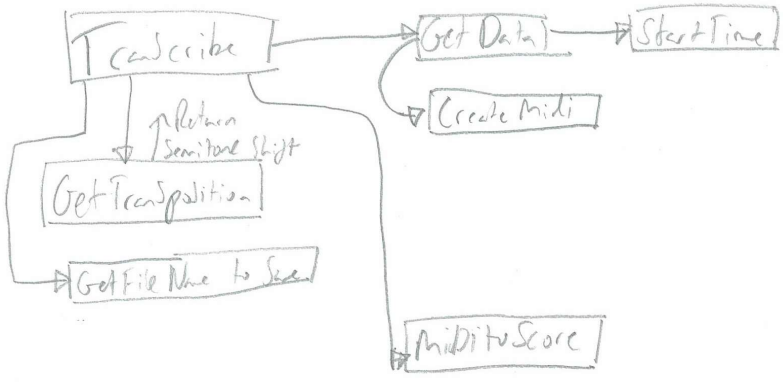


What additional Libraries are required?

Library	**What am I using it for**
PyQt5	Graphical Interface - much nicer output than Tkinter & Faster
Pandas	For formatting SQL Query Results
Numpy	Makes it easier to analyse audio chunks and apply operations.
Dropbox	The Dropbox API is used to access the files which are hosted on dropbox. It also allows us to upload files from the program with a Developer ID.
Aubio	The module which is identifying frequencies from 1024 frame samples and applying the YIN algorithm and Spectral analysis..

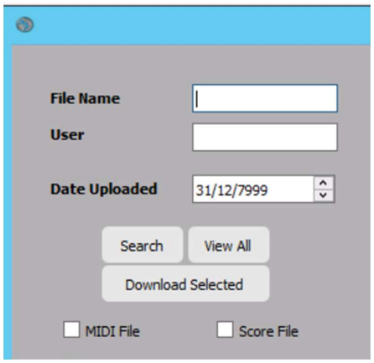


PyAudio	The main framework being used to take in the audio stream.
Pyknon	This is used to create a midi file and to save the data received into the midi format.
os	Used to run the conversion DOS script from Midi to PDF.
sys	Used to terminate the program and any child processes.
Time	Used to create timers and delay processes.
Datetime	Used to correctly retrieve and format the date for the Database entry. It is also used to assist the timers.
Threading	Used to run parallel programs, the GUI and the processes it spawns on button clicks.
Requests	Used to download files once selected from the database
__future__ division	Used to subdivide note lengths in fractional form.
Pyodbc	This library is used to create a connection and send queries to the SQL database
queue	This library is used to transfer data between simultaneously running threads.
fbs	This module compiles pyqt5 projects
dropboxUPLOAD	This module is self created in order to assist with file uploads and SQL queries to the database, it is also used to initialise connections to the database and the dropbox api.
findAudioDevice	This self made module is so that the user can select which audio device to use, it gathers all possible inputs and their API indexes.
locateResources	This self made module is used to locate resource files and their paths, this makes it easier rather than using exact file paths.



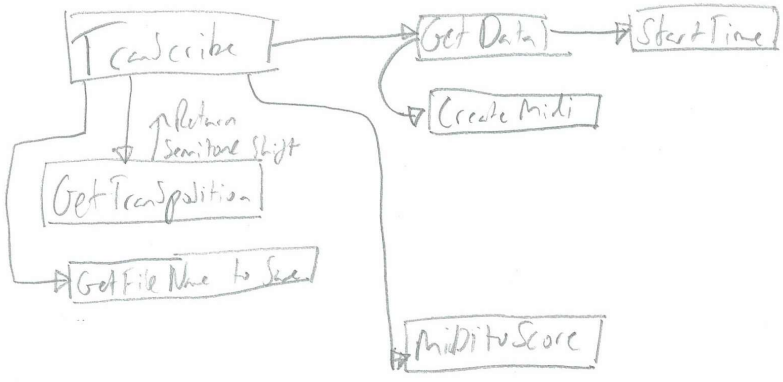
The SQL Query

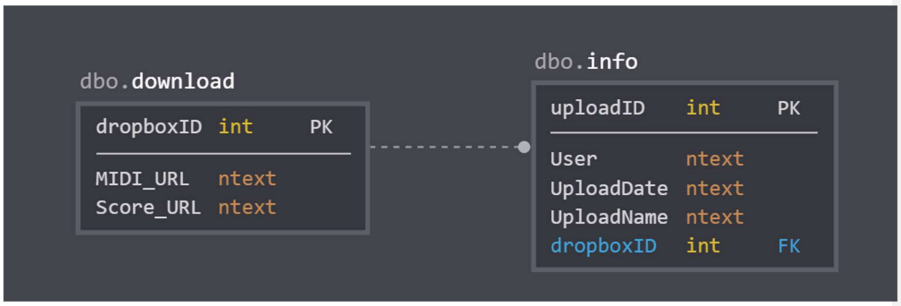
In order to use the different search parameters I will need to build a different SQL statement for each. I will do this by **"SELECT * FROM info WHERE cast([{SEARCHOPTION}] as nvarchar(max)) LIKE '%{SEARCHPARAMETER}%' AND WHERE cast([{SEARCHOPTION2}] as nvarchar(max)) LIKE '%{SEARCHPARAMETER2}%'…"** by searching this way I can search by singular conditions or multiple in whatever combination there is. It will also include similar results to maximise the correct results being found.



In order for the user to view all of the results I will simply select all of the data from the database with "**SELECT * FROM info",** where the info is the table which stores the details of each file.

Entity Relationship Diagram





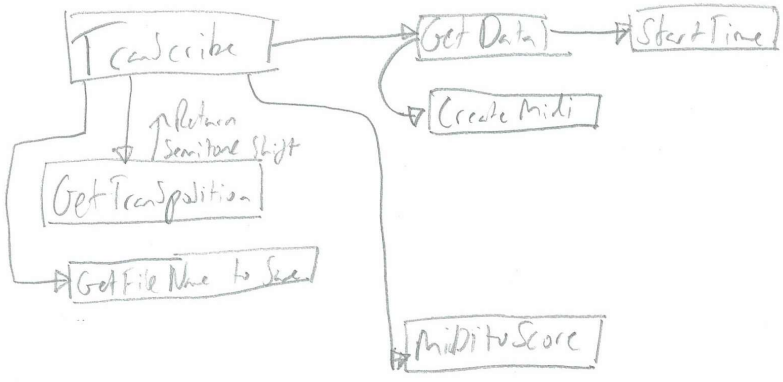
Algorithm Pseudocode

Real Time Audio Analysis

```
CREATE AUDIO STREAM
SET PITCH DETECTION SETTINGS
SET STARTTIME
READ 1024 BLOCK FROM STREAM
EXTRACT PITCH FROM BLOCK
DETERMINE THE MAIN PITCH FROM BLOCK
```

Post Recording Analysis

```
OPEN AUDIO WAVEFORM
APPLY YIN
MERGE CONSECUTIVE FRAMES
DETERMINE LENGTH OF FRAME EVENT
CREATE IMAGE OF SCORE AS LIST
```



4097 Oscar Lindenbaum 62113 Wheatley Park School

Computer Science NEA - NoteSwitch

BPM Detection

OPEN AUDIO WAVEFORM
APPLY SPECTRAL DIFFERENCE FUNCTION
IDENTIFY ONSETS
IDENTIFY BEAT LENGTHS
LOCATE DOWNBEAT ONSETS
EXTRACT DOWNBEAT ONSETS FROM BEAT INDICES
OUTPUT BPM

MIDI File Generation

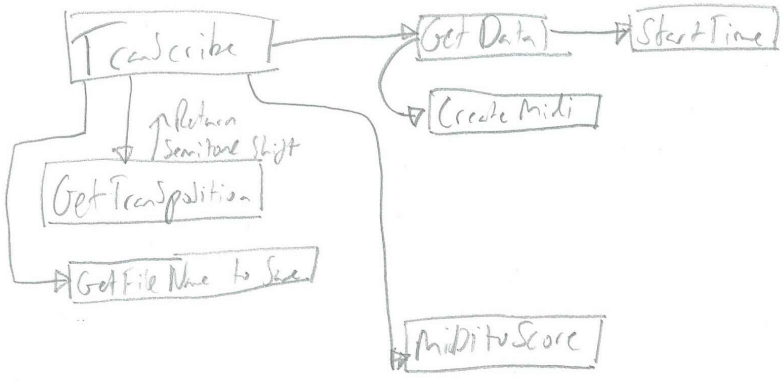
USE PYKNON GENMIDI
CREATE MIDI FILE CLASS
ADD DATA TO CLASS
SAVE DATA AS FILE

MIDI File Conversion

LOCATE midi FILE LOCATION
LOCATE MuseScore
CONVERT USING MuseScore MS-DOS SCRIPT

Generate Dropbox ID (HASH)

OBTAIN DOWNLOAD URL FROM DROPBOX
TAKE THE LAST 5 CHARACTERS AND CHARACTERS 10-5
ADD FIRST SET OF 5 TOGETHER
ADD SECOND SET OF 5 TOGETHER
MULTIPLY THE TOTALS TOGETHER



4097 Oscar Lindenbaum 62113 Wheatley Park School

Computer Science NEA - NoteSwitch

Generate Upload ID (HASH)

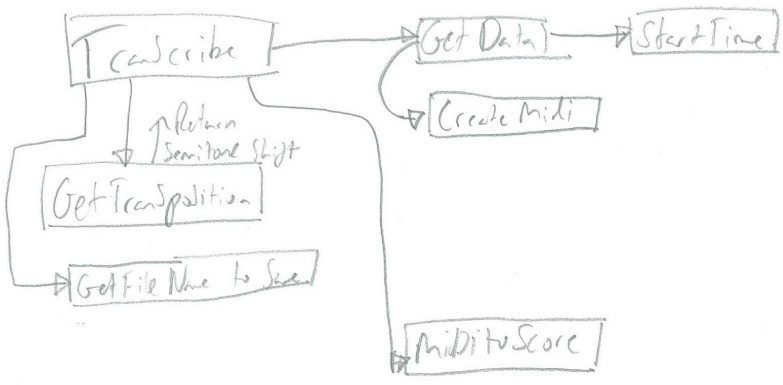
```
OBTAIN DOWNLOAD URL FROM DROPBOX
TAKE THE LAST 5 CHARACTERS AND CHARACTERS 10-5
ADD FIRST SET OF 5 TOGETHER
ADD SECOND SET OF 5 TOGETHER
MULTIPLY THE TOTALS TOGETHER
```

File Upload

```
GATHER FILE NAMES
GATHER FILE LOCATIONS
ESTABLISH DROPBOX CONNECTION
UPLOAD SCORE FILE
GET SCORE FILE URL
UPLOAD MIDI FILE
GET MIDI FILE URL
CLOSE DROPBOX CONNECTION
GATHER ADDITIONAL FILE INFORMATION (file name, date uploaded, user)
ESTABLISH SQL DATABASE CONNECTION
INSERT DATA INTO downloadTable
INSERT DATA INTO infoTable
```

SQL Database Search

```
ESTABLISH SQL DATABASE CONNECTION
SQL: SELECT * FROM info WHERE cast([SEARCH RECORD] as
nvarchar(max))='[SEARCH PARAM]''
GET RECORDID
```



File Download

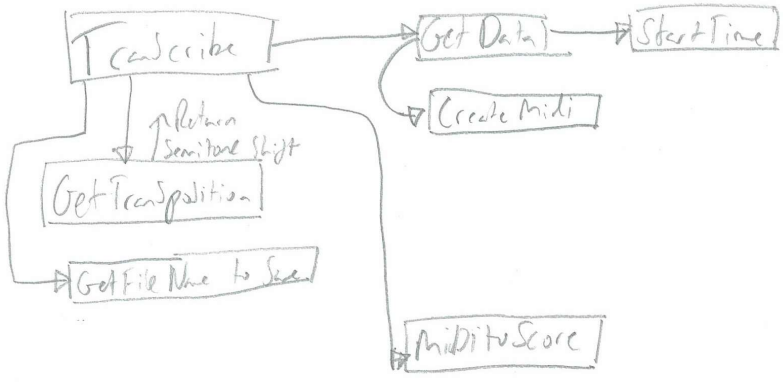
```
GET uploadID
GET dropboxID
GET scoreURL
GET midiURL
DOWNLOAD score
DOWNLOAD midi
```

How will it work?

The user will initially be presented with the main menu where they can select which option they want to start. Depending on the option, the necessary window opens.

- For the Transpose window the program requires two inputs, the start and end key for the transposition. The user can also set the sensitivity of the note identification. When the user selects the start button an audio stream will be created with PyAudio. The program will then take chunks of 1024 frames from the stream at a time and use a Fourier Transform to identify the note, this is taken x times where x is inversely proportional the sensitivity set by the user. An average is then taken from these set of notes and the most commonly identified note will be the 'chosen' note that is currently being played. Once the note has been identified the transposed counterpart will be calculated by cycling around a list of keys by n indexes where n in the difference between the keys in semitones. Then the program will display the text note and its graphical score by selecting the image from the resources folder. *Note: All notes and changed to only be written with sharp signs (#) opposed to flats (b).*

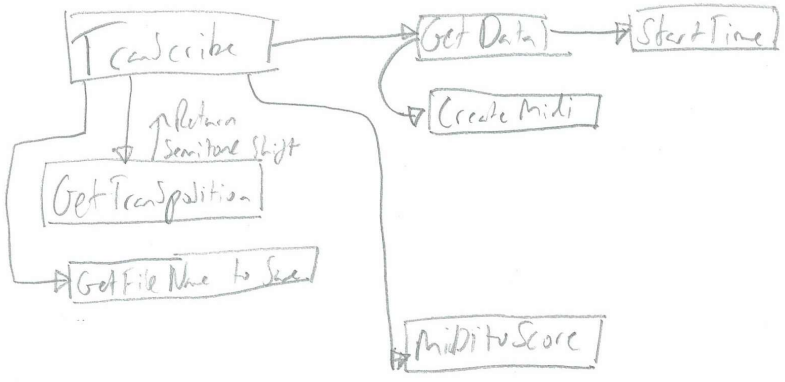
- For the Transcribe Window the program requires four inputs from the user. Once again the start and end keys but additionally a unique user and file name. This is so their won't be filename collisions on the file host. If the user attempts to press the start button while any inputs aren't filled a warning message will be raised and will continue to until all inputs are successful. When the start button is pressed the program will record all audio, until the stop button is pressed, and save it to a waveform file called *recording.wav.* After the recording has been saved we can apply YIN's algorithm through aubio to extract all note values and onsets, from this data we will then concatenate the notes which are the same and produce a smaller list with all note onsets, note frequencies and note offsets. From which we can create duration times. The program will store all this data into a self made data structure class. Next



the midi file will be created by Pyknon, after the file is created the information we collected will be entered as separate Notes or Rests in order they were played. Next the BPM needs to be calculated from the audio clip, as explained earlier we will use spectral analysis through aubio. We do this be extracting the beats detected for one 'HOP' and then dividing 60 by the time between the beats and taking an average. We then add this information to the MIDI file. The program then uses MuseScore to convert the MIDI file to a PDF score file, now both files have been created they will be uploaded to the file server and the database updated. It will do this by initialising a connection to the dropbox API and then uploading the files into their folders, it will also pull a download url for each file. A unique Hash value will then be generated for both the download urls and the database entry, it will add this data to the server by creating the database connection and then inserting the download urls with their hashes as the primary key. Additionally it will add the username, date created and filename to the info table with the other Hash being the primary key for this table. The other button the user can press the reset button which removes any previous inputs.

- The Search window will open a search prompt with a table of results also shown. The user is presented with 3 possible text inputs, the filename, username and the date. If the user presses the search button a SQL query is generated like on page 34, then the results are processed by pandas and presented in the table. The other option for the user is to view all results, this will put all the data into the table. The next thing the user can do is download the score files, they can select an entry in the table and then download both the MIDI file and Score file or either one. The program does this by fetching the unique hash of the selected item and then searching the download table for both the download urls. Then the program fetches the data through the requests module and saves the files in the current directory as **filename**_midi/score.

- The settings window only allows the user to select the audio interface to use, they can refresh the device list which uses pyaudio to look through all audio devices and filters them to only microphone inputs, this is then presented in a dropdown box which the user can choose from and then apply.



Downloading the Files

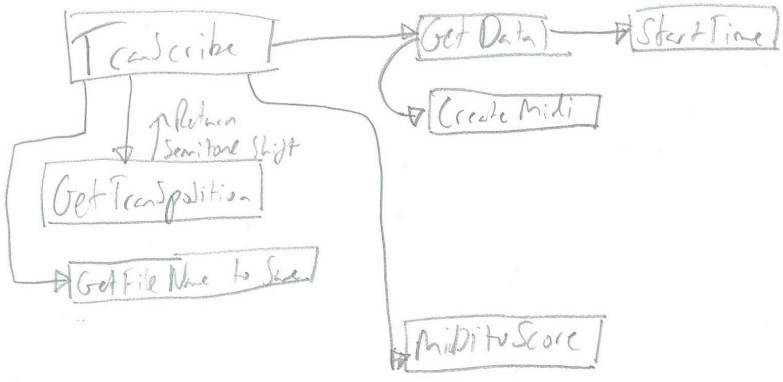
After the user has established what file they want to download I will need to gather the dropboxID and then the URL locations of the files. I will first get the dropboxID from the INFO table which is the primary key in another table called DOWNLOAD. This table contains the urls needed to download the files, in order to get the correct URLS we will use the dropboxID with the command **"SELECT [MIDI_URL],[Score_URL] FROM download WHERE [dropboxID] = ({DROPBOXID})"**. Now that we have the dropbox URLS all we need to do is download them by making a request for the file and saving the contents as the correct file type.

Identifying the Note Played

I have decided it will be best to use a Discrete Fourier Transform using 1024 frames per buffer. This will allow me to analyse the audio with enough accuracy while also being close to realtime. I will apply the fourier transform and then collect the 15 data points, round to the nearest integer value to remove any negligible decimals, and then finally take an average frequency. This will detract from the accuracy if multiple notes were played in quick succession but will increase the note identification accuracy. Only notes that are louder than 50db are recorded to eliminate silence. In order to convert the frequency to note values I will use a predefined table which is stored alongside the program. *See Appendix iii,iv*

Extracting Note Onsets and Note Durations

As mentioned earlier, I will be using the YIN algorithm to determine the onsets of all notes. When applying the algorithm I will also output the onsets of silence to allow me to create rests after analysing the signal. To decipher between rests lengths and note lengths I will search through every frame and its output from YIN (Note Frequency and Onset) and merge all consecutive notes and rests into its initial onset and the offset time. By doing this I can create a list of notes and rests with their lengths in the order that they occur, creating an 'image' of the score to be created.



Creating a MIDI File of the Audio Stream

SImilarly to identifying the note, when creating a MIDI file of the audio stream I will need to collect the note length, and onset time. Additionally the tempo and time signature. At the same time of identifying the note there will be a system stopwatch which record the length of any note over 0.01 seconds long to 3 decimal places.

Converting MIDI File to Score File

Given the timescale of the project, in order to create the Score file I will use a free software package that has this built in. All I will need to do is call this script by running this MS-DOS file.

```
@echo off
start "CONVERT" "C:\Program Files (x86)\MuseScore 2\bin\MuseScore.exe" %~1 -o
%~2%.pdf
EXIT /B 0
```

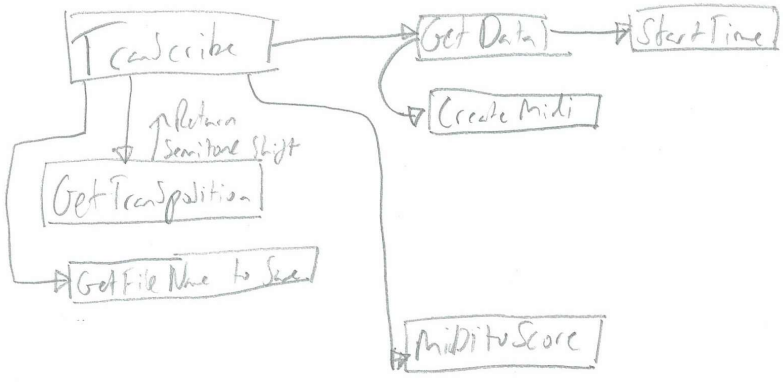
This script, where input %~1 is the MIDI file and %~2 is the output file, will automatically create the score file and save it.

Uploading Files to DropBox and Extracting URL

To upload the files to the database I will first have to upload them onto the file host and create downloadable urls. First I will establish a connection to the DropBox API Folder which is setup for the files. Then I will use the API to upload files into the correct directories within the folder. For the URL I will request a temporary link for the file I just uploaded. This URL will be used to fetch any files the program needs to download.

Adding a New entry to the Database

Once I have created all the files needed and the respective urls I will need to add this data to



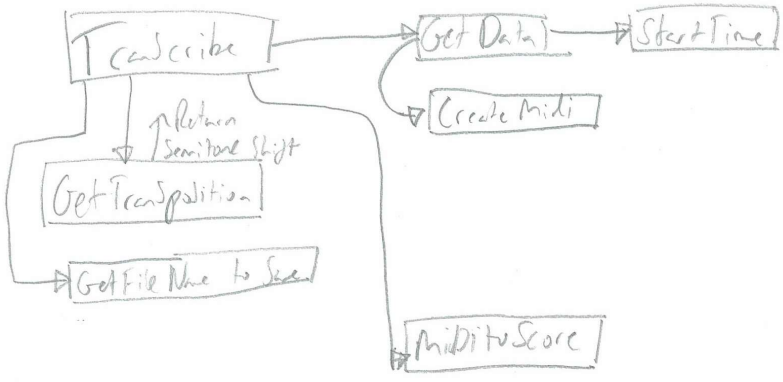
the database. Firstly I will need to generate a Hash for each item so I can identify the unique urls that are associated with each record. This will take multiple SQL queries to add the new record to all of the tables. The first query would be to add the URLS to the hash in the download table.
“INSERT INTO download VALUES(DROPBOXHASH,’MIDIURL’,’SCOREURL’)”
The second to add the attributes of this entry into the info table, also to link the URLS to the file details.
“**INSERT INTO info VALUES (USER,’DATE’,’NAME’,UPLOADHASH,DROPBOXHASH)”**

Displaying real time identified notes

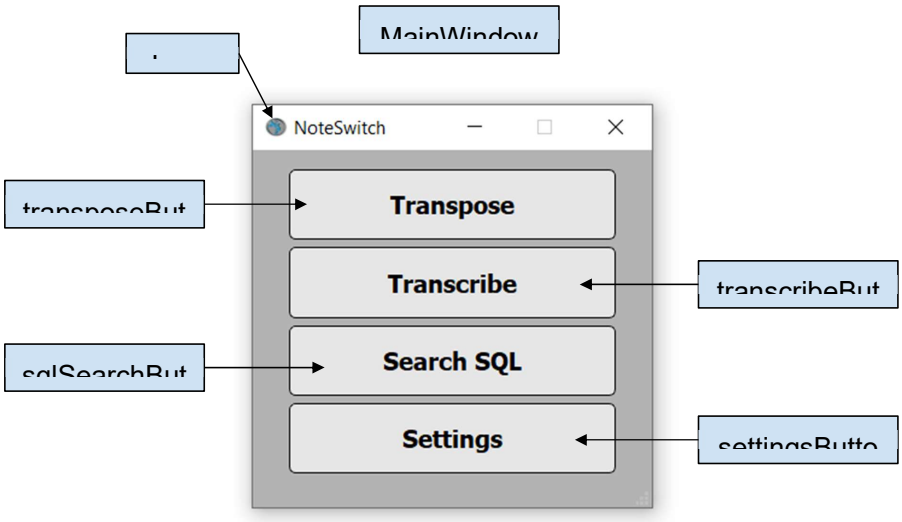
As the real time note identification requires a separate thread to have a constant audio stream, we will also need another thread to display these notes to the user at the same time, because they are running in different threads I will use a global queue called identifiedNoteQueue which will have the notes identified enqueued into it. This is useful because it saves time reassigning variables and sending them through different functions, as the queue uses the FIFO method it means we can continue to identify notes without overwriting the last note, before it has been shown to the user.

Class Inheritance Diagram

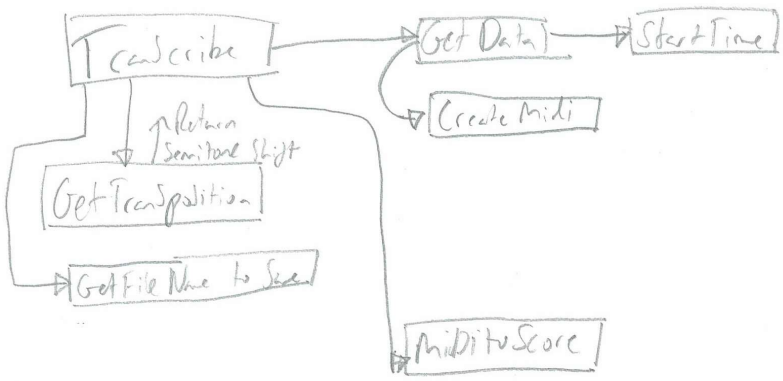
See appendix vi



Technical Solution

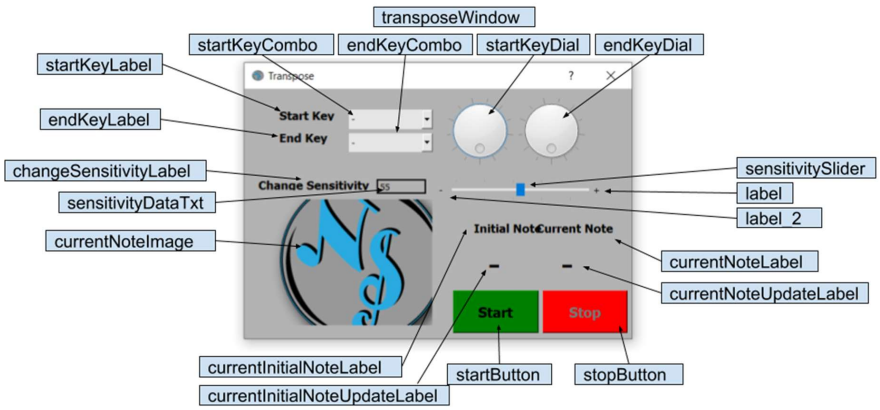


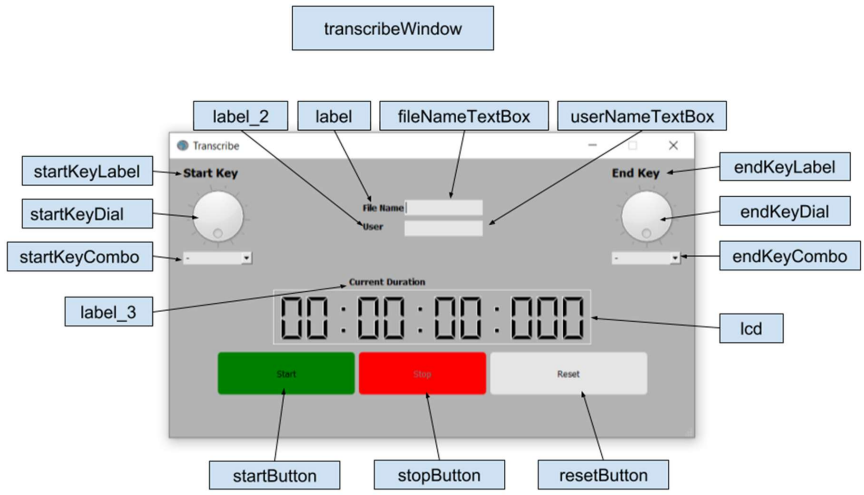
GUI Design Explained

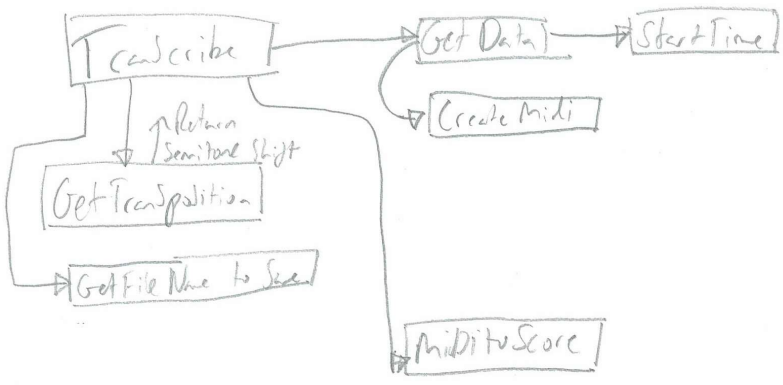


4097 Oscar Lindenbaum 62113 Wheatley Park School

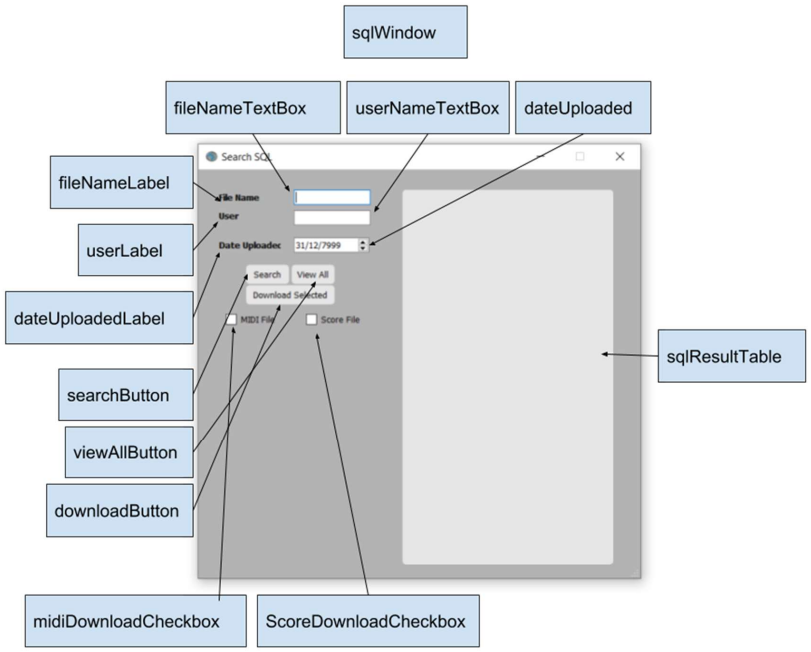
Computer Science NEA - NoteSwitch

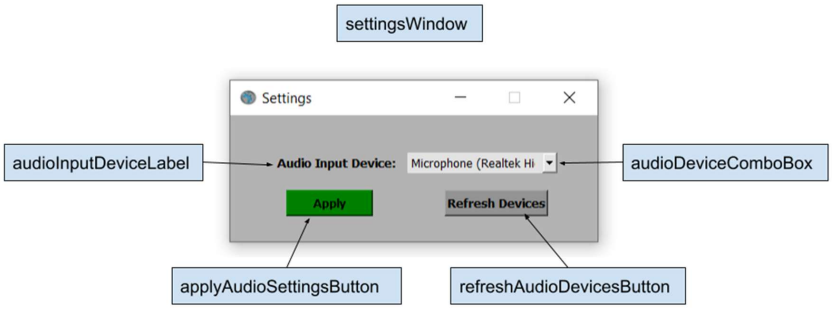


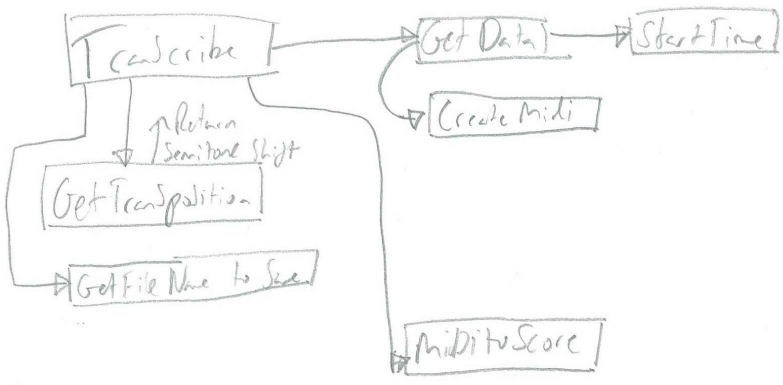




Computer Science NEA - NoteSwitch



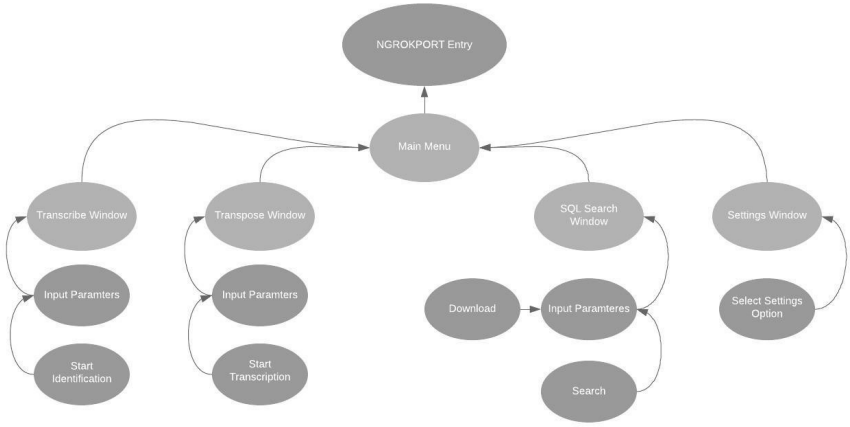




4097 Oscar Lindenbaum 62113 Wheatley Park School

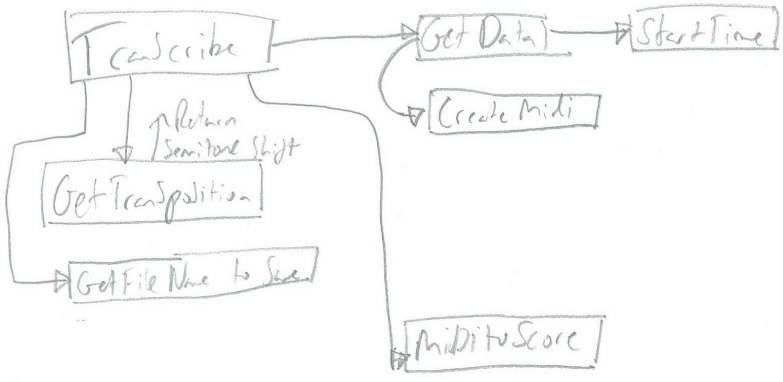
Computer Science NEA - NoteSwitch

System Overview

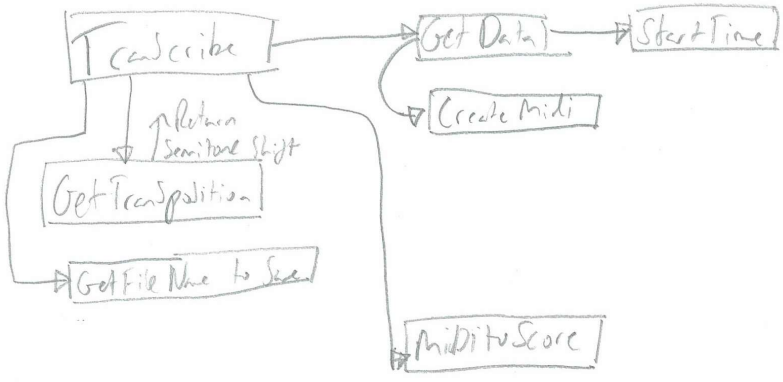


Procedures and Global Variable Summary

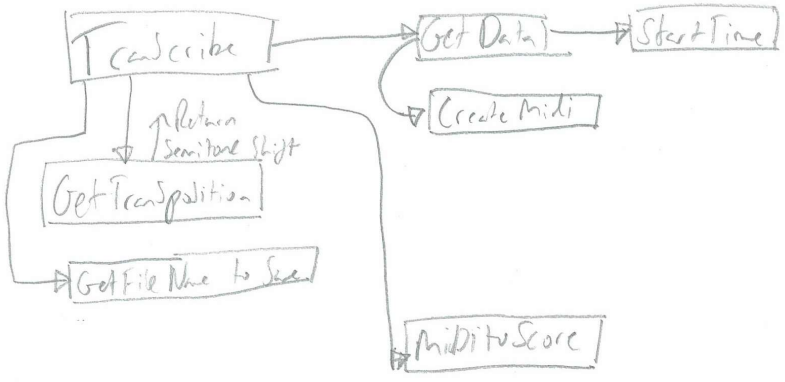
main			
Global Variables	**Purpose**		
df	Stores the SQL results from queries		
pa	Pyaudio object		
stream	Pyaudio data stream		



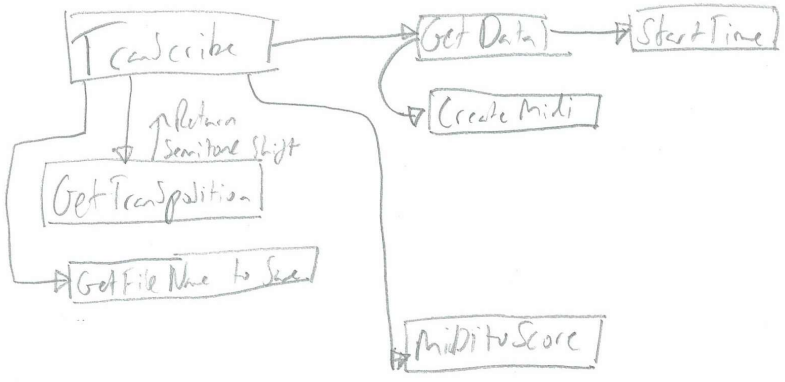
frames	Data stored from the audio feed to be constructed into a WAV file		
ms,s,m,h	The LCD components, milliseconds,seconds,minutes,hours		
scribeTimer	The timer thread		
liveAudioIdentification	The object to identify the live notes		
NGROKPORT	Port needed to access server		
app	The parent GUI		
MainWindow	The Main Window Object		
mainUI	The Main Window components		
transposeWindow	The Transpose Window Object		
transUI	The Transpose Window components		
transcribeWindow	The Transcribe Window Object		
scribeUI	The Transcibe Window Components		
settingsWindow	The Settings Window Object		
settingsUI	The Settings Window Components		
errorWindow	The Error Window Object		
errorUI	The Error Window Components		
sqlWindow	The SQL Window Object		



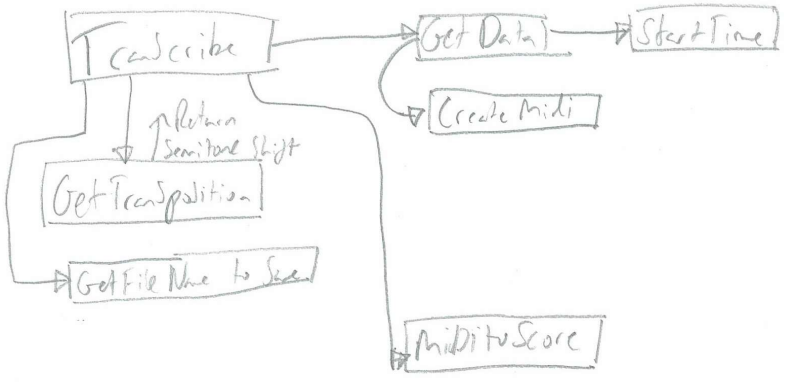
sqlWindowUI	The SQL Window Components		
downloadWindow	The Download Window Object		
downloadWindowUI	The Download Window Components		
transAudioTR	The parent note identification Thread		
keys	The 12 musical keys		
LOGO_PATH	The path for the Logo file		
Procedures	**Sub-Procedures**	**Parameters**	**Purpose**
CLASS guiThread(QThread)			Threads the main application GUI
	SUB __init__	self	Initialises the Thread
	SUB __del__	self	Safely Terminates the Thread
	SUB run	self	Starts the Thread
CLASS downloadWindowStart(QThread)			Threads the downloading from SQL Search
	SUB __init__	self	Initialises the Thread
	SUB __del__	self	Safely Terminates the Thread
	SUB run	self	Starts the Thread
CLASS transAudioThread(QThread)			Threads the audio Identification
	SUB __init__	self	Initialises the Thread
	SUB __del__	self	Safely Terminates the Thread
	SUB run	self	Starts the Thread
CLASS timerThread(QThread)			Threads the LCD Timer
	SUB __init__	self	Initialises the Thread



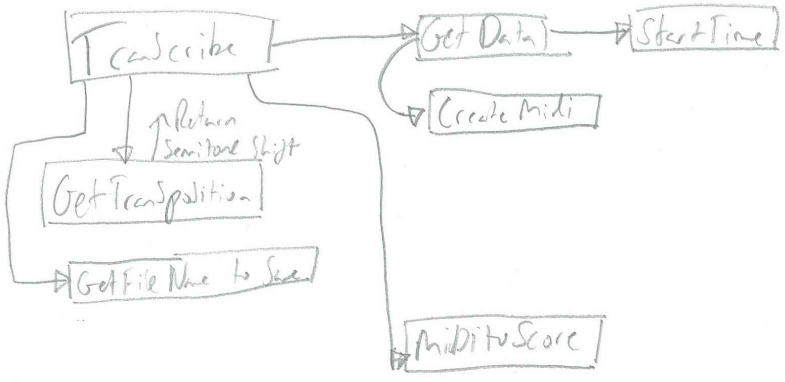
	SUB __del__	self	Safely Terminates the Thread
	SUB run	self	Starts the Thread
CLASS Ui_MainWindow(object)			Creates the MainWindow
	SUB setupUi	self, MainWindow	Sets up the GUI Components
	SUB retranslateUi	self, MainWindow	Allows all the GUI components to be translated into another language
CLASS Ui_Error(object)			Creates the Error Window
	SUB setupUi	self, MainWindow	Sets up the GUI Components
	SUB retranslateUi	self, MainWindow	Allows all the GUI components to be translated into another language
CLASS Ui_sqlWindow(object)			Creates the sqlWindow
	SUB setupUi	self, MainWindow	Sets up the GUI Components
	SUB refreshTable	self, df	Refreshes the data being shown to the user in the results table
	SUB retranslateUi	self, MainWindow	Allows all the GUI components to be translated into another language
CLASS PandasModel(QtCore.QAbstractTableModel)			
	SUB __init__	self, data, parent=None	Sets up the results table to accept modeled data
	SUB rowCount	self, parent=None	Counts the number of rows in the data
	SUB columnCount	self, parent=None	Counts the number of columns in the data



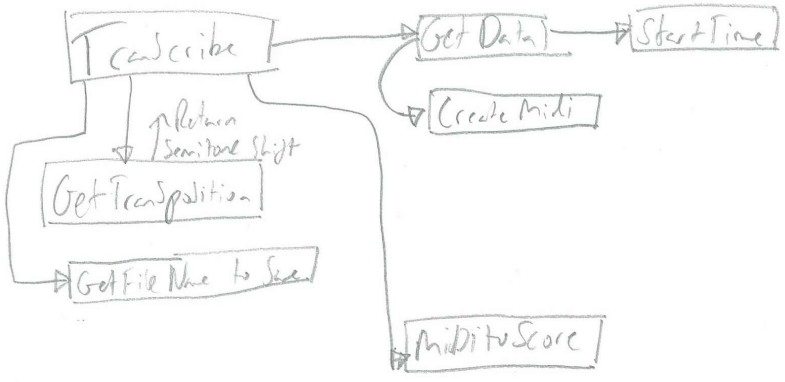
	SUB data	self, index, role=QtCore.Qt.DisplayRole	Converts the SQL results into a pandas model
SUB searchButton_clicked			Generates the SQL Query and executes it against the server, then shows the results
SUB viewAllButton_clicked			Gets all data entries from the server and shows the results
SUB generateSQL		searchOptions,searchType, data	Generates the actual SQL Query to be executed depending on the user input
SUB excepthook		excType, excValue, tracebackobj	Catches any PyQt errors and saves it to a log file. Taken from https://www.riverbankcomputing.com/pipermail/pyqt/2009-May/022961.html
SUB downloadSelected_clicked			Checks the required parameters are set then downloads the files
SUB download		url,name	Sends requests and saves the data at the endpoint
SUB getDownloadLinks		ID	Searches the server by ID and gets the url locations of each file
SUB setupSqlSlots			Sets up the functions to run on specific user interaction
CLASS Worker(QObject)			Gathers the audio data from the stream until told to stop
	SUB __init__	self	Sets up the flag to working so the worker will start



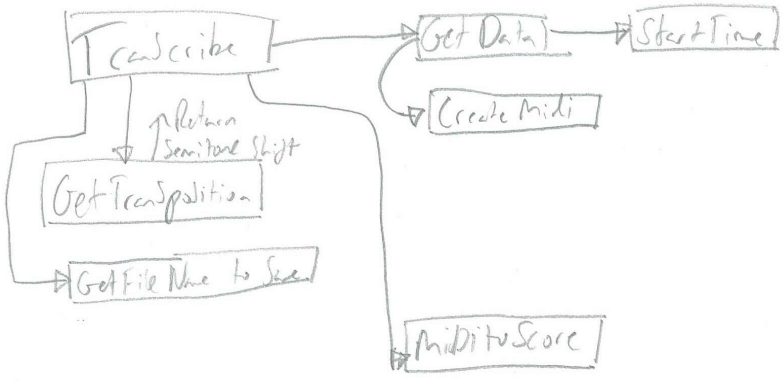
	SUB work	self	The function for the worker to run, it creates an audio stream and collects frames
CLASS Ui_scribeWindow(QObject)			Creates the scribeWindow
	SUB setupUi	self, MainWindow	Sets up the GUI Components
	SUB start_loop	self	Starts the Worker loop and the TimerThread
	SUB stop_loop	self	Stops the Worker from running
	SUB loop_finished	self	Runs when the worker has finished, it saves the frames collected as a WAV file.
	SUB TimerReset	self	Sets the LCD Timer to 00:00:00:00
	SUB TimerStart	self	Starts the Timer Thread so it starts counting
	SUB TimerTime	self	The 'tick' of the clock for the timer
	SUB retranslateUi	self, MainWindow	Allows all the GUI components to be translated into another language
CLASS Ui_Dialog(object)			Creates the Dialog Window
	SUB setupUi	self, Dialog	Sets up the GUI Components
	SUB retranslateUi	self, Dialog	Allows all the GUI components to be translated into another language
CLASS Ui_settingsWindow(object)			Creates the settingsWindow
	SUB setupUi	self, settingsWindow	Sets up the GUI Components



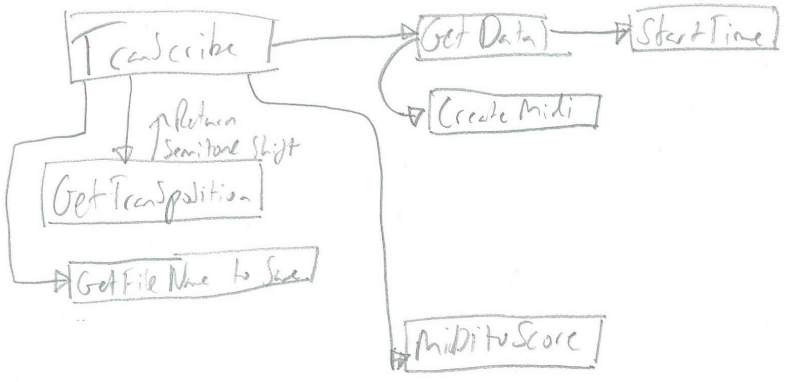
	SUB retranslateUi	self, settingsWindow	Allows all the GUI components to be translated into another language
CLASS Ui_downloadWindow(object)			Creates the downloadWindow
	SUB setupUi	self, Dialog	Sets up the GUI Components
	SUB retranslateUi	self, Dialog	Allows all the GUI components to be translated into another language
SUB getStartKeyData			Displays the starting key set by the user
SUB getEndKeyData			Displays the ending key set by the user
SUB stopButton_clicked			Run when the transpose stop button is clicked, re-enables the user inputs and stops the associated threads safely
SUB convertNoteToOnlySharps		note	Converts any given note to standard form for the program without flats
SUB transposeNote		note	Shifts the note value up or down the amount specified by the user
SUB updateNoteLabel		note	Finds the images for the starting and ending note and shows them to the user
SUB getSensitivityData			Displays the value for the sensitivity set by the user
SUB returnSensitivityData			Returns the value for sensitivity set by the user



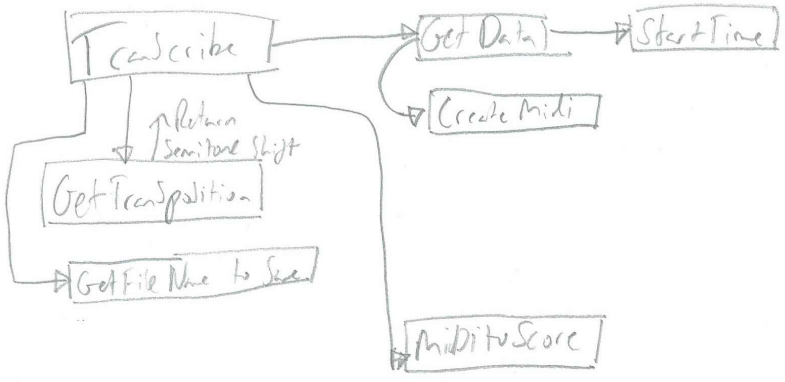
SUB startButton_clicked			Runs when the transpose start button is clicked, disables the user inputs and starts the live identification thread
SUB transposeButton_clicked			Runs when the transpose main window button is clicked, it makes the window visible
SUB transcribeButton_clicked			Runs when the transcribe main window button is clicked, it makes the window visible
SUB transcribeStartButton_clicked			Runs when the transcribe start button is clicked, it checks that all needed inputs are filled and then lets the worker start
SUB generateDropboxHash		url,name,cnxn	Generates a suitable hash for the dropboxID, checks if the hash is unique
SUB generateUploadHash		url,name,cnxn	Generates a suitable hash for the uploadID, checks if the hash is unique
SUB checkUniqueName		name	Checks that the hash is unique using linear probing
SUB debugTrace			Used for debugging while the PyQt thread is running
SUB save			Saves the data recorded and uploads it to the filehost and the server



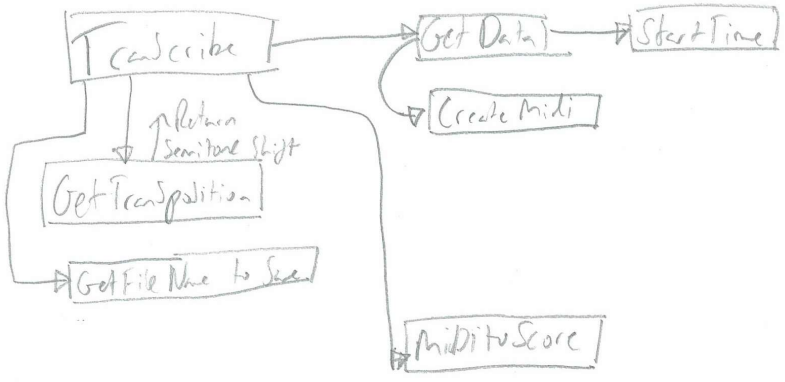
SUB transcribeStopButton_clicked			Runs when the transcribe stop button is clicked, it re-enables all inputs and stops the LCD timer
SUB transcribeResetButton_clicked			Runs when the transcribe reset button is clicked, it sets the window back to its default setup
SUB setupScribeSlots			Sets up the functions to run on specific user interaction
SUB calculateTransposition		ui,keys	Calculates the value to shift by given the users starting and ending keys
SUB sqlButton_clicked			Runs when the main window sql button is clicked, it makes the window visible
SUB setupSlotsTrans			Sets up the functions to run on specific user interaction
SUB setupSlotsMain			Sets up the functions to run on specific user interaction
SUB applyAudioSettings			Runs when the settings window apply button is clicked, it saves the user settings into a text file then hides the settings window
SUB setupSettingsSlots			Sets up the functions to run on specific user interaction
SUB refreshAudioDevices			Runs when the settings window refresh button is clicked, it updates the list of audio devices available for selection



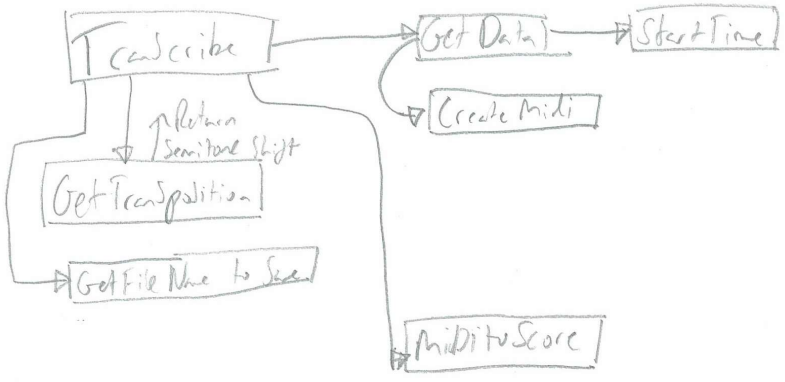
SUB settingsButton_clicked			Runs when the main window settings button is clicked, it refreshes the audio devices and then makes the window visible
SUB closeErrorWindow_clicked			Ran when the error window ignore button is clicked, it hides the window
SUB setupSlotsError			Sets up the functions to run on specific user interaction
SUB raiseError		message,colouring="",text=""	Raises a custom message to the user in a popup box, its colour can be set or the default is used, it can also have a custom message
CLASS liveNoteThread(threading.Thread)			Opens and identifies notes from a live audio stream
	SUB __init__	self,name='liveNoteThread', sensitivity=0.6	Creates the threads and queue, sets up the FREQ and NOTE lists
	SUB run	self	Runs the noteIdentification thread until a stop event is set
	SUB noteIdentification	self	Opens the audio stream and performs FFT and then takes the most common note in x samples, where x is inversely proportional to the sensitivity slider
	SUB closest	self,num	Finds the closest musical note given a floating frequency value
	SUB join	self	Safely terminates the threads



SUB whileGo			Runs the audioIdentification thread at the same time as updating the GUI for the user with the notes identified
CLASS AppContext(Application Context)			Runs the GUI without causing PyQt errors
	SUB run	self	Sets up all the threads, Windows with their slots and Ui, then runs the main app
notateWAVpostRecording			
Procedure	**Sub-Procedures**	**Parameters**	**Purpose**
CLASS dataStore:			To store all the information about the audio file
	SUB __init__	self	Creates the variables for storing the different parts of the data
	SUB view	self	Displays all the data
	SUB add	self,note,duration,onset	Adds the note data into the data structure, with the rest being note "-1"
	SUB setTempo	self,bpm	Sets the tempo for the midi file
	SUB export	self	Strips the leading empty data from the structure
SUB toValueOctave		txt	Converts a given note into its octave value and note value
SUB save		AudioData,name,transpositionValue	Generates the midi file then converts it to a score file
SUB debugTrace			Used to debug while the PyQt GUI is running



SUB analyse		fileName,AudioData	Applies the YIN algorithm to the WAV file and calculates note durations
SUB getBPM		path	Uses spectral analysis to calculate the BPM
SUB transposeNote		note,transpositionValue	Transposes a given note by a user specified amount, this version preserves the octave shift
SUB postAnalyseAudio		wav_file,name,transposition Value	Creates the audio structure then analyses it and saves the data.
locateResources			
Procedures	**Parameters**	**Purpose**	
SUB findMostSimilarFile	fileToFind,possible	Finds the closest file name in the given directory to a name	
SUB findFile	fileToFind	Finds all the files in the directory and then runs findMostSimilarFile	
dropboxUPLOAD			
Procedures	**Parameters**	**Purpose**	
SUB connect_DB		Initialises a connection to the filehost	
SUB upload	midiFileName,score FileName	Uploads the files onto the filehost into their desired folders and returns the download links	
SUB connect_SQL	NGROKPORT	Initialises a connection to the server	
SUB query	cmd,cnxn	Executes a Query against the server	
SUB checkHash	HASH,column,cnxn	Checks if a Hash already exists in the table	
SUB start	NGROKPORT	Starts the SQL Connection	
SUB stop		Stops the SQL Connection	



findAudioDevices			
Procedures	**Parameters**	**Purpose**	
locate		Searches through all the available audio devices and returns only the microphone capable devices	

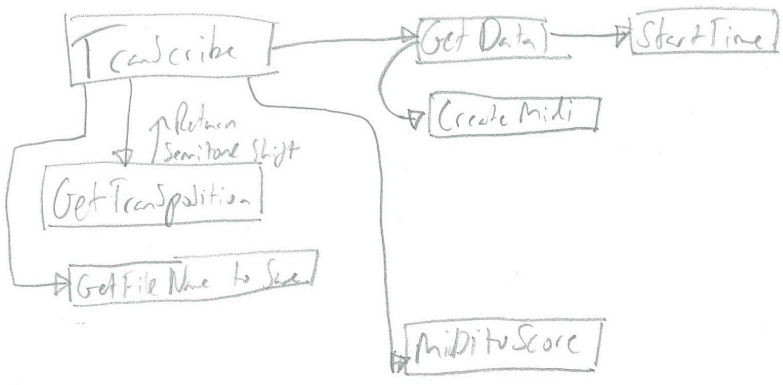
Structured Code

See appendix v.

Code Debunked

Comments and Explanations are formatted like this.

Main.py



```
from fbs_runtime.application_context import ApplicationContext
from PyQt5 import QtCore, QtGui, QtWidgets
import sys,os
from PyQt5.QtCore import pyqtSlot,QThread,pyqtSignal,QObject
from PyQt5.QtGui import *
import time, random , datetime
import dropboxUPLOAD as sql
import locateResources
import requests
global df
import threading
import queue
import pyaudio
import aubio
import numpy as np
import notateWAVpostRecording
```

class *guiThread*

```
class guiThread(QThread):
       def __init__(self):
              QThread.__init__(self)
       def __del__(self):
              self.wait()
       def run(self):
              app.exec_()
```

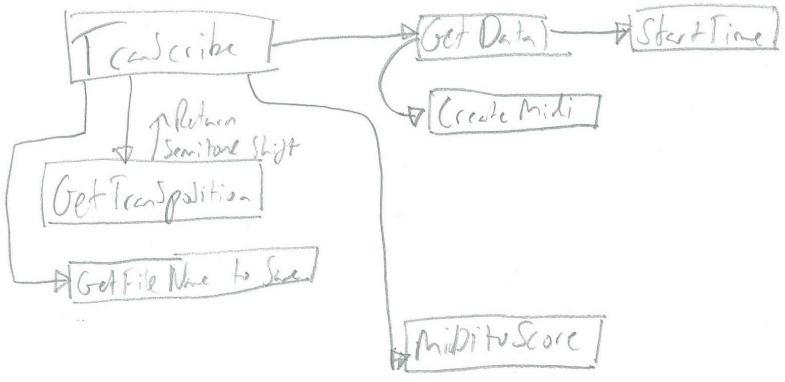
This class when controls the entire gui, it does this without interrupting other processes by threading it using QThread.

class *downloadWindowStart*

```
class downloadWindowStart(QThread):
        def __init__(self):
               QThread.__init__(self)
        def __del__(self):
               self.wait()
        def run(self):
                downloadSelected_clicked()
```

This class creates a background process which downloads the selected items from the table. It is backgrounded so it doesn't halt other processes.

class *transAudioThread*



```
class transAudioThread(QThread):
    def __init__(self):
        QThread.__init__(self)
    def __del__(self):
        self.wait()
    def run(self):
        whileGo()
        ##audio function start()
```

This class runs the whileGo() function in another thread, the whileGo() function will be started when the transposition button is clicked.

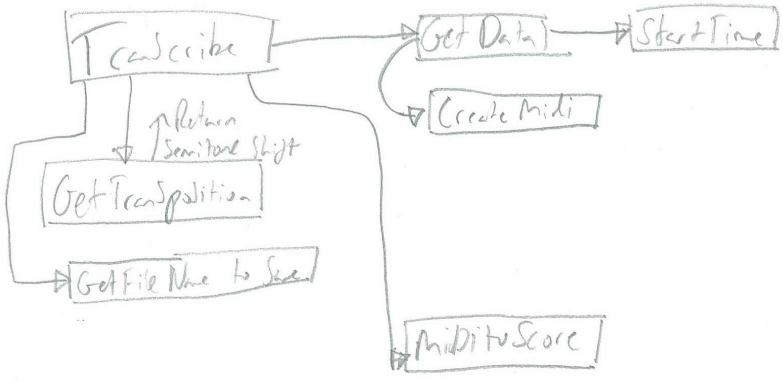
***class** timerThread*

```
class timerThread(QThread):
    def __init__(self):
        QThread.__init__(self)
    def __del__(self):
        self.wait()
    def run(self):
        scribeUI.timer.start(1)
```

This class threads the on screen timer in the transcription window. This is required as otherwise it would halt other processes.

class Ui_MainWindow

```
##MainWindow
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.setFixedSize(331, 295)
        MainWindow.setTabShape(QtWidgets.QTabWidget.Rounded)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.splitter = QtWidgets.QSplitter(self.centralwidget)
        self.splitter.setGeometry(QtCore.QRect(30, 16, 271, 251))
        self.splitter.setOrientation(QtCore.Qt.Vertical)
        self.splitter.setObjectName("splitter")
        self.transposeButton = QtWidgets.QPushButton(self.splitter)
        font = QtGui.QFont()
        font.setPointSize(15)
        self.transposeButton.setFont(font)
        self.transposeButton.setObjectName("transposeButton")
        self.transcribeButton = QtWidgets.QPushButton(self.splitter)
        font = QtGui.QFont()
        font.setPointSize(15)
```

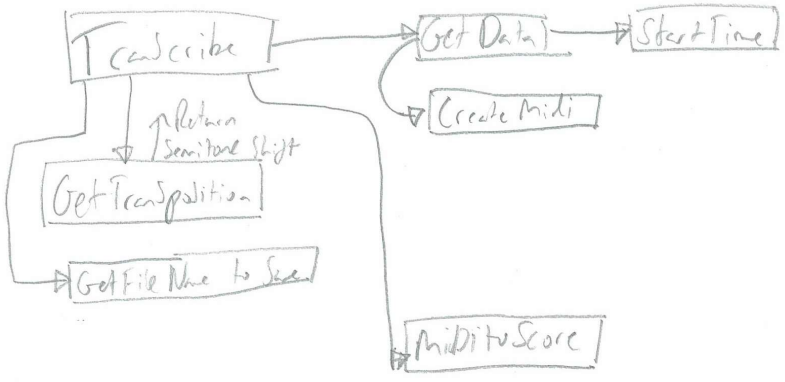


```
        self.transcribeButton.setFont(font)
        self.transcribeButton.setObjectName("transcribeButton")
        self.sqlSearchButton = QtWidgets.QPushButton(self.splitter)
        font = QtGui.QFont()
        font.setPointSize(15)
        self.sqlSearchButton.setFont(font)
        self.sqlSearchButton.setObjectName("sqlSearchButton")
        self.settingsButton = QtWidgets.QPushButton(self.splitter)
        font = QtGui.QFont()
        font.setPointSize(15)
        self.settingsButton.setFont(font)
        self.settingsButton.setObjectName("settingsButton")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 331, 26))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "NoteSwitch"))
        self.transposeButton.setText(_translate("MainWindow", "Transpose"))
        self.transcribeButton.setText(_translate("MainWindow", "Transcribe"))
        self.sqlSearchButton.setText(_translate("MainWindow", "Search SQL"))
        self.settingsButton.setText(_translate("MainWindow", "Settings"))
```

This creates the initial menu window, it contains four buttons: Transpose,Transcribe, Search SQL and Settings.

class Ui_Error

```
##Error Window
class Ui_Error(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("ErrorWindow")
        MainWindow.setFixedSize(390, 289)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.errorLabel = QtWidgets.QLabel(self.centralwidget)
        self.errorLabel.setGeometry(QtCore.QRect(80, 40, 211, 51))
        font = QtGui.QFont()
        font.setPointSize(20)
```

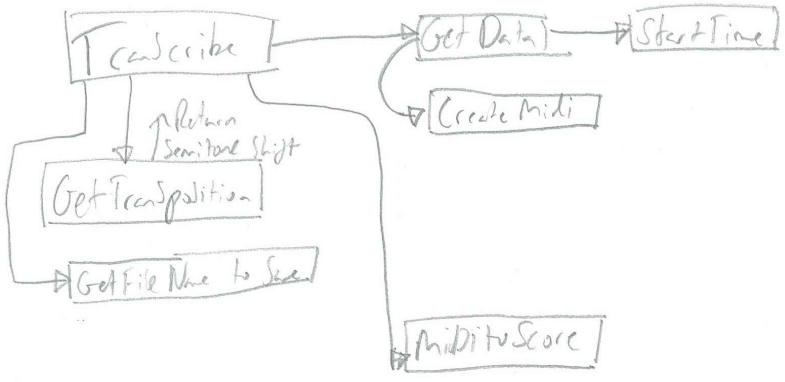


```
        font.setBold(True)
        font.setWeight(75)
        self.errorLabel.setFont(font)
        self.errorLabel.setTextFormat(QtCore.Qt.RichText)
        self.errorLabel.setScaledContents(False)
        self.errorLabel.setAlignment(QtCore.Qt.AlignCenter)
        self.errorLabel.setObjectName("errorLabel")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(20, 90, 341, 111))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.label.setFont(font)
        self.label.setAlignment(QtCore.Qt.AlignCenter)
        self.label.setWordWrap(True)
        self.label.setObjectName("label")
        self.ignoreButton = QtWidgets.QPushButton(self.centralwidget)
        self.ignoreButton.setGeometry(QtCore.QRect(130, 200, 111, 31))
        self.ignoreButton.setObjectName("ignoreButton")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 390, 26))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        MainWindow.setWindowIcon(QIcon(LOGO_PATH))
        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)



    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("Transposition", "ErrorWindow"))
        self.errorLabel.setText(_translate("ErrorWindow", "Error !"))
        self.label.setText(_translate("ErrorWindow", "errorMessageContentsHere"))
        self.ignoreButton.setText(_translate("ErrorWindow", "Ignore"))
```

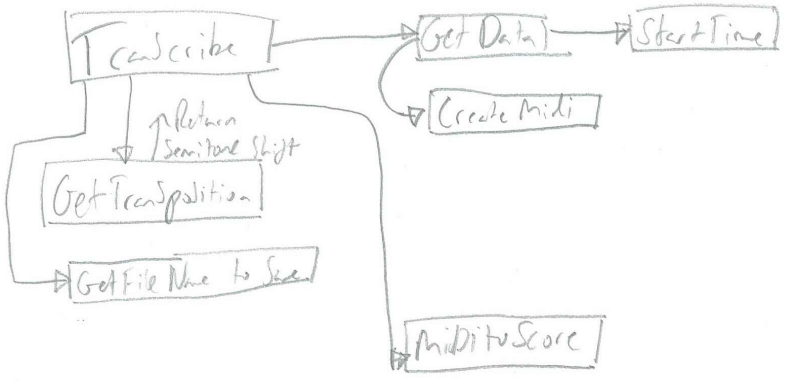
This creates the error window, it contains a dynamic text placeholder which will be changed depending on the message intended to be shown. There is also an ignore button.

class Ui_sqlWindow

```
##SQL WINDOW
class Ui_sqlWindow(object):
    def setupUi(self, MainWindow):
```

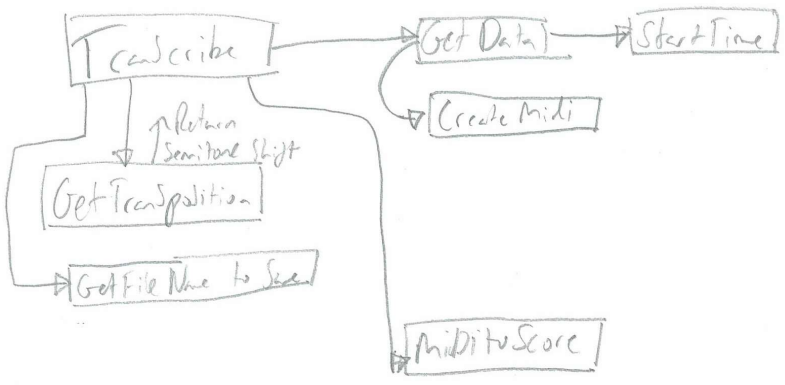


```
        MainWindow.setObjectName("sqlWindow")
        MainWindow.setFixedSize(650, 600)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.fileNameLabel = QtWidgets.QLabel(self.centralwidget)
        self.fileNameLabel.setGeometry(QtCore.QRect(30, 30, 81, 21))
        self.fileNameLabel.setObjectName("fileNameLabel")
        self.userLabel = QtWidgets.QLabel(self.centralwidget)
        self.userLabel.setGeometry(QtCore.QRect(30, 60, 55, 16))
        self.userLabel.setObjectName("userLabel")
        self.fileNameTextBox = QtWidgets.QLineEdit(self.centralwidget)
        self.fileNameTextBox.setGeometry(QtCore.QRect(140, 30, 113, 22))
        self.fileNameTextBox.setObjectName("fileNameTextBox")
        self.userNameTextBox = QtWidgets.QLineEdit(self.centralwidget)
        self.userNameTextBox.setGeometry(QtCore.QRect(140, 60, 113, 22))
        self.userNameTextBox.setObjectName("userNameTextBox")
        self.dateUploadedLabel = QtWidgets.QLabel(self.centralwidget)
        self.dateUploadedLabel.setGeometry(QtCore.QRect(30, 100, 91, 21))
        self.dateUploadedLabel.setObjectName("dateUploadedLabel")
        self.dateUploaded = QtWidgets.QDateEdit(self.centralwidget)
        self.dateUploaded.setGeometry(QtCore.QRect(140, 100, 111, 22))
        self.dateUploaded.setObjectName("dateUploaded")
        sqlWindowUI.dateUploaded.setDate(datetime.date(7999,12,31))
        self.downloadButton = QtWidgets.QPushButton(self.centralwidget)
        self.downloadButton.setGeometry(QtCore.QRect(70, 170, 130, 28))
        self.downloadButton.setObjectName("downloadButton")
        self.splitter = QtWidgets.QSplitter(self.centralwidget)
        self.splitter.setGeometry(QtCore.QRect(40, 210, 241, 20))
        self.splitter.setOrientation(QtCore.Qt.Horizontal)
        self.splitter.setObjectName("splitter")
        self.midiDownloadCheckbox = QtWidgets.QCheckBox(self.splitter)
        self.midiDownloadCheckbox.setObjectName("midiDownloadCheckbox")
        self.scoreDownloadCheckbox = QtWidgets.QCheckBox(self.splitter)
        self.scoreDownloadCheckbox.setObjectName("scoreDownloadCheckbox")
        self.searchButton = QtWidgets.QPushButton(self.centralwidget)
        self.searchButton.setGeometry(QtCore.QRect(70, 140, 63, 28))#80, 140, 111, 28
        self.searchButton.setObjectName("searchButton")
        self.viewAllButton = QtWidgets.QPushButton(self.centralwidget)
        self.viewAllButton.setGeometry(QtCore.QRect(137, 140, 63, 28))
        self.viewAllButton.setObjectName("viewAllButton")
        self.sqlResultTable = QtWidgets.QTableView(MainWindow)
        self.sqlResultTable.setStyle(QtWidgets.QStyleFactory.create('Fusion'))
        self.sqlResultTable.setSelectionBehavior(self.sqlResultTable.SelectRows)

self.sqlResultTable.setSelectionMode(QtWidgets.QAbstractItemView.SingleSelection)
        self.sqlResultTable.setGeometry(QtCore.QRect(300, 30, 310, 550))
        self.sqlResultTable.setObjectName("sqlResultTable")
        self.sqlResultTable.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)

##self.sqlResultTable.setHorizontalHeaderLabels(["ID","User","UploadName","UploadDate"]
```



```
)
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 26))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        ##SHELL TEXT
        self.shellTextBoxSQL = QtWidgets.QLabel(self.centralwidget)
        self.shellTextBoxSQL.setGeometry(QtCore.QRect(20, 240, 251, 301))
        self.shellTextBoxSQL.setObjectName("shellTextBoxSQL")
        self.shellTextBoxSQL.setWordWrap(True)
        self.shellTextBoxDownload = QtWidgets.QLabel(self.centralwidget)
        self.shellTextBoxDownload.setGeometry(QtCore.QRect(20, 350, 251, 301))
        self.shellTextBoxDownload.setObjectName("shellTextBoxDownload")
        self.shellTextBoxDownload.setWordWrap(True)


        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)


    def refreshTable(self, df):
        self.sqlResultTable.setModel(PandasModel(df))
        self.sqlResultTable.setColumnHidden(4,True)
        self.sqlResultTable.setColumnHidden(3,True)###Hiding ID Columns

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("sqlWindow", "Search SQL"))
        self.fileNameLabel.setText(_translate("sqlWindow", "File Name"))
        self.userLabel.setText(_translate("sqlWindow", "User"))
        self.dateUploadedLabel.setText(_translate("sqlWindow", "Date Uploaded"))
        self.searchButton.setText(_translate("sqlWindow", "Search"))
        self.downloadButton.setText(_translate("sqlWindow","Download Selected"))
        self.midiDownloadCheckbox.setText(_translate("sqlWindow", "MIDI File"))
        self.scoreDownloadCheckbox.setText(_translate("sqlWindow", "Score File"))
        self.shellTextBoxSQL.setText(_translate("sqlWindow",""))
        self.shellTextBoxDownload.setText(_translate("sqlWindow",""))
        self.viewAllButton.setText(_translate("sqlWindow", "View All"))
```

This creates the sql search window: It contains three buttons, two text inputs, two checkboxes, a date input, and a table of results which will show the search results from the sql queries. The function refreshTable() will update the table shown to the user after the contents have been changed.



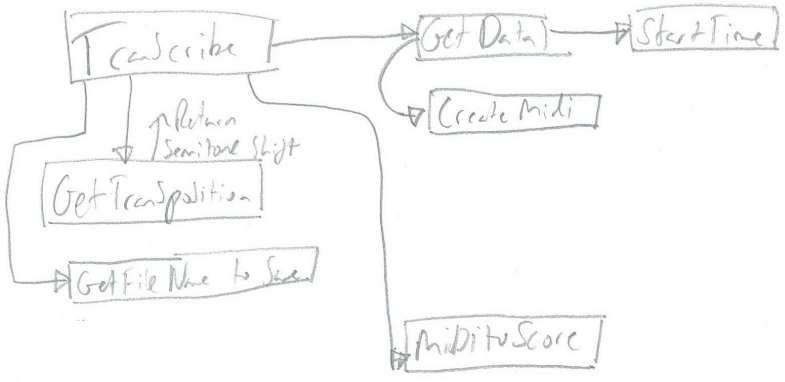
class PandasModel

```
class PandasModel(QtCore.QAbstractTableModel):
    def __init__(self, data, parent=None):
        QtCore.QAbstractTableModel.__init__(self, parent)
        self._data = data

    def rowCount(self, parent=None):
        return len(self._data.values)

    def columnCount(self, parent=None):
        return self._data.columns.size

    def data(self, index, role=QtCore.Qt.DisplayRole):
        if index.isValid():
            if role == QtCore.Qt.DisplayRole:
                return QtCore.QVariant(str(
                    self._data.values[index.row()][index.column()]))
        return QtCore.QVariant()
```

This class converts a pandas sql result and creates a 'model', enabling it to be shown in the search results table.

searchButton_clicked()

```
@pyqtSlot()
def searchButton_clicked():
    fileName = sqlWindowUI.fileNameTextBox.text()
    userName = sqlWindowUI.userNameTextBox.text()
    date =  ("/".join(str(sqlWindowUI.dateUploaded.date().toPyDate()).split("-")[::-
1]))
    searchOptions = [False,False,False]
    searchType = ["UploadName","User","UploadDate"]
    data = [fileName,userName,date]
    if fileName == "":
        searchOptions[0] = False
    else:
        searchOptions[0] = True

    if userName == "":
        searchOptions[1] = False
    else:
        searchOptions[1] = True

    if date == "31/12/7999":
        searchOptions[2] = False
```



```
    else:
        searchOptions[2] = True
    if any(searchOptions):
        cnxn = sql.start(NGROKPORT)
        global df
        df = sql.query(generateSQL(searchOptions,searchType,data),cnxn)
        sqlWindowUI.df = df
        sqlWindowUI.refreshTable(df)
        ###sql.close()

    else:
        sqlWindowUI.shellTextBoxSQL.setText("No Search Parameters")
```

This function has the pyqt slot decorator which means it can be ran in the event of a button click which will be setup later in setupSlots. The function gets the inputs from the user and then decides which options have been changed by the user so it can generate a valid SQL statement to search the database. It then runs the generated query and refreshes the results table.

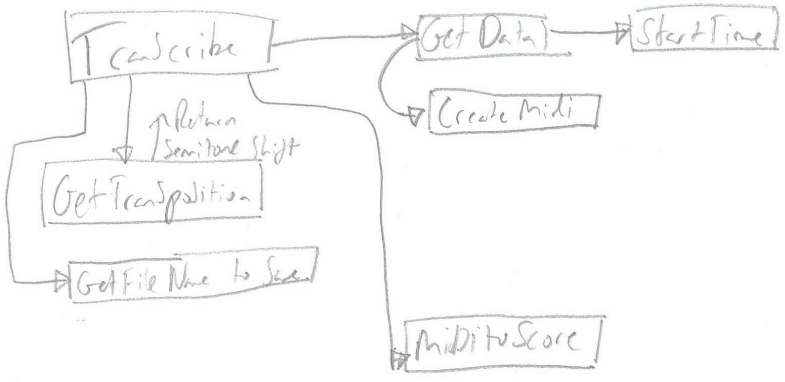
viewAllButton_clicked()

```
@pyqtSlot()
def viewAllButton_clicked():
        cnxn = sql.start(NGROKPORT)
        global df
        df = sql.query("SELECT * FROM info",cnxn)
        sqlWindowUI.df = df
        sqlWindowUI.refreshTable(df)
        ###sql.close()
```

*This function is like the other searchButton function except it doesn't generate any query as we want all the data. It then executes the 'SELECT * FROM info' statement and refreshes the table.*

generateSQL()

```
def generateSQL(searchOptions,searchType,data):
    query = "SELECT * FROM info WHERE "
    idxs = [i for i, x in enumerate(searchOptions) if x == True]
    for idx in idxs:
        current = "cast([{}] as nvarchar(max)) LIKE '%{}%' AND
".format(searchType[idx],data[idx])
```

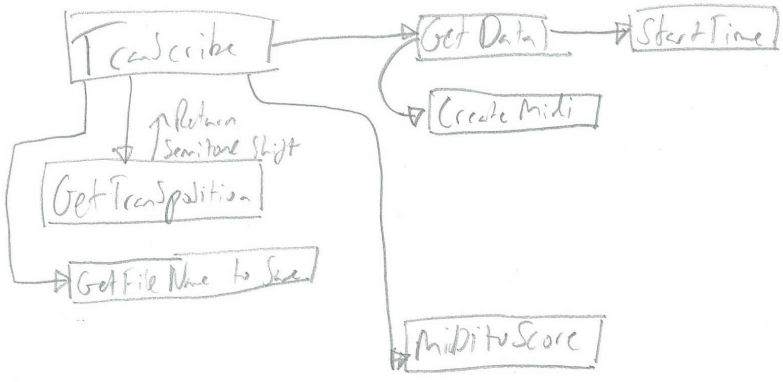


```
        query += current
    if query[-5:]==" AND ":
        query = query[0:len(query)-5]
    sqlWindowUI.shellTextBoxSQL.setText(query)
    return query
```

This function is given boolean values for each search option and its parameters. For all True searches it will append 'cast([parameterType] as nvarchar(max)) LIKE '%parameter%' AND' to its current query. After all true options have been appended it will remove the trailing 'AND' from the statement.

downloadSelected_clicked()

```
@pyqtSlot()
def downloadSelected_clicked():
    if sqlWindowUI.midiDownloadCheckbox.checkState() == 2:
        midiCheck = True
    else:
        midiCheck = False
    if sqlWindowUI.scoreDownloadCheckbox.checkState() == 2:
        scoreCheck = True
    else:
        scoreCheck = False
    idxSelected = sqlWindowUI.sqlResultTable.selectedIndexes()


    if midiCheck == False and scoreCheck == False:
        raiseError("No File Type Selected to download")
        out =  ("ERROR: NOT SELECTED")
    elif len(idxSelected)==0:
        raiseError("No Entry Selected")
        out =  ("ERROR: NO ENTRY SELECTED")
    else:
        idx = idxSelected[0].row()
        out = ("""MIDIFILE CHECKED {}
SCOREFILE CHECKED {}
INDEX SELECTED {}""".format(midiCheck,scoreCheck,idx))
        sqlWindowUI.shellTextBoxDownload.setText(out)
        dropboxID = sqlWindowUI.df.dropboxID[idx]
        midiurl,scoreurl,urlList = getDownloadLinks(dropboxID)
        if midiCheck:
            out += "\n MIDI URL: {}".format(midiurl)
        else:
            urlList.remove(midiurl)

        if scoreCheck:
            out += "\n Score URL: {}".format(scoreurl)
```



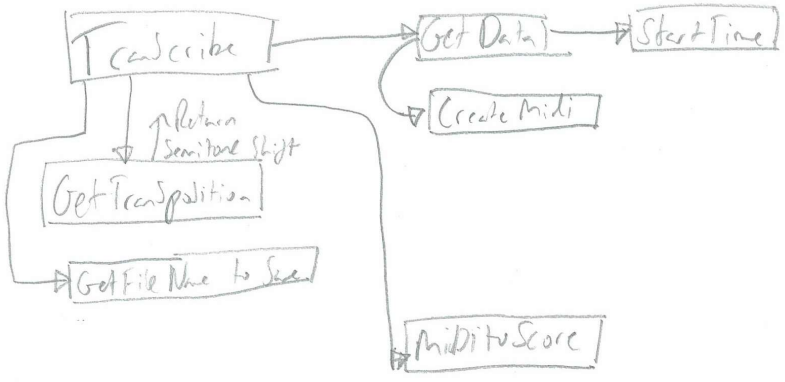
```
        else:
            urlList.remove(scoreurl)


    urlList=urlList
    nameList =
[str(sqlWindowUI.df.UploadName[idx])+"_midiFile.mid",str(sqlWindowUI.df.UploadName[idx]
)+"_ScoreFile.pdf"]
    sqlWindowUI.shellTextBoxDownload.setText(out)
     ##download
    import math
    for url in urlList:
            downloadWindow.show()
            downloadWindowUI.current_status_label.setText("Downloading:
{}".format(url))
            downloadWindowUI.progressBar.setValue(0)
            time.sleep(0.1)
            downloadWindow.update()
            downloadWindowUI.progressBar.setValue(50)
            time.sleep(0.1)
            downloadWindowUI.progressBar.setValue(100)
            print ("DOWNLOADING")
            download(url,nameList[urlList.index(url)])
    time.sleep(1)
    downloadWindow.hide()
```

This function starts by checking which file types have been downloaded and deciding which to download, it then gathers the selected entry by the user in the results table. If no entry or file types have been selected the function halts and raises an error using the raiseError() function. Next it will get the dropboxID from the results table(not visible to the user) and get the links to download by using getDownloadLinks() with dropboxID as the parameter. By doing this it will get the urls of each file it needs to download. Following this it will fetch the binary data from each url through the download function and save it as 'name_fileType.extension'. A download window is also shown while the files are fetched.

download()

```
def download(url,name):
        try:
                req = requests.get(url)
                open(name,"wb").write(req.content)
                raiseError("Files Downloaded",'#77dd77',"Success!")
        except Exception as e:
                raiseError(e)
```

The download function sends requests for the url it receives and saves the content at the



url to the machine. If successful a success message is displayed using raiseError() with additional parameters to change the colour from red to green.

getDownloadLinks()

```
def getDownloadLinks(ID):
        #TYPE = Score_URL or MIDI_URL
        cnxn = sql.start(NGROKPORT)
        QRY = "SELECT [MIDI_URL],[Score_URL] FROM download WHERE [dropboxID] =
({})".format(ID)
        df = sql.query(QRY,cnxn)
        ###sql.stop()
        midiurl = df.MIDI_URL[0]
        scoreurl = df.Score_URL[0]
        urlList = [midiurl,scoreurl]
        return midiurl,scoreurl,urlList
```

This function takes dropboxID and runs a sql query to find the two urls associated with the id and returns the result.

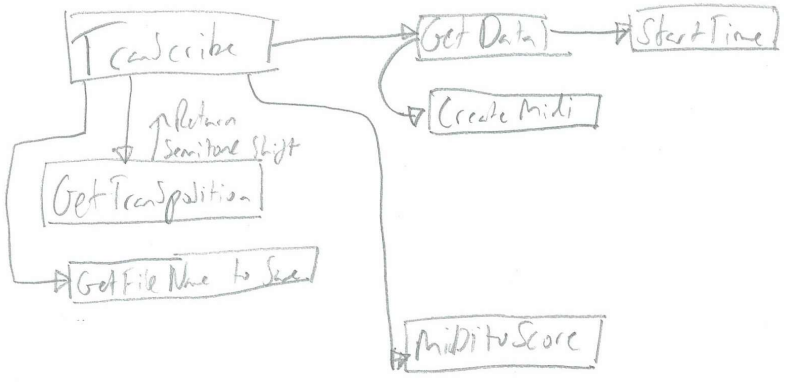
setupSqlSlots()

```
def setupSqlSlots():
    sqlWindowUI.downloadButton.clicked.connect(downloadSelected_clicked)
    sqlWindowUI.searchButton.clicked.connect(searchButton_clicked)
    sqlWindowUI.viewAllButton.clicked.connect(viewAllButton_clicked)
```

This setupSlots function connects the buttons in the sql search window. When the download button is clicked, the downloadSelected_clicked function runs. When the search button is clicked the searchButton_clicked function runs. When the viewAllButton button is clicked the viewAllButton_clicked function is ran.

class Worker()

```
##Recording Worker
class Worker(QObject):
    finished = pyqtSignal()  # our signal out to the main thread to alert it we've
completed our work
    def __init__(self):
        super(Worker, self).__init__()
        self.working = True  # this is our flag to control our loop

    def work(self):
        import pyaudio
        import wave
        global pa
```

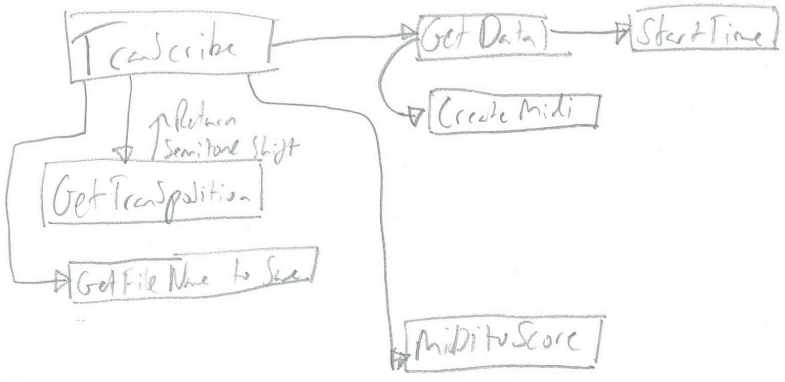


```
        pa = pyaudio.PyAudio()
        DEVICEIDX =
int(open(locateResources.findFile("audioSettingsFile.txt"),"r").readlines()[1].split(",
")[1])
        global stream
        stream = pa.open(format=pyaudio.paFloat32,
        channels=1, rate=44100, input=True,
        frames_per_buffer=1024,input_device_index=DEVICEIDX)
        global frames
        frames = []
        while self.working:
                print("I'm gathering data")
                data = stream.read(1024)
                frames.append(data)

        self.finished.emit() # alert our gui that the loop stopped
```

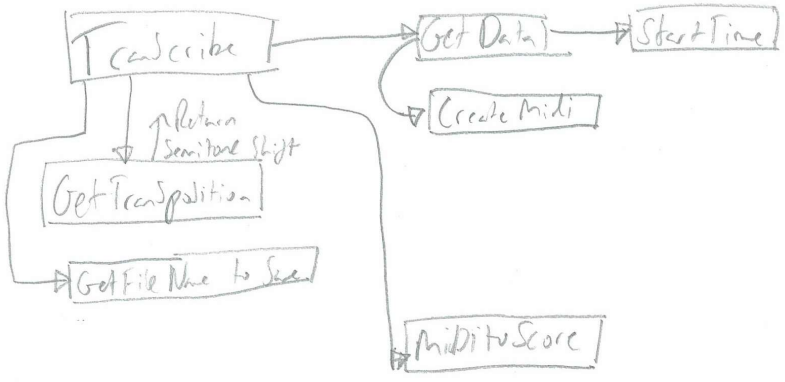
This class uses 'workers' to complete the function work, the work function creates a pyaudio audio stream and collects slices of 1024 bytes from it, it then appends this to the global variable frames. This will continue to happen until it receives a stop signal which will toggle self.working to false and will stop the data gathering.

class Ui_scribeWindow()

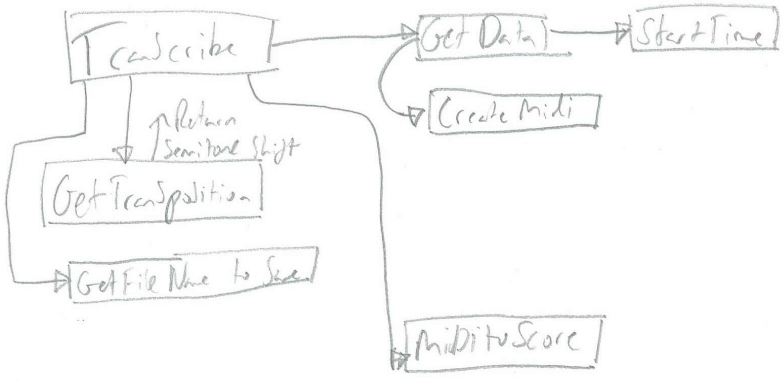
```
##ScribeClass
class Ui_scribeWindow(QObject):
    def setupUi(self, MainWindow):
        global ms,s,m,h
        global scribeTimer
        scribeTimer = timerThread()
        ms,s,m,h = 0,0,0,0
        MainWindow.setObjectName("scribeWindow")
        MainWindow.setFixedSize(759, 403)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")

        self.lcd = QtWidgets.QLCDNumber(self.centralwidget)
        self.lcd.setGeometry(QtCore.QRect(150, 190, 461, 80))
        self.lcd.setObjectName("lcd")
        self.timer = QtCore.QTimer(self.centralwidget)
        time = "{0:02d}:{1:02d}:{2:02d}:{3:03d}".format(h,m,s,ms)
        self.lcd.setDigitCount(len(time))
        self.lcd.display(time)

        self.fileNameTextBox = QtWidgets.QLineEdit(self.centralwidget)
        self.fileNameTextBox.setGeometry(QtCore.QRect(340, 60, 113, 22))
        self.fileNameTextBox.setObjectName("fileNameTextBox")
        self.label = QtWidgets.QLabel(self.centralwidget)
```



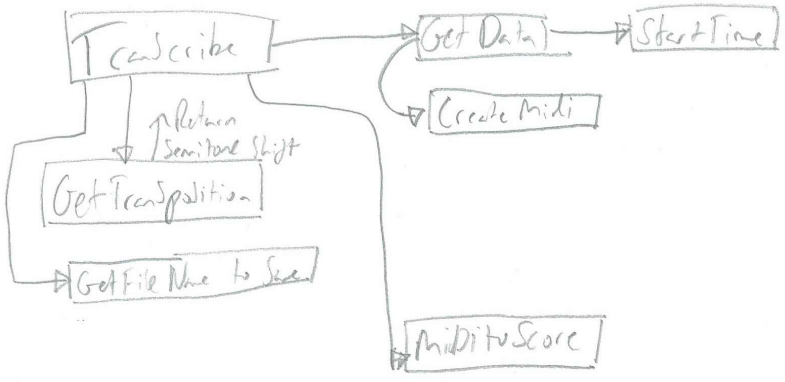
```
        self.label.setGeometry(QtCore.QRect(280, 60, 81, 21))
        self.label.setObjectName("label")
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(280, 90, 55, 16))
        self.label_2.setObjectName("label_2")
        self.userNameTextBox = QtWidgets.QLineEdit(self.centralwidget)
        self.userNameTextBox.setGeometry(QtCore.QRect(340, 90, 113, 22))
        self.userNameTextBox.setObjectName("userNameTextBox")
        self.label_3 = QtWidgets.QLabel(self.centralwidget)
        self.label_3.setGeometry(QtCore.QRect(260, 170, 181, 16))
        self.label_3.setObjectName("label_3")
        self.splitter = QtWidgets.QSplitter(self.centralwidget)
        self.splitter.setGeometry(QtCore.QRect(20, 10, 100, 143))
        self.splitter.setOrientation(QtCore.Qt.Vertical)
        self.splitter.setObjectName("splitter")
        self.startKeyLabel = QtWidgets.QLabel(self.splitter)
        font = QtGui.QFont()
        font.setPointSize(10)
        font.setBold(True)
        font.setWeight(75)
        self.startKeyLabel.setFont(font)
        self.startKeyLabel.setCursor(QtGui.QCursor(QtCore.Qt.ArrowCursor))
        self.startKeyLabel.setObjectName("startKeyLabel")
        self.startKeyDial = QtWidgets.QDial(self.splitter)
        self.startKeyDial.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
        self.startKeyDial.setMinimum(0)
        self.startKeyDial.setMaximum(13)
        self.startKeyDial.setSingleStep(1)
        self.startKeyDial.setPageStep(2)
        self.startKeyDial.setProperty("value", 0)
        self.startKeyDial.setSliderPosition(0)
        self.startKeyDial.setOrientation(QtCore.Qt.Horizontal)
        self.startKeyDial.setWrapping(True)
        self.startKeyDial.setNotchTarget(13.0)
        self.startKeyDial.setNotchesVisible(True)
        self.startKeyDial.setObjectName("startKeyDial")
        self.startKeyCombo = QtWidgets.QComboBox(self.splitter)
        self.startKeyCombo.setEditable(False)
        self.startKeyCombo.setMaxVisibleItems(13)
        self.startKeyCombo.setMinimumContentsLength(1)
        self.startKeyCombo.setDuplicatesEnabled(True)
        self.startKeyCombo.setObjectName("startKeyCombo")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
```



```
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.splitter_2 = QtWidgets.QSplitter(self.centralwidget)
        self.splitter_2.setGeometry(QtCore.QRect(640, 10, 100, 143))
        self.splitter_2.setOrientation(QtCore.Qt.Vertical)
        self.splitter_2.setObjectName("splitter_2")
        self.endKeyLabel = QtWidgets.QLabel(self.splitter_2)
        font = QtGui.QFont()
        font.setPointSize(10)
        font.setBold(True)
        font.setWeight(75)
        self.endKeyLabel.setFont(font)
        self.endKeyLabel.setCursor(QtGui.QCursor(QtCore.Qt.ArrowCursor))
        self.endKeyLabel.setObjectName("endKeyLabel")
        self.endKeyDial = QtWidgets.QDial(self.splitter_2)
        self.endKeyDial.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
        self.endKeyDial.setMinimum(0)
        self.endKeyDial.setMaximum(13)
        self.endKeyDial.setSingleStep(1)
        self.endKeyDial.setPageStep(2)
        self.endKeyDial.setProperty("value", 0)
        self.endKeyDial.setSliderPosition(0)
        self.endKeyDial.setOrientation(QtCore.Qt.Horizontal)
        self.endKeyDial.setWrapping(True)
        self.endKeyDial.setNotchTarget(12.0)
        self.endKeyDial.setNotchesVisible(True)
        self.endKeyDial.setObjectName("endKeyDial")
        self.endKeyCombo = QtWidgets.QComboBox(self.splitter_2)
        self.endKeyCombo.setEditable(False)
        self.endKeyCombo.setMaxVisibleItems(13)
        self.endKeyCombo.setMinimumContentsLength(1)
        self.endKeyCombo.setDuplicatesEnabled(True)
        self.endKeyCombo.setObjectName("endKeyCombo")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.splitter_3 = QtWidgets.QSplitter(self.centralwidget)
```

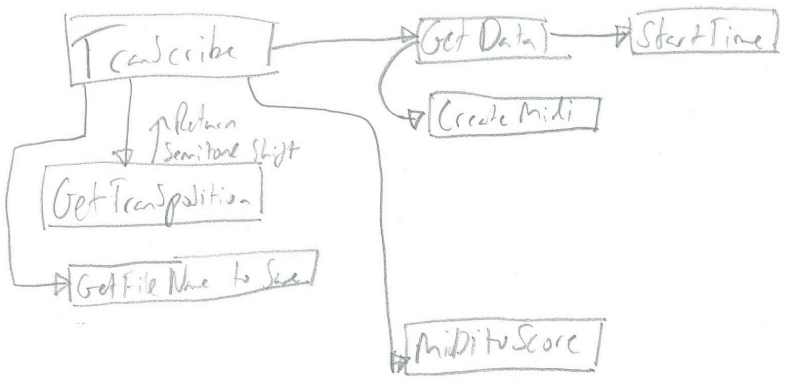


```
        self.splitter_3.setGeometry(QtCore.QRect(70, 280, 621, 61))
        self.splitter_3.setOrientation(QtCore.Qt.Horizontal)
        self.splitter_3.setObjectName("splitter_3")
        self.startButton = QtWidgets.QPushButton(self.splitter_3)
        self.startButton.setObjectName("startButton")
        self.stopButton = QtWidgets.QPushButton(self.splitter_3)
        self.stopButton.setObjectName("stopButton")
        self.resetButton = QtWidgets.QPushButton(self.splitter_3)
        self.resetButton.setObjectName("resetButton")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 759, 26))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.stopButton.setEnabled(False)
        MainWindow.setWindowIcon(QIcon(LOGO_PATH))

        self.retranslateUi(MainWindow)
        self.startKeyCombo.setCurrentIndex(0)
        self.endKeyCombo.setCurrentIndex(0)

self.startKeyDial.valueChanged['int'].connect(self.startKeyCombo.setCurrentIndex)
        self.endKeyDial.valueChanged['int'].connect(self.endKeyCombo.setCurrentIndex)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

        self.thread = None ##add
        self.worker = None ##add

        self.startButton.clicked.connect(self.start_loop)  ##add

    def start_loop(self):
            self.thread = QThread()  # a new thread to run our background tasks in
            self.worker = Worker()  # a new worker to perform those tasks
            self.worker.moveToThread(self.thread)  # move the worker into the thread,
do this first before connecting the signals

            self.thread.started.connect(self.worker.work)  # begin our worker object's
loop when the thread starts running
            self.stopButton.clicked.connect(self.stop_loop)  # stop the loop on the
stop button click
            self.worker.finished.connect(self.loop_finished)  # do something in the gui
when the worker loop ends
            self.worker.finished.connect(self.thread.quit)  # tell the thread it's time
to stop running
            self.worker.finished.connect(self.worker.deleteLater)  # have worker mark
itself for deletion
            self.thread.finished.connect(self.thread.deleteLater)  # have thread mark
```



Commented [7]: the start_loop function sets up threads and workers to collect the audio data(the worker class object). It also connects the stop button to the stop_loop function which stops the workers and threads collecting the data, after the worker has stopped it will then call loop_finished, once finished it will quit the thread and mark itself for deletion. The main part of the function reads the audio device index from audioSettingsFile.txt and then starts the thread and onscreen timer . The stop_loop toggles the self.worker status to False so the worker stops without errors. loop_finished is another important function in this class as it analyses the recorded audio. It first terminates the audio stream to free up the processor and then creates 'recording.wav' which is then filled with the audio data collected by the worker. Finally the save() function is the final step for it to run.

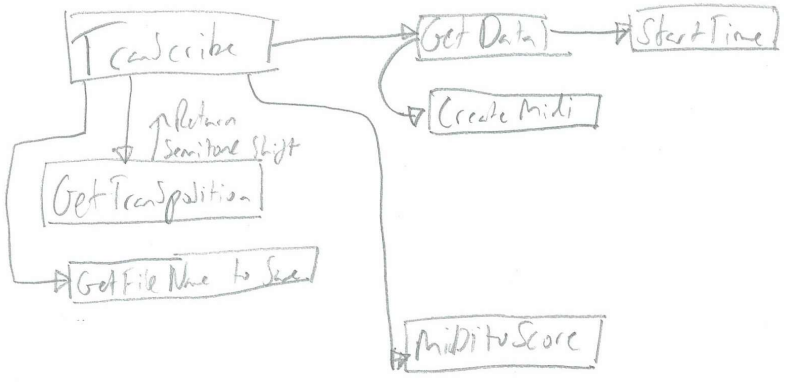
```
itself for deletion
            try:
                    idx =
open("audioSettingsFile.txt","r").readlines()[1].split(",")[1]
                    if idx == "":
                            raiseError("Please set Audio Device in settings")
                    else:
                            print ("LOOP SHOULD START")
                            scribeUI.TimerStart()
                            self.thread.start()
            except FileNotFoundError:
                    raiseError("Please open settings")

    def stop_loop(self):
        self.worker.working = False

    def loop_finished(self):
        import wave
        import pyaudio
        print("I've saved the data")
        stream.stop_stream()
        stream.close()
        pa.terminate()

        waveFile = wave.open("recording.wav", 'wb')
        waveFile.setnchannels(1)
        waveFile.setsampwidth(pa.get_sample_size(pyaudio.paFloat32))
        waveFile.setframerate(44100)
        waveFile.writeframes(b''.join(frames))
        waveFile.close()
        print ("Saving should resume")
        save()

    def TimerReset(self):
        global s,m,h,ms
        s,m,h,ms = 0,0,0,0
        self.timer.stop()
        time = "{0:02d}:{1:02d}:{2:02d}:{3:03d}".format(h,m,s,ms)
        self.lcd.setDigitCount(len(time))
        self.lcd.display(time)

    def TimerStart(self):
        global s,m,h,ms
        scribeTimer.run()

    def TimerTime(self):
        global s,m,h,ms
        if ms < 999:
            ms+=1
        else:
```



Commented [8]: The TimerReset function resets the on screen timer by settings each value to 0. TimerStart runs the scribeTimer Thread which increments the timer in the background. Finally TimerTime correctly ticks the timer by incrementing the seconds when 999milliseconds have passed and the minutes when 59 seconds have passed. If the case happens when the timer surpasses 24 hours the timer is stopped to avoid an overflow and an error is raised saying the recording length limit has been reached.

```
            if s < 59:
                s += 1
                ms = 0
            elif s == 59 and m < 59:
                m+=1
                s = 0
                ms = 0
            else:
                if m < 59:
                    s = 0
                    m += 1
                elif m == 59 and h < 24:
                    h += 1
                    m = 0
                    s = 0
                    ms = 0
                else:
                    self.timer.stop()
                    raiseError("Recording Too Long")

        time = "{0:02d}:{1:02d}:{2:02d}:{3:03d}".format(h,m,s,ms)

        self.lcd.setDigitCount(len(time))
        self.lcd.display(time)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("scribeWindow", "Transcribe"))
        self.label.setText(_translate("scribeWindow", "File Name"))
        self.label_2.setText(_translate("scribeWindow", "User"))
        self.label_3.setText(_translate("scribeWindow", "Current Duration"))
        self.startKeyLabel.setText(_translate("scribeWindow", "Start Key"))
        self.startKeyCombo.setItemText(0, _translate("scribeWindow", "-"))
        self.startKeyCombo.setItemText(1, _translate("scribeWindow", "C"))
        self.startKeyCombo.setItemText(2, _translate("scribeWindow", "C#/Db"))
        self.startKeyCombo.setItemText(3, _translate("scribeWindow", "D"))
        self.startKeyCombo.setItemText(4, _translate("scribeWindow", "D#/Eb"))
        self.startKeyCombo.setItemText(5, _translate("scribeWindow", "E"))
        self.startKeyCombo.setItemText(6, _translate("scribeWindow", "F"))
        self.startKeyCombo.setItemText(7, _translate("scribeWindow", "F#/Gb"))
        self.startKeyCombo.setItemText(8, _translate("scribeWindow", "G"))
        self.startKeyCombo.setItemText(9, _translate("scribeWindow", "G#/Ab"))
        self.startKeyCombo.setItemText(10, _translate("scribeWindow", "A"))
        self.startKeyCombo.setItemText(11, _translate("scribeWindow", "A#/Bb"))
        self.startKeyCombo.setItemText(12, _translate("scribeWindow", "B"))
        self.endKeyLabel.setText(_translate("scribeWindow", "End Key"))
        self.endKeyCombo.setItemText(0, _translate("scribeWindow", "-"))
        self.endKeyCombo.setItemText(1, _translate("scribeWindow", "C"))
        self.endKeyCombo.setItemText(2, _translate("scribeWindow", "C#/Db"))
        self.endKeyCombo.setItemText(3, _translate("scribeWindow", "D"))
```

Commented [9]: need to say what retranslateui function does

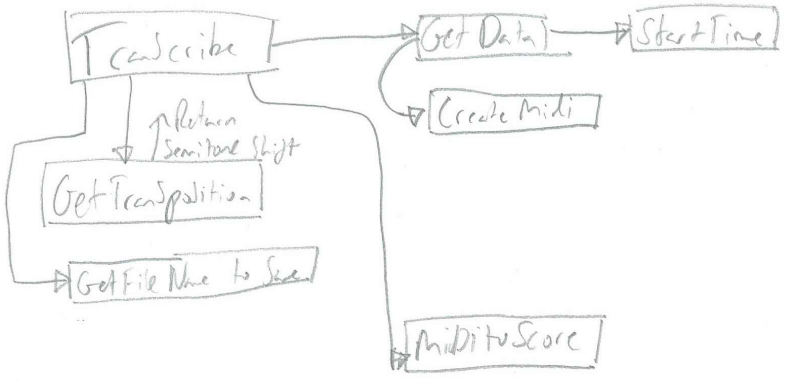


```
        self.endKeyCombo.setItemText(4, _translate("scribeWindow", "D#/Eb"))
        self.endKeyCombo.setItemText(5, _translate("scribeWindow", "E"))
        self.endKeyCombo.setItemText(6, _translate("scribeWindow", "F"))
        self.endKeyCombo.setItemText(7, _translate("scribeWindow", "F#/Gb"))
        self.endKeyCombo.setItemText(8, _translate("scribeWindow", "G"))
        self.endKeyCombo.setItemText(9, _translate("scribeWindow", "G#/Ab"))
        self.endKeyCombo.setItemText(10, _translate("scribeWindow", "A"))
        self.endKeyCombo.setItemText(11, _translate("scribeWindow", "A#/Bb"))
        self.endKeyCombo.setItemText(12, _translate("scribeWindow", "B"))
        self.startButton.setText(_translate("scribeWindow", "Start"))
        self.stopButton.setText(_translate("scribeWindow", "Stop"))
        self.resetButton.setText(_translate("scribeWindow", "Reset"))
```

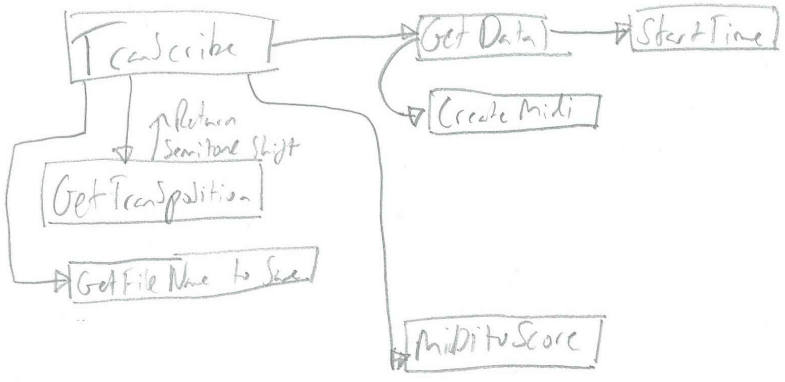
This class is the transcribe window, it contains three buttons, two text inputs and two dial keys. Also visible to the user is a stopwatch which will increment while the recording is taking place. the start_loop function sets up threads and workers to collect the audio data(the worker class object). It also connects the stop button to the stop_loop function which stops the workers and threads collecting the data, after the worker has stopped it will then call loop_finished, once finished it will quit the thread and mark itself for deletion. The main part of the function reads the audio device index from audioSettingsFile.txt and then starts the thread and on screen timer . The stop_loop toggles the self.worker status to False so the worker stops without errors. loop_finished is another important function in this class as it analyses the recorded audio. It first terminates the audio stream to free up the processor and then creates 'recording.wav' which is then filled with the audio data collected by the worker. Finally the save() function is the final step for it to run. The TimerReset function resets the on screen timer by setting each value to 0. TimerStart runs the scribeTimer Thread which increments the timer in the background. Finally TimerTime correctly ticks the timer by incrementing the seconds when 999 milliseconds have passed and the minutes when 59 seconds have passed. If the case happens when the timer surpasses 24 hours the timer is stopped to avoid an overflow and an error is raised saying the recording length limit has been reached.

class Ui_Dialog()

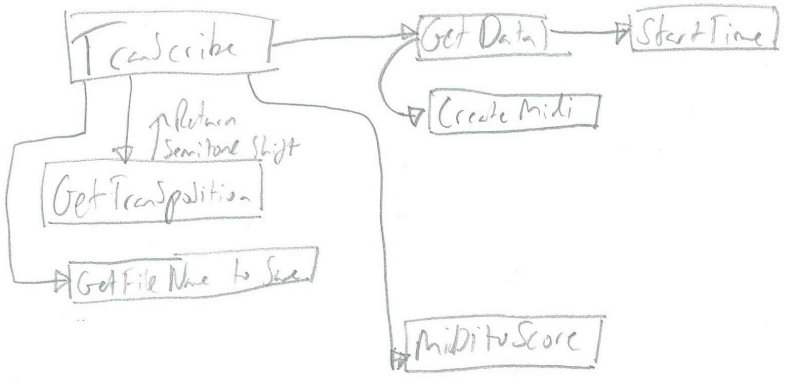
```
##TransClass
class Ui_Dialog(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.setFixedSize(557, 359)
        self.sensitivityDataTxt = QtWidgets.QLabel(Dialog)
        self.sensitivityDataTxt.setGeometry(QtCore.QRect(190, 130, 71, 21))
        self.sensitivityDataTxt.setFrameShape(QtWidgets.QFrame.WinPanel)
```



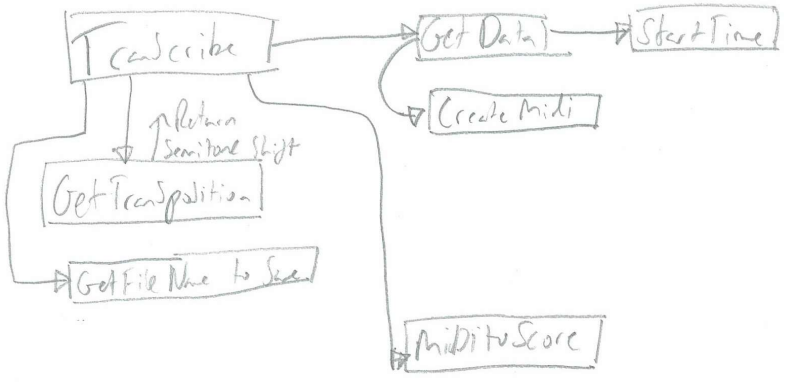
```
        self.sensitivityDataTxt.setWordWrap(False)
        self.sensitivityDataTxt.setObjectName("sensitivityDataTxt")
        self.startKeyLabel = QtWidgets.QLabel(Dialog)
        self.startKeyLabel.setGeometry(QtCore.QRect(50, 30, 101, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        font.setBold(True)
        font.setWeight(75)
        self.startKeyLabel.setFont(font)
        self.startKeyLabel.setCursor(QtGui.QCursor(QtCore.Qt.ArrowCursor))
        self.startKeyLabel.setObjectName("startKeyLabel")
        self.endKeyLabel = QtWidgets.QLabel(Dialog)
        self.endKeyLabel.setGeometry(QtCore.QRect(50, 60, 101, 21))
        font = QtGui.QFont()
        font.setPointSize(10)
        font.setBold(True)
        font.setWeight(75)
        self.endKeyLabel.setFont(font)
        self.endKeyLabel.setCursor(QtGui.QCursor(QtCore.Qt.ArrowCursor))
        self.endKeyLabel.setObjectName("endKeyLabel")
        self.changeSensitivityLabel = QtWidgets.QLabel(Dialog)
        self.changeSensitivityLabel.setGeometry(QtCore.QRect(20, 130, 181, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        font.setBold(True)
        font.setWeight(75)
        self.changeSensitivityLabel.setFont(font)
        self.changeSensitivityLabel.setObjectName("changeSensitivityLabel")
        self.currentNoteLabel = QtWidgets.QLabel(Dialog)
        self.currentNoteLabel.setGeometry(QtCore.QRect(420, 190, 181, 21))
        font = QtGui.QFont()
        font.setPointSize(10)
        font.setBold(True)
        font.setWeight(75)
        self.currentNoteLabel.setFont(font)
        self.currentNoteLabel.setObjectName("currentNoteLabel")
        self.currentNoteUpdateLabel = QtWidgets.QLabel(Dialog)
        self.currentNoteUpdateLabel.setGeometry(QtCore.QRect(455, 210, 111, 81))
###dev
        self.currentInitialNoteLabel = QtWidgets.QLabel(Dialog)
        self.currentInitialNoteLabel.setGeometry(QtCore.QRect(330, 190, 181, 21))
        self.currentInitialNoteLabel.setFont(font)
        self.currentInitialNoteLabel.setObjectName("currentInitialNoteLabel")
        self.currentInitialNoteUpdateLabel = QtWidgets.QLabel(Dialog)
        self.currentInitialNoteUpdateLabel.setGeometry(QtCore.QRect(350, 210, 111, 81))
        font = QtGui.QFont()
        font.setPointSize(30)
        self.currentInitialNoteUpdateLabel.setFont(font)

self.currentInitialNoteUpdateLabel.setObjectName("currentInitialNoteUpdateLabel")
```



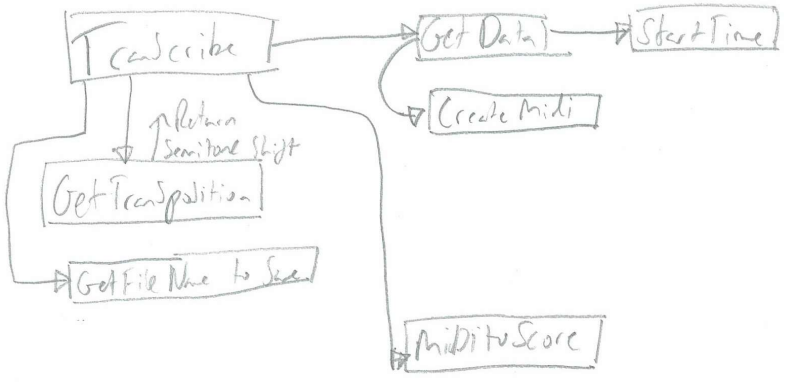
```
        font = QtGui.QFont()
        font.setPointSize(30)
        self.currentNoteUpdateLabel.setFont(font)
        self.currentNoteUpdateLabel.setObjectName("currentNoteUpdateLabel")
        self.verticalLayoutWidget = QtWidgets.QWidget(Dialog)
        self.verticalLayoutWidget.setGeometry(QtCore.QRect(19, 159, 251, 181))
        self.verticalLayoutWidget.setObjectName("verticalLayoutWidget")
        self.verticalLayout = QtWidgets.QVBoxLayout(self.verticalLayoutWidget)
        self.verticalLayout.setContentsMargins(0, 0, 0, 0)
        self.verticalLayout.setObjectName("verticalLayout")
        self.currentNoteImage = QtWidgets.QLabel(self.verticalLayoutWidget)
        self.currentNoteImage.setText("")
        self.currentNoteImage.setObjectName("currentNoteImage")
        self.verticalLayout.addWidget(self.currentNoteImage)
        self.splitter = QtWidgets.QSplitter(Dialog)
        self.splitter.setGeometry(QtCore.QRect(290, 10, 200, 100))
        self.splitter.setOrientation(QtCore.Qt.Horizontal)
        self.splitter.setObjectName("splitter")
        self.startKeyDial = QtWidgets.QDial(self.splitter)
        self.startKeyDial.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
        self.startKeyDial.setMinimum(0)
        self.startKeyDial.setMaximum(13)
        self.startKeyDial.setSingleStep(1)
        self.startKeyDial.setPageStep(2)
        self.startKeyDial.setProperty("value", 0)
        self.startKeyDial.setSliderPosition(0)
        self.startKeyDial.setOrientation(QtCore.Qt.Horizontal)
        self.startKeyDial.setWrapping(True)
        self.startKeyDial.setNotchTarget(13.0)
        self.startKeyDial.setNotchesVisible(True)
        self.startKeyDial.setObjectName("startKeyDial")
        self.endKeyDial = QtWidgets.QDial(self.splitter)
        self.endKeyDial.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
        self.endKeyDial.setMinimum(0)
        self.endKeyDial.setMaximum(13)
        self.endKeyDial.setSingleStep(1)
        self.endKeyDial.setPageStep(2)
        self.endKeyDial.setProperty("value", 0)
        self.endKeyDial.setSliderPosition(0)
        self.endKeyDial.setOrientation(QtCore.Qt.Horizontal)
        self.endKeyDial.setWrapping(True)
        self.endKeyDial.setNotchTarget(12.0)
        self.endKeyDial.setNotchesVisible(True)
        self.endKeyDial.setObjectName("endKeyDial")
        self.splitter_2 = QtWidgets.QSplitter(Dialog)
        self.splitter_2.setGeometry(QtCore.QRect(150, 30, 121, 61))
        self.splitter_2.setOrientation(QtCore.Qt.Vertical)
        self.splitter_2.setObjectName("splitter_2")
```



```
        self.startKeyCombo = QtWidgets.QComboBox(self.splitter_2)
        self.startKeyCombo.setEditable(False)
        self.startKeyCombo.setMaxVisibleItems(13)
        self.startKeyCombo.setMinimumContentsLength(1)
        self.startKeyCombo.setDuplicatesEnabled(True)
        self.startKeyCombo.setObjectName("startKeyCombo")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.startKeyCombo.addItem("")
        self.endKeyCombo = QtWidgets.QComboBox(self.splitter_2)
        self.endKeyCombo.setEditable(False)
        self.endKeyCombo.setMaxVisibleItems(13)
        self.endKeyCombo.setMinimumContentsLength(1)
        self.endKeyCombo.setDuplicatesEnabled(True)
        self.endKeyCombo.setObjectName("endKeyCombo")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.splitter_3 = QtWidgets.QSplitter(Dialog)
        self.splitter_3.setGeometry(QtCore.QRect(280, 130, 241, 32))
        self.splitter_3.setOrientation(QtCore.Qt.Horizontal)
        self.splitter_3.setObjectName("splitter_3")
        self.label_2 = QtWidgets.QLabel(self.splitter_3)
        self.label_2.setAutoFillBackground(False)
        self.label_2.setObjectName("label_2")
        self.sensitivitySlider = QtWidgets.QSlider(self.splitter_3)
        self.sensitivitySlider.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
        self.sensitivitySlider.setMinimum(10)
        self.sensitivitySlider.setMaximum(100)
        self.sensitivitySlider.setSingleStep(10)
```



```
        self.sensitivitySlider.setPageStep(1)
        self.sensitivitySlider.setProperty("value", 55)
        self.sensitivitySlider.setSliderPosition(55)
        self.sensitivitySlider.setOrientation(QtCore.Qt.Horizontal)
        self.sensitivitySlider.setInvertedAppearance(True)
        self.sensitivitySlider.setInvertedControls(False)
        self.sensitivitySlider.setTickPosition(QtWidgets.QSlider.TicksBothSides)
        self.sensitivitySlider.setTickInterval(20)
        self.sensitivitySlider.setObjectName("sensitivitySlider")
        self.label = QtWidgets.QLabel(self.splitter_3)
        self.label.setObjectName("label")
        self.splitter_4 = QtWidgets.QSplitter(Dialog)
        self.splitter_4.setGeometry(QtCore.QRect(300, 290, 251, 61))
        self.splitter_4.setOrientation(QtCore.Qt.Horizontal)
        self.splitter_4.setObjectName("splitter_4")
        self.startButton = QtWidgets.QPushButton(self.splitter_4)
        self.startButton.setObjectName("startButton")
        self.stopButton = QtWidgets.QPushButton(self.splitter_4)
        self.stopButton.setObjectName("stopButton")
        self.stopButton.setEnabled(False)
        self.retranslateUi(Dialog)
        self.startKeyCombo.setCurrentIndex(0)
        self.endKeyCombo.setCurrentIndex(0)
        self.endKeyDial.valueChanged['int'].connect(self.endKeyCombo.setCurrentIndex)

self.startKeyDial.valueChanged['int'].connect(self.startKeyCombo.setCurrentIndex)

self.sensitivitySlider.valueChanged['int'].connect(self.sensitivityDataTxt.setNum)

self.startKeyCombo.currentIndexChanged['int'].connect(self.startKeyDial.setValue)
        QtCore.QMetaObject.connectSlotsByName(Dialog)

    def retranslateUi(self, Dialog):
        _translate = QtCore.QCoreApplication.translate
        Dialog.setWindowTitle(_translate("Dialog", "Transpose"))
        self.sensitivityDataTxt.setText(_translate("Dialog", "55"))
        self.startKeyLabel.setText(_translate("Dialog", "Start Key"))
        self.endKeyLabel.setText(_translate("Dialog", "End Key"))
        self.changeSensitivityLabel.setText(_translate("Dialog", "Change Sensitivity"))
        self.currentNoteLabel.setText(_translate("Dialog", "Current Note"))
        self.currentNoteUpdateLabel.setText(_translate("Dialog", "-"))
        self.currentInitialNoteLabel.setText(_translate("Dialog", "Initial Note"))
        self.currentInitialNoteUpdateLabel.setText(_translate("Dialog", "-"))
        self.startKeyCombo.setItemText(0, _translate("Dialog", "-"))
        self.startKeyCombo.setItemText(1, _translate("Dialog", "C"))
        self.startKeyCombo.setItemText(2, _translate("Dialog", "C#/Db"))
        self.startKeyCombo.setItemText(3, _translate("Dialog", "D"))
        self.startKeyCombo.setItemText(4, _translate("Dialog", "D#/Eb"))
        self.startKeyCombo.setItemText(5, _translate("Dialog", "E"))
        self.startKeyCombo.setItemText(6, _translate("Dialog", "F"))
```

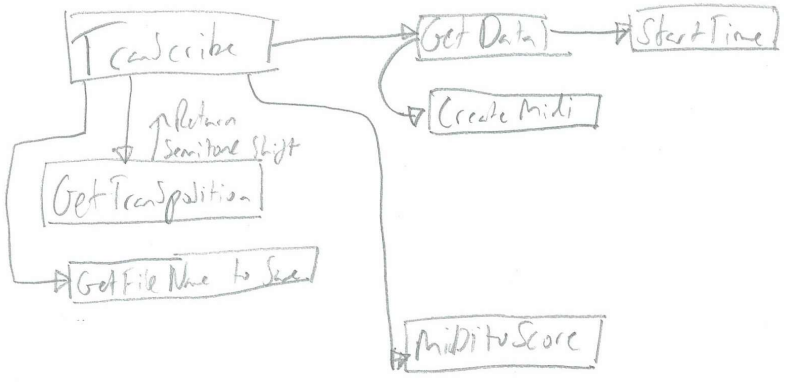


```
        self.startKeyCombo.setItemText(7, _translate("Dialog", "F#/Gb"))
        self.startKeyCombo.setItemText(8, _translate("Dialog", "G"))
        self.startKeyCombo.setItemText(9, _translate("Dialog", "G#/Ab"))
        self.startKeyCombo.setItemText(10, _translate("Dialog", "A"))
        self.startKeyCombo.setItemText(11, _translate("Dialog", "A#/Bb"))
        self.startKeyCombo.setItemText(12, _translate("Dialog", "B"))
        self.endKeyCombo.setItemText(0, _translate("Dialog", "-"))
        self.endKeyCombo.setItemText(1, _translate("Dialog", "C"))
        self.endKeyCombo.setItemText(2, _translate("Dialog", "C#/Db"))
        self.endKeyCombo.setItemText(3, _translate("Dialog", "D"))
        self.endKeyCombo.setItemText(4, _translate("Dialog", "D#/Eb"))
        self.endKeyCombo.setItemText(5, _translate("Dialog", "E"))
        self.endKeyCombo.setItemText(6, _translate("Dialog", "F"))
        self.endKeyCombo.setItemText(7, _translate("Dialog", "F#/Gb"))
        self.endKeyCombo.setItemText(8, _translate("Dialog", "G"))
        self.endKeyCombo.setItemText(9, _translate("Dialog", "G#/Ab"))
        self.endKeyCombo.setItemText(10, _translate("Dialog", "A"))
        self.endKeyCombo.setItemText(11, _translate("Dialog", "A#/Bb"))
        self.endKeyCombo.setItemText(12, _translate("Dialog", "B"))
        self.label_2.setText(_translate("Dialog", "-"))
        self.label.setText(_translate("Dialog", "+"))
        self.startButton.setText(_translate("Dialog", "Start"))
        self.stopButton.setText(_translate("Dialog", "Stop"))
```

This class creates the transposition window, it contains four text lines, two key dials, two dropdown boxes, two buttons, a sensitivity slider and an image. The image is where the score of the transposed note will appear, and the slider for adjusting how many data points the program takes before choosing the note.

class Ui_settingsWindow()

```
class Ui_settingsWindow(object):
    def setupUi(self, settingsWindow):
        settingsWindow.setObjectName("settingsWindow")
        settingsWindow.setFixedSize(394, 135)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(settingsWindow.sizePolicy().hasHeightForWidth())
        settingsWindow.setSizePolicy(sizePolicy)
        self.audioInputDeviceLabel = QtWidgets.QLabel(settingsWindow)
        self.audioInputDeviceLabel.setGeometry(QtCore.QRect(50, 40, 151, 21))
        self.audioInputDeviceLabel.setObjectName("audioInputDeviceLabel")
        self.applyAudioSettingsButton = QtWidgets.QPushButton(settingsWindow)
        self.applyAudioSettingsButton.setGeometry(QtCore.QRect(60, 80, 93, 28))
```

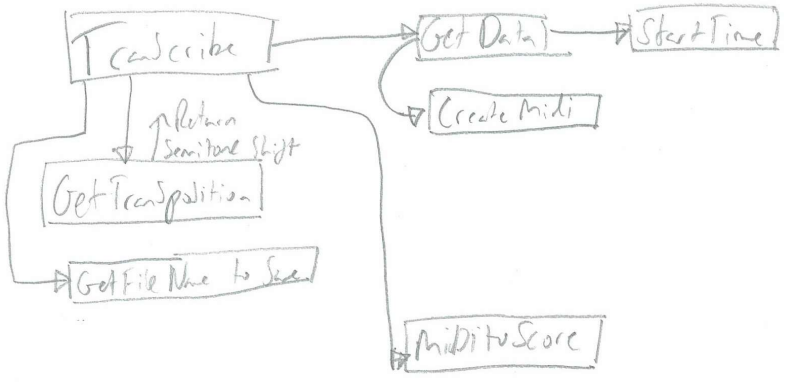


```
        font = QtGui.QFont()
        font.setPointSize(7)
        self.applyAudioSettingsButton.setFont(font)
        self.applyAudioSettingsButton.setObjectName("applyAudioSettingsButton")
        self.refreshAudioDevicesBUtton = QtWidgets.QPushButton(settingsWindow)
        self.refreshAudioDevicesBUtton.setGeometry(QtCore.QRect(230, 80, 111, 28))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.refreshAudioDevicesBUtton.sizePolicy().hasHeightForWi
dth())
        self.refreshAudioDevicesBUtton.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setStyleStrategy(QtGui.QFont.PreferAntialias)
        self.refreshAudioDevicesBUtton.setFont(font)
        self.refreshAudioDevicesBUtton.setObjectName("refreshAudioDevicesBUtton")
        self.audioDeviceComboBox = QtWidgets.QComboBox(settingsWindow)
        self.audioDeviceComboBox.setGeometry(QtCore.QRect(190, 40, 161, 22))
        self.audioDeviceComboBox.setObjectName("audioDeviceComboBox")

        self.retranslateUi(settingsWindow)
        QtCore.QMetaObject.connectSlotsByName(settingsWindow)

    def retranslateUi(self, settingsWindow):
        _translate = QtCore.QCoreApplication.translate
        settingsWindow.setWindowTitle(_translate("settingsWindow", "Settings"))
        self.audioInputDeviceLabel.setText(_translate("settingsWindow", "Audio Input
Device:"))
        self.applyAudioSettingsButton.setText(_translate("settingsWindow", "Apply"))
        self.refreshAudioDevicesBUtton.setText(_translate("settingsWindow", "Refresh
Devices"))
```

This window is the settings window and is only for selecting the audio interface to be used, it does this by showing a list of available devices in a dropdown box and being able to apply it with an 'Apply' button. There is also a refresh button which will check for new audio devices that have been connected or removed.

class Ui_downloadWindow()

```
class Ui_downloadWindow(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName("Dialog")
        Dialog.setFixedSize(343, 115)
        self.current_status_label = QtWidgets.QLabel(Dialog)
```



```
        self.current_status_label.setGeometry(QtCore.QRect(20, 20, 271, 16))
        self.current_status_label.setObjectName("current_status_label")
        self.progressBar = QtWidgets.QProgressBar(Dialog)
        self.progressBar.setGeometry(QtCore.QRect(20, 60, 311, 31))
        self.progressBar.setSizeIncrement(QtCore.QSize(10, 0))
        self.progressBar.setProperty("value", 0)
        self.progressBar.setInvertedAppearance(False)
        self.progressBar.setObjectName("progressBar")

        self.retranslateUi(Dialog)
        QtCore.QMetaObject.connectSlotsByName(Dialog)

    def retranslateUi(self, Dialog):
        _translate = QtCore.QCoreApplication.translate
        Dialog.setWindowTitle(_translate("Dialog", "Dialog"))
        self.current_status_label.setText(_translate("Dialog", "Downloading: "))
```

This class creates the download Ui, it contains a progress bar and a label stating the percentage.

###Slots

getStartKeyData()

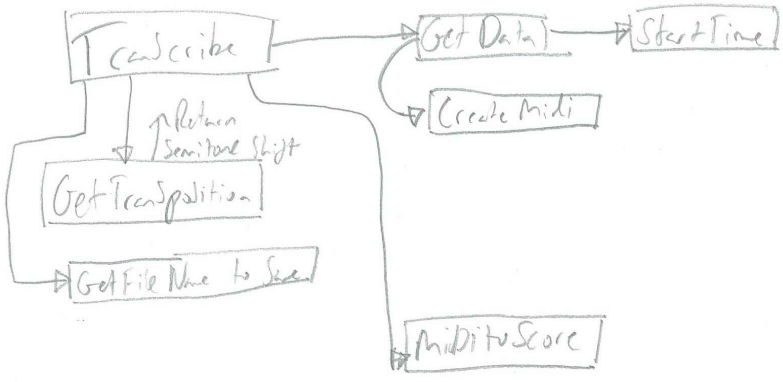
```
@pyqtSlot()
def getStartKeyData():
        print ("Start Key: {}".format(transUI.startKeyCombo.currentText()))
```

This function displays on the screen which key has been selected by the user as the starting key, it gets the key by getting the current set text of the dropdown box as it is linked to the key dial.

getEndKeyData()

```
@pyqtSlot()
def getEndKeyData():
        print ("End Key: {}".format(transUI.endKeyCombo.currentText()))
```

This function displays on the screen which key has been selected by the user as the ending key, it gets the key by getting the current set text of the dropdown box as it is linked to the key dial.



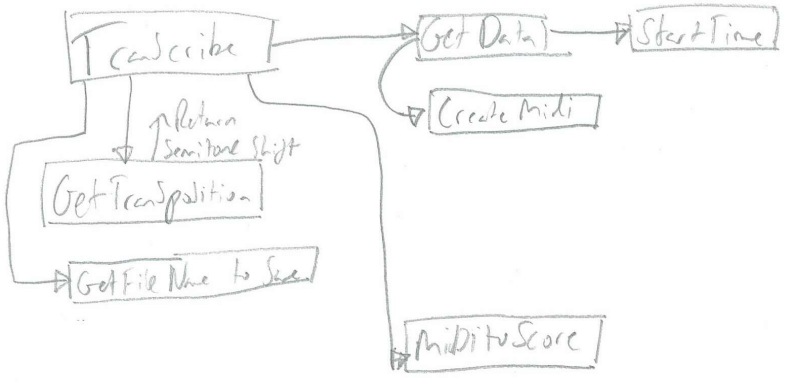
stopButton_clicked()

```
@pyqtSlot()
def stopButton_clicked():
        print ("STOP BUTTON")
        transUI.stopButton.setEnabled(False)
        transUI.startKeyCombo.setEnabled(True)
        transUI.endKeyCombo.setEnabled(True)
        transUI.endKeyDial.setEnabled(True)
        transUI.startKeyDial.setEnabled(True)
        transUI.sensitivitySlider.setEnabled(True)
        transUI.startButton.setEnabled(True)
        try:
                print ("Attempting to stop threads")
                liveAudioIdentification.join()
                print ("Stopped audio identification thread")
                time.sleep(0.5)
                print ("Attempting to stop transAudio Thread")
                transAudioTR.terminate()
                print ("Stopped transAudio Thread")
        except Exception as e:
                raiseError("StopButton Error")
                print (e)
```

This pyqtslot function is ran when the stop button is clicked on the transpose Ui. It reenables the inputs for the user and then stops the running threads one at a time starting with liveAudioIdentification then the audio feed, transAudioTR. If the threads fail to terminate an error is raised with the details. The reason it will wait half a second is because transAudioTR contains the liveAudioIdentification thread so an error will occur.

convertNoteToOnlySharps()

```
def convertNoteToOnlySharps(note):
        if note == "B#":
                return "C"
        elif note == "Eb":
                return "D#"
        elif note =="Ab":
                return "G#"
        elif note =="Bb":
                return "A#"
        elif note =="Cb":
                return "B"
        elif note =="Db":
                return "C#"
        elif note == "Fb":
```

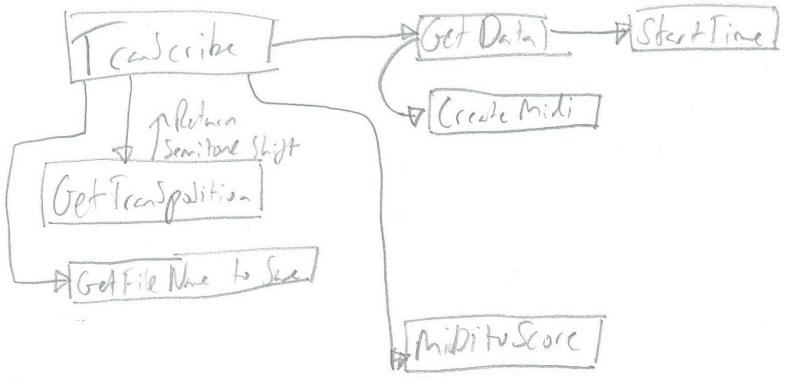


```
                return "E"
        elif note == "Gb":
                return "F#"
        else:
                return note
```

This function turns any notes which contain flat symbols 'b' if there are any to its relative note with sharps '#'. This is because the program only knows how to deal with sharps.

transposeNote()

```
def transposeNote(note):
        transpositionValue = calculateTransposition(transUI,keys)[1]
        note = convertNoteToOnlySharps(note)
        startIdx = keys.index(note)
        if startIdx == None:
                raiseError("[e] starIdx is None")
        if transpositionValue*-1 >0:
                transpositionValue = len(keys)+transpositionValue
        while transpositionValue>len(keys):
                transpositionValue-=len(keys)
        transpositionValue=transpositionValue+startIdx
        try:
                newNote = keys[transpositionValue]
        except:
                if transpositionValue>len(keys):
                        transpositionValue=transpositionValue-len(keys)
                elif transpositionValue*-1>0:
                        transpositionValue = len(keys)+transpositionValue

                try:
                        newNote = keys[transpositionValue]
                except Exception as e:
                        print ("THAT G#")
                        print (e)
                        print (len(keys))
                        print
(note,calculateTransposition(transUI,keys)[1],transpositionValue)
                        if transpositionValue == 12:
                                transpositionValue == 0
                                print ("MANUALLY CHANGE TRANSPO VALUE")


        try:
                return newNote
        except UnboundLocalError:
                return newNote
```



transposeNote() takes an input of a note and returns the transposed counterpart which is specified by the user, it first gets the value to transpose by from the function calculateTransposition and then removes any flat symbols from it using convertNoteToOnlySharps, then it reduces or increases the transposition value until it falls within the length of the list(keys). This is so there are no Index Errors and the relative note is referenceable. The newNote is then obtained by getting the note at the transposed index, and is returned.

updateNoteLabel()

```
@pyqtSlot()
def updateNoteLabel(note):
        transUI.currentInitialNoteUpdateLabel.setText(note[0].upper()+note[1:])
        ##transpose into next key
        note = transposeNote(note)
        path = locateResources.findFile(note+".png")
        transUI.currentNoteUpdateLabel.setText(note.upper())
        notePicture = QPixmap(path)
        notePicture = notePicture.scaled(250, 250)
        transUI.currentNoteImage.setPixmap(notePicture)
```

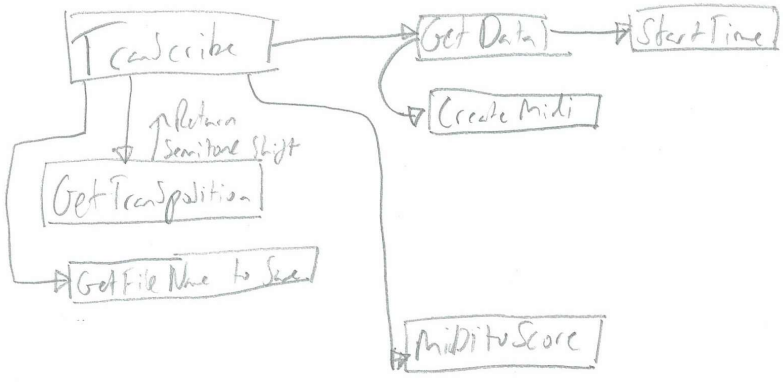
This function first converts the note string into a specified format of /([A-Z]){1}[#]?)/ or one uppercase letter followed by one or zero '#' symbols. The transposed counterpart is then returned. The function locateResources is called to find the path to the image for the current note. This is then set visible to the user by making a Pixmap of it and attaching it to the transpose windows image space, at the same time the text version of the note is displayed.

getSensitivityData()

```
@pyqtSlot()
def getSensitivityData():
    print ("Sensitivity: {}".format(transUI.sensitivityDataTxt.text()))
```

This pyqt slot function displays the current value set by the sensitivity slider every time it is changed.

returnSensitivityData()



```
def returnSensitivityData():
        return transUI.sensitivityDataTxt.text()
```

This function returns the current value set by the sensitivity slider every time it is changed.

startButton_clicked()

```
@pyqtSlot()
def startButton_clicked():
    if calculateTransposition(transUI,keys)[0] == True:
        print ("START BUTTON")
        transUI.startButton.setEnabled(False)
        transUI.startKeyCombo.setEnabled(False)
        transUI.endKeyCombo.setEnabled(False)
        transUI.endKeyDial.setEnabled(False)
        transUI.startKeyDial.setEnabled(False)
        transUI.sensitivitySlider.setEnabled(False)
        transUI.stopButton.setEnabled(True)
        try:
                if open("audioSettingsFile.txt","r").readlines()[1].split(",")[1] ==
"":
                        raiseError("No Audio Device Selected, Open Settings")
                transAudioTR.start()
        except FileNotFoundError:
                raiseError("Please adjust settings")
    else:
        raiseError("Transposition Keys Not Defined")
```

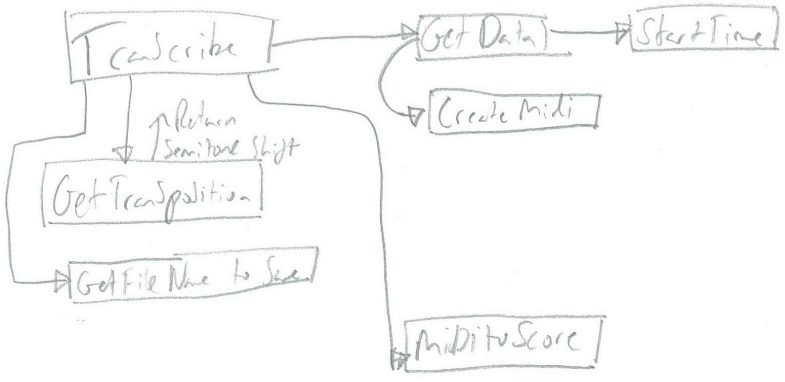
This pyqt slot function is run when the start button is pressed in the transpose window, it disables all user inputs and then reads the audio device to use from audioSettingsFile.txt and then starts the transAudioTR thread. Otherwise an error is raised depending on if an audio device hasn't been selected or other inputs haven't been adjusted.

transposeButton_clicked()

```
@pyqtSlot()
def transposeButton_clicked():
    transposeWindow.show()
```

This pyqt slot function is run when the transpose option button is clicked in the main window of the app, it will then make the transpose window visible to the user.

transcribeButton_clicked()



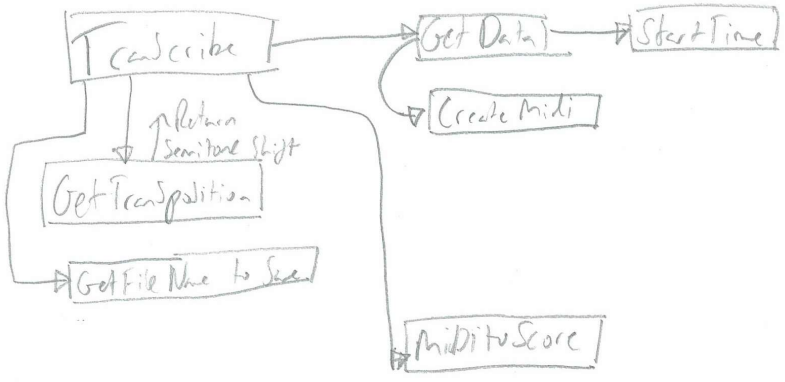
```
@pyqtSlot()
def transcribeButton_clicked():
    transcribeWindow.show()
```

This pyqt slot function is run when the transcribe option button is clicked in the main window of the app, it will then make the transcribe window visible to the user.

transcribeStartButton_clicked()

```
@pyqtSlot()
def transcribeStartButton_clicked():
    if calculateTransposition(scribeUI,keys)[0]:
            if scribeUI.userNameTextBox.text() != "":
                    if scribeUI.fileNameTextBox.text() != "":
                            scribeUI.startButton.setEnabled(False)
                            scribeUI.stopButton.setEnabled(True)
                            scribeUI.resetButton.setEnabled(False)
                            scribeUI.startKeyDial.setEnabled(False)
                            scribeUI.endKeyDial.setEnabled(False)
                            scribeUI.startKeyCombo.setEnabled(False)
                            scribeUI.endKeyCombo.setEnabled(False)
                            scribeUI.fileNameTextBox.setEnabled(False)
                            scribeUI.userNameTextBox.setEnabled(False)
                    else:
                            raiseError("File Name Required")
            else:
                    if scribeUI.fileNameTextBox.text() =="":
                            raiseError("File Name Required")
                    else:
                            raiseError("Username Required")


    else:
            try:
                    idx =
open("audioSettingsFile.txt","r").readlines()[1].split(",")[1]
                    if idx == "":
                            raiseError("Please set Audio Device in settings")
                    else:
                            print ("LOOP SHOULD START")
            except FileNotFoundError:
                    raiseError("Please open settings")
```

This pyqt slot function checks if any required settings are not set by the user and raises an appropriate error message if they haven't been set. If all settings have been given a value it then gets the audio device to use from audioSettingsFile.txt and raises an appropriate



error message if that is also not set.

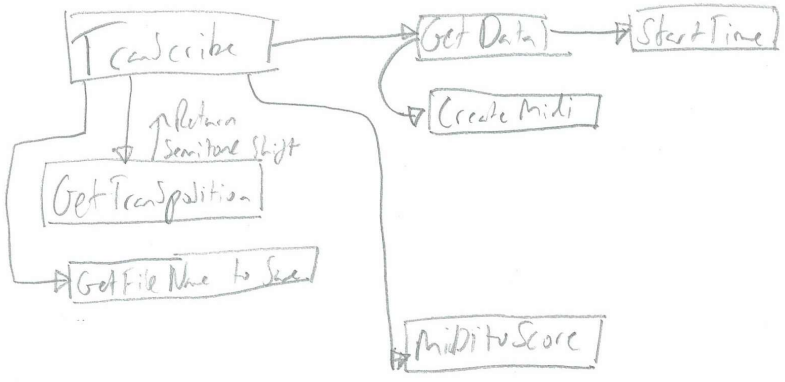
generateDropboxHash()

```
def generateDropboxHash(url,name,cnxn):
        unique = False
        name = "/midi/"+name+"_midiFile.mid"
        tempurl = cnxn.files_get_temporary_link(name).link.split("/")[5]
        tempurl = tempurl.encode("utf-8")
        part1 = tempurl[-5:]
        part2 = tempurl[-10:-5]
        total1 = 0
        total2 = 0
        for char in part1:
                total1+=char
        for char in part2:
                total2 +=char
        HASH = total1*total2
        unique = sql.checkHash(HASH,"dropboxID",sql.start(NGROKPORT))#check if exists
        while unique == False:
                HASH = HASH + 1
                unique = sql.checkHash(HASH,"dropboxID",sql.start(NGROKPORT))
        return HASH
```

This function creates a hash given the url and the name of the file. It first gets the url of the file from the dropbox connection and extracts the last ten characters of it, splitting them into two groups of five. Then for each group of five it adds up the values and multiplies it by the other group. This creates an initial hash for the dropboxID, then it will use the function checkHash to see if this is unique, if it isn't it will use linear probing to find the next available hash. The hash value is then returned.

generateUploadHash()

```
def generateUploadHash(url,name,cnxn):
        unique = False
        name = "/score/"+name+"_scoreFile.pdf"
        tempurl = cnxn.files_get_temporary_link(name).link.split("/")[5]
        tempurl = tempurl.encode("utf-8")
        part1 = tempurl[-5:]
        part2 = tempurl[-10:-5]
        total1 = 0
        total2 = 0
        for char in part1:
```



```
                total1+=char
        for char in part2:
                total2+=char
        HASH = total1*total2
        unique = sql.checkHash(HASH,"uploadID",sql.start(NGROKPORT))#check if exists
        while unique == False:
                HASH = HASH + 1
                unique = sql.checkHash(HASH,"uploadID",sql.start(NGROKPORT))
        return HASH
```

This function creates a hash given the url and the name of the file. It first gets the url of the file from the dropbox connection and extracts the last ten characters of it, splitting them into two groups of five. Then for each group of five it adds up the values and multiplies it by the other group. This creates an initial hash for the uploadID, then it will use the function checkHash to see if this is unique, if it isn't it will use linear probing to find the next available hash. The hash value is then returned.

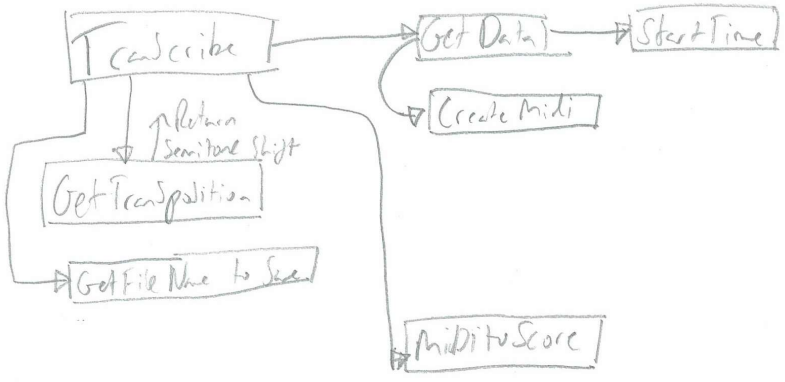
checkUniqueName()

```
def checkUniqueName(name):
        unique = False
        cnxn = sql.connect_DB()
        try:
                name = "/score/"+name+"_scoreFile.pdf"
                cnxn.files_get_temporary_link(name)
                unique = False
        except:
                unique = True
        return unique
```

This function checks if any other files have already been created an uploaded to the filehost with that name, it first establishes a connection to the dropbox and then scans the score directory for the specified file. If it exists the name isn't unique so the user will have to choose another.

debugTrace()

```
def debugTrace():
    from PyQt5.QtCore import pyqtRemoveInputHook, pyqtRestoreInputHook
    import pdb
    import sys
    pyqtRemoveInputHook()
    try:
        debugger = pdb.Pdb()
```

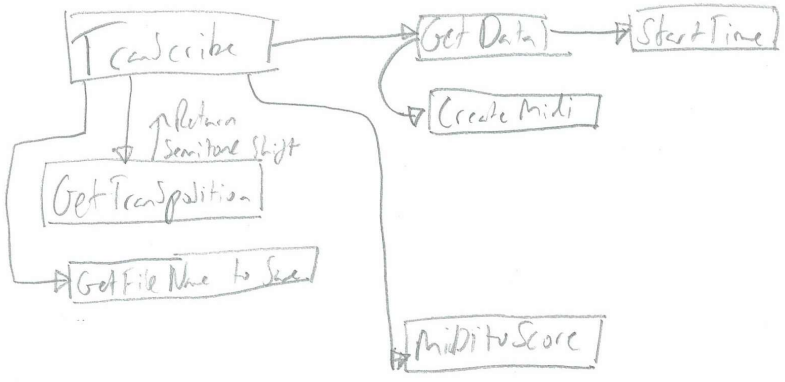


```
        debugger.reset()
        debugger.do_next(None)
        user_frame = sys._getframe().f_back
        debugger.interaction(user_frame, None)
    finally:
        pyqtRestoreInputHook()
```

This function is used to debug with pdb alongside PyQt applications without an endless output from PyQt.
Source https://stackoverflow.com/questions/1736015/debugging-a-pyqt4-app

save()

```
def save():
        try:
                import notateWAVpostRecording
        except Exception as e:
                print ("[E] {}".format(e))
        name = scribeUI.fileNameTextBox.text()

notateWAVpostRecording.postAnalyseAudio("recording.wav",name,calculateTransposition(scr
ibeUI,keys)[1])

        upload = sql.upload ##upload(midiFileName,scoreFileName,connection):
        user = scribeUI.userNameTextBox.text()
        date = datetime.datetime.today().strftime('%d/%m/%Y')
        if checkUniqueName(name) == True:
                cnxn = sql.start(NGROKPORT)
                midiurl , scoreurl, dbconnection =
upload(name+"_midiFile.mid",name+"_scoreFile.pdf")

                dropboxHash = generateDropboxHash(midiurl,name,dbconnection)
                uploadHash = generateUploadHash(scoreurl,name,dbconnection)
                try:
                        query = "INSERT INTO download
VALUES({},'{}','{}')".format(dropboxHash,midiurl.decode("utf-8"),scoreurl.decode("utf-
8"))
                        downloadDF = sql.query(query,cnxn)
                        query = "INSERT INTO info
VALUES('{}','{}','{}',{},{})".format(user,date,name,uploadHash,dropboxHash)
                        infoDF = sql.query(query,cnxn)
                        cnxn.commit()
                        raiseError("Files Successfully Uploaded","#77dd77","Upload
Status")
                        transcribeResetButton_clicked() ##reset

                except:
```



```
                    raiseError("Error Uploading Files")
        else:
                raiseError("Name Not Unique, please choose a different name")
```

The function save analyses the saved audio and updates the database with the new information. First the recording made from the user is put through postAnalyseAudio, which is a function to generate the MIDI and Score Files. Next we pull the information entered by the user to be used later. Then we check if the file name is unique, if it is we create a connection to the sql database and then proceed to upload the MIDI and Score files to the file server and get their urls, we do this using the upload function. After the files have been uploaded we generate hashes for the dropboxID and uploadID. If everything is successful we start by appending the database with the new information. First the 'download' table is appended because of the primary key dependency, with the dropboxID and the download urls. And then next the 'info' table with the user details and the uploadID and the dropboxID. If no errors occur we commit the changes to the database and show a success message using the raiseError function, then finally the user inputs are reset alongside the timer. If errors do occur the corresponding message is displayed.

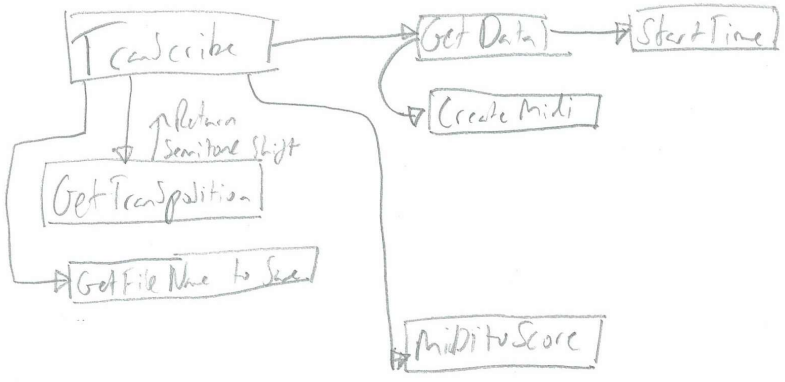
transcribeStopButton_clicked()

```
@pyqtSlot()
def transcribeStopButton_clicked():
    scribeUI.startButton.setEnabled(True)
    scribeUI.stopButton.setEnabled(False)
    scribeUI.resetButton.setEnabled(True)
    scribeUI.startKeyDial.setEnabled(True)
    scribeUI.endKeyDial.setEnabled(True)
    scribeUI.startKeyCombo.setEnabled(True)
    scribeUI.endKeyCombo.setEnabled(True)
    scribeUI.fileNameTextBox.setEnabled(True)
    scribeUI.userNameTextBox.setEnabled(True)
    scribeUI.timer.stop()
    ###save()
```

This pyqt slot function reenables all buttons for the user and stops the timer thread.

transcribeResetButton_clicked()

```
@pyqtSlot()
def transcribeResetButton_clicked():
    scribeUI.startButton.setEnabled(True)
```



```
    scribeUI.stopButton.setEnabled(False)
    scribeUI.resetButton.setEnabled(True)
    scribeUI.startKeyDial.setEnabled(True)
    scribeUI.endKeyDial.setEnabled(True)
    scribeUI.startKeyCombo.setEnabled(True)
    scribeUI.endKeyCombo.setEnabled(True)
    scribeUI.fileNameTextBox.setEnabled(True)
    scribeUI.userNameTextBox.setEnabled(True)
    scribeUI.startKeyDial.setValue(0)
    scribeUI.endKeyDial.setValue(0)
    scribeUI.fileNameTextBox.setText("")
    scribeUI.userNameTextBox.setText("")



    scribeUI.TimerReset()
```

This pyqt slot function sets all user inputs back to their default state and enables them again. It also resets the on screen timer to 00:00:00.

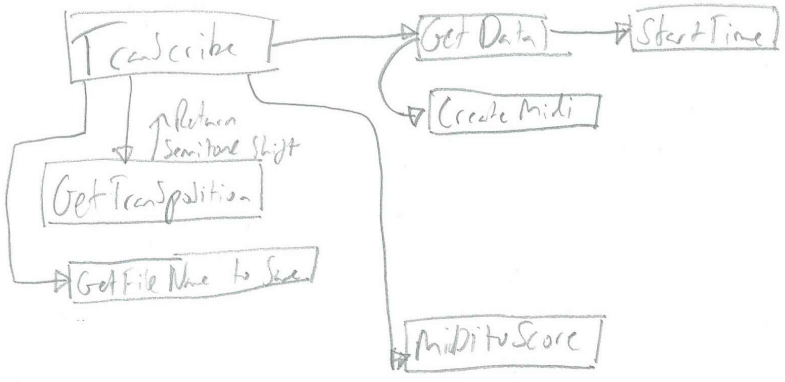
setupScribeSlots()

```
def setupScribeSlots():
    scribeUI.startButton.clicked.connect(transcribeStartButton_clicked)
    scribeUI.stopButton.clicked.connect(transcribeStopButton_clicked)
    scribeUI.resetButton.clicked.connect(transcribeResetButton_clicked)
    scribeUI.timer.timeout.connect(scribeUI.TimerTime)
```

This setupSlots connects the buttons for the transcribe window. When the start button is clicked the transcribeStartButton_clicked function is ran. When the stop button is clicked the transcribeStopButton_clicked function is ran. When the reset button is clicked the transcribeResetButton_clicked function is ran and finally when any time has passed the function TimerTime is called to start 'tick'.

calculateTransposition()

```
def calculateTransposition(ui,keys):
    idxStart = ui.startKeyCombo.currentIndex()
    idxStop = ui.endKeyCombo.currentIndex()
    if idxStart == 0 or idxStop == 0 or idxStart == -1 or idxStop == -1:
        print ("ERROR : Keys Not Defined")
        raiseError("Keys Not Defined")
        return False
    else:
        idxStart = idxStart-1 ##Change from indexing starting at 1 to at 0
```



```
        idxStop = idxStop-1
        if idxStart <0:
            idxStart+12
        if idxStart < 0 :
            idxStop+12
        start = keys[idxStart]
        stop = keys[idxStop]
        transposition = (idxStop-idxStart)
        print (str(transposition),"Semitones")
        return True,transposition
```

This function calculates the needed transposition specified by the user. First it grabs the indexes selected by the user and reduces them by one to change from indexing from 1 to indexing at 0. If no keys are selected by the user an error is raised by raiseError and returns False to stop the program from continuing. Otherwise the transposition is cycled so it falls between 0 and 12 so the indexing works easily, the function returns True because a transposition is defined so the program continues and also returns the transposition value so other functions can use it.

sqlButton_clicked()

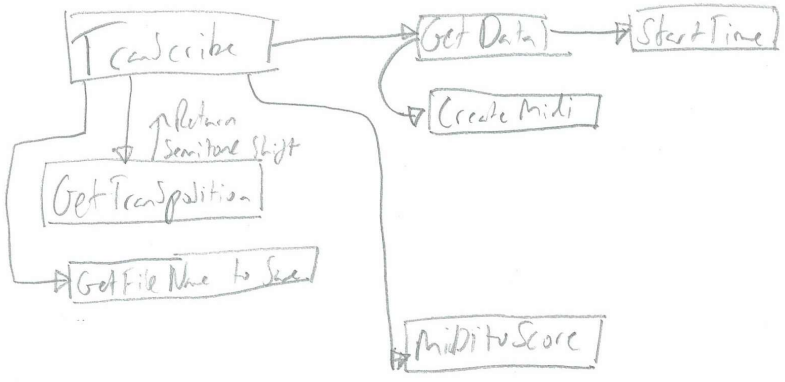
```
@pyqtSlot()
def sqlButton_clicked():
        sqlWindow.show()
```

This pyqt slot function shows the SQL Search window when the sql search option button is clicked in the main window.

setupSlotsTrans()

```
def setupSlotsTrans():
    transUI.startKeyCombo.currentTextChanged.connect(getStartKeyData)
    transUI.endKeyCombo.currentTextChanged.connect(getEndKeyData)
    transUI.stopButton.clicked.connect(stopButton_clicked)
    transUI.sensitivitySlider.valueChanged.connect(getSensitivityData)
    transUI.startButton.clicked.connect(startButton_clicked)
```

This setupSlots function is for the transpose window, it connects the user inputs to their functions. When the startKeyCombo's value is changed the getStartKeyData function is ran. When the endKeyCombo value is changed the getEndKeyData function is ran. When the stop button is clicked the stopButton_clicked function is ran. When the sensitivitySlider value is changed the getSensitivityData function is ran and when the start button is clicked the startButton_clicked function is ran.



setupSlotsMain()

```
def setupSlotsMain():
    mainUI.transposeButton.clicked.connect(transposeButton_clicked)
    mainUI.transcribeButton.clicked.connect(transcribeButton_clicked)
    mainUI.sqlSearchButton.clicked.connect(sqlButton_clicked)
    mainUI.settingsButton.clicked.connect(settingsButton_clicked)
```

This setupSlots function sets up the buttons for the main startup window. When the transpose button is clicked the transposeButton_clicked function is ran. When the transcribe button is clicked the transcribeButton_clicked function is ran. When the sql Search button is clicked the sqlButton_clicked function is ran and when the settings button is clicked the settingsButton_clicked function is ran.

applyAudioSettings()

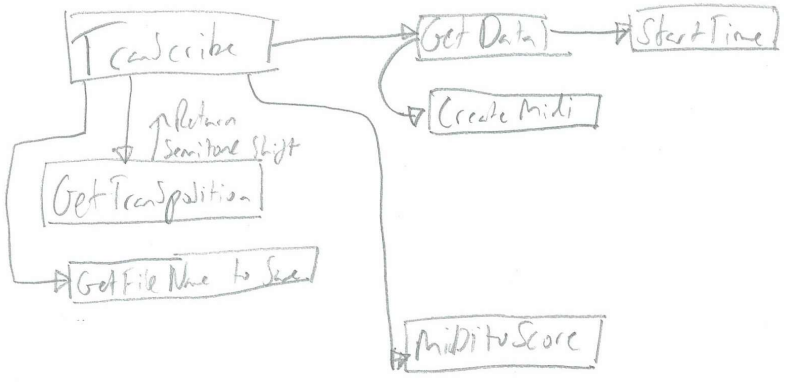
```
@pyqtSlot()
def applyAudioSettings():
        raiseError("Settings Applied!",'#77dd77',"Success!")
        audioDeviceSettingIs = settingsUI.audioDeviceComboBox.currentText()
        audioDeviceSettingIndex = settingsUI.audioDeviceComboBox.currentIndex()
        print ("audio device selected = "+str(audioDeviceSettingIs))
        with open("audioSettingsFile.txt","w") as f:
                f.write("FORMAT:DEVICENAME,INDEX\n")
                f.write("{},{}".format(audioDeviceSettingIs,
                                       audioDeviceSettingIndex))
                f.close()
        settingsWindow.hide()
```

This pyqt slot function saves the settings decided by the user. Initially a success message is shown to the user through raiseError then selected device name from the dropdown box and the device index are saved into the file audioSettingsFile.txt in the format DEVICENAME,INDEX. As this is the only editable setting that needs to be saved, once the apply button is pressed the settings window closes.

setupSettingsSlots()

```
def setupSettingsSlots():
        settingsUI.refreshAudioDevicesBUtton.clicked.connect(refreshAudioDevices)
        settingsUI.applyAudioSettingsButton.clicked.connect(applyAudioSettings)
```

This setupSlots function connects the settings window buttons. When the refresh button is pressed the function refreshAudioDevices is ran and when the apply button is pressed the applyAudioSettings function is run.



refreshAudioDevices()

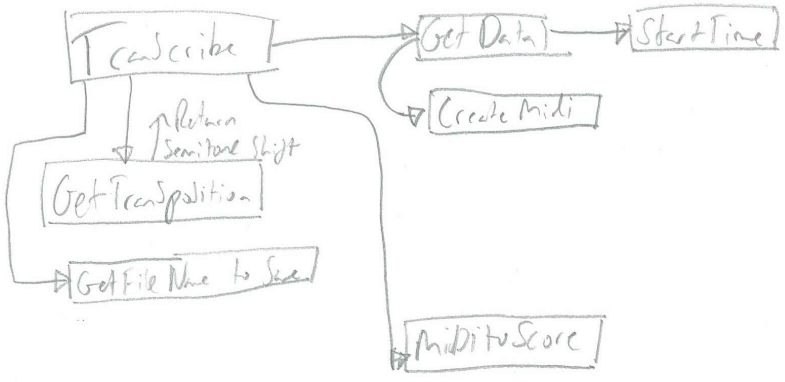
```
@pyqtSlot()
def refreshAudioDevices():
        import findAudioDevice
        audioDeviceList= findAudioDevice.locate()
        if len(audioDeviceList)==0:
            settingsUI.audioDeviceComboBox.clear()
            settingsUI.audioDeviceComboBox.addItem("No Devices Found")

        else:
            for device in audioDeviceList:
                if device[0] in [settingsUI.audioDeviceComboBox.itemText(i) for i in
range(settingsUI.audioDeviceComboBox.count())]:
                    pass
                else:
                    settingsUI.audioDeviceComboBox.addItem(device[0])
        audioDeviceComboBoxContents = [settingsUI.audioDeviceComboBox.itemText(i) for i
in range(settingsUI.audioDeviceComboBox.count())]
        for deviceAdded in audioDeviceComboBoxContents :
            if deviceAdded not in [a[0] for a in audioDeviceList]:
                #device disconnected

settingsUI.audioDeviceComboBox.removeItem(audioDeviceComboBoxContents.index(deviceAdded
))
```

This pyqt slot function gets the list of available audio devices and puts them into the settings window dropdown box. First it imports the module fundAudioDevices and finds all the devices, it receives a list of information about the devices. If there are no devices found the only item in the dropdown box is set to convey this. Otherwise the device is added to the dropdown box. If the dropdown box already contains this data a duplicate is not added. If there is a device in the dropdown box which isn't in the most recent list (device has been disconnected) it is removed from the dropdown box.

settingsButton_clicked()

```
@pyqtSlot()
def settingsButton_clicked():
        refreshAudioDevices()
        settingsWindow.show()
```

This pyqt slot function is run when the settings button is clicked, first it runs the refreshAudioDevices function then it shows the settings window to the user.



closeErrorWindow_clicked()

```
@pyqtSlot()
def closeErrorWindow_clicked():
    errorWindow.hide()
```

This pyqt slot function is run when the ignore button is clicked in the error window. It hides the error window from the user.

setupSlotsError()

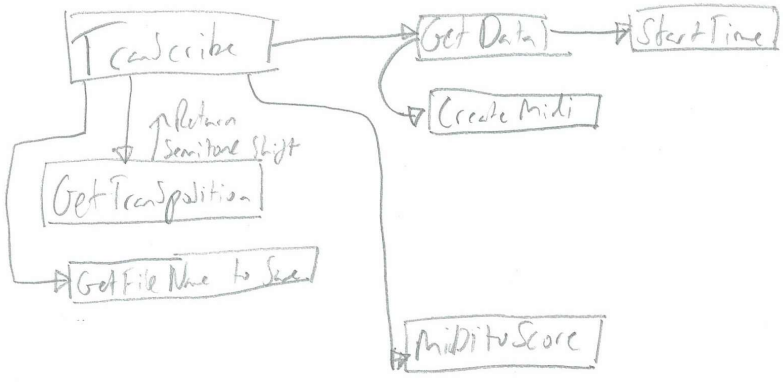
```
def setupSlotsError():
    errorUI.ignoreButton.clicked.connect(closeErrorWindow_clicked)
```

This setupSlots function sets up the button on the error window, when the button is clicked the function closeErrorWindow_clicked is ran.

raiseError()

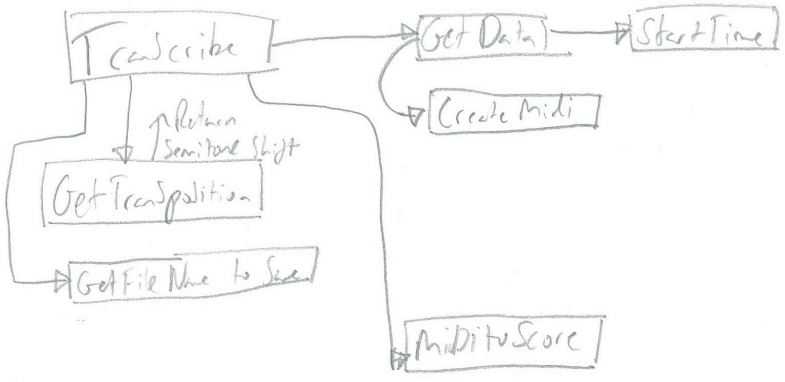
```
def raiseError(message,colouring="",text=""):
    if colouring=="":
            errorWindow.setStyleSheet("""QMainWindow{{
background-color: {}
}};""".format("#FF6961"))
    else:
            errorWindow.setStyleSheet("""QMainWindow{{
background-color: {}
}};""".format(colouring))
    if text=="":
            errorUI.errorLabel.setText("Error!")
            errorWindow.setWindowTitle("Error Message")
    else:
            errorUI.errorLabel.setText(text)
            errorWindow.setWindowTitle("{}".format(text))
    errorUI.label.setText(str(message))
    errorWindow.show()
```

This function is used to show a window with a variable message in it. The first if statement checks if the keyword argument is being used, if it is the background colour of the window is set to the hex code of the variable colouring. Likewise there is an optional text variable to set which is the title of the window, and message is the main content for the error. When called the window is shown to the user.



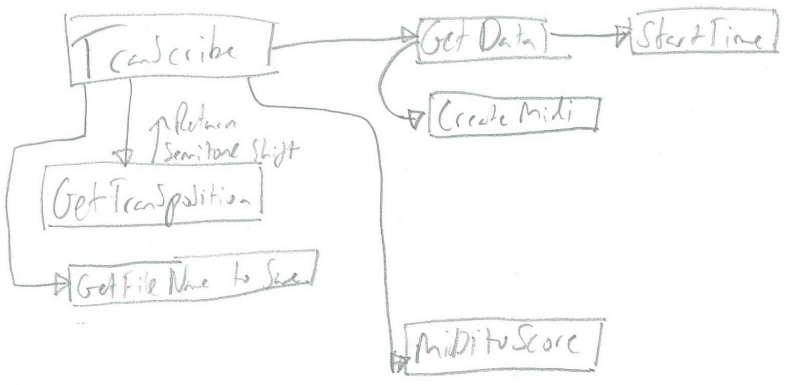
liveNoteThread()

```
class liveNoteThread(threading.Thread):
        def __init__(self,name='liveNoteThread',sensitivity=0.6):
                self._stopevent = threading.Event()
                self.identifiedNoteQueue = queue.Queue()
                self.sensitivity = sensitivity
                self.p = list()
                self.stream = list()
                freqFileLocation = locateResources.findFile("freq.txt")
                f = open(freqFileLocation,"r")
                self.FREQ = list()
                for line in f:
                        if line =="":
                                pass
                        else:

self.FREQ.append(line.replace("\n","").replace("\xa0",""))
                f.close()
                notesFileLocation = locateResources.findFile("note.txt")
                f = open(notesFileLocation,"r")
                self.NOTE = list()
                for line in f:
                        if line =="":
                                pass
                        else:

self.NOTE.append(line.replace("\n","").replace("\xa0",""))
                f.close()
                self.npFREQ = np.asarray([float(p) for p in self.FREQ])
                threading.Thread.__init__(self, name=name)

        def run(self):
                while not self._stopevent.isSet():
                    self.noteIdentification()

        def noteIdentification(self):
                self.p = pyaudio.PyAudio()
                # Open stream.
                DEVICEIDX =
int(open(locateResources.findFile("audioSettingsFile.txt"),"r").readlines()[1].split(",
")[1])
                print ("Device IDX {}".format(DEVICEIDX))
                try:
                        self.stream = self.p.open(format=pyaudio.paFloat32,
                    channels=1, rate=44100, input=True,
                    frames_per_buffer=1024,input_device_index=DEVICEIDX)
                        pDetection = aubio.pitch("default", 2048,
                    2048//2, 44100)
```



```
                    pDetection.set_unit("Hz")
                    pDetection.set_silence(-100)
                    sumPitch=list()
                    print ("stream initialised")
            except Exception as e:
                    print ("[E] {}".format(e))
            while True:
                data = self.stream.read(1024)
                samples = np.fromstring(data,
                    dtype=aubio.float_type)
                pitch = pDetection(samples)[0]
                sens = 5.0*(int(self.sensitivity)/6)
                if len(sumPitch)<sens:
                    sumPitch.append(round(pitch))
                else:
                    accu=int()
                    for item in sumPitch:
                        accu +=item
                    frequenc = round(np.mean(accu)/20)
                    self.closest(frequenc)
            ##              print frequenc
                    sumPitch=list()

    def closest(self,num):
            try:
                    idx = (np.abs(self.npFREQ-num)).argmin()
                    p = self.NOTE[idx]
                    self.identifiedNoteQueue.put(p)
            except Exception as e:
                    print ("[E] {}".format(e))
                    print ("[D] variable num == {}".format(num))
                    print ("[D] variable idx == {}".format(idx))
                    print ("[D] variable p == {}".format(p))
                    print ("[D] variable self.npFREQ == {}".format(self.npFREQ))

    def join(self):
            self.stream.stop_stream()
            self.stream.close()
            self.p.terminate()
            self._stopevent.set()
            threading.Thread.join(self)
```

This thread is for identifying the notes coming in from the live audio feed in the transpose window. On creation variables are declared with the most important being the identifiedNoteQueue and the audio streams. After the variables are declared the frequency file is loaded using the locateResources module and subsequently the FREQ variable is created which is every note in the file turned into a list, with the escape characters removed. The same happens for the note file and the NOTE list. Finally the

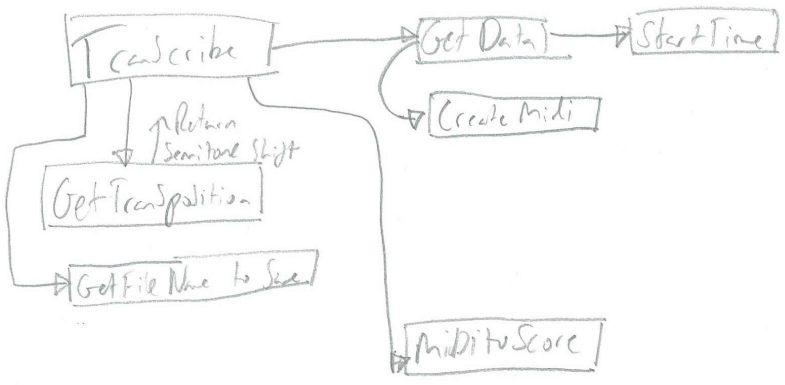


thread is initialised so it contains the attributes of a thread. The run function calls the noteIdentification function until the stopEvent is signaled. noteIdentification is a function that takes in an audio stream and identifies the note being played, first it creates the audio stream using pyaudio and uses the audio device specified in audioSettingsFile.txt . Then it will take 1024 frames per buffer and apply the pitch analysis. Also defined here is the limits of the analysis such as the sample rate being 44100Hz and the units. For every buffer taken a numpy array is created so we can use aubio's pitch detection (Fourier Transform) to decipher the frames. The sensitivity variable is defined as 5/6ths of the sensitivity value which ranges from 10 to 100, this value is used to determine how many samples are taken before an average is taken. With all the identified frequencies we will take an average to determine the most probable note that was played across the duration, with the average frequency we then pass it to the closest function which finds the closest musical note to this value.
The closest function uses numpy to find the value which has the smallest difference to our pitch, it then finds the index of this frequency in the FREQ list. As both FREQ and NOTE are indexed in the same way, the note at the index of the frequency is the closest note. Finally the closest function puts the note into the identifiedNoteQueue so it can be accessed by the listening thread whileGo.
The join function is used to terminate the threads without causing any errors, it first stops the audio stream and then the pyaudio object and then finally the noteIdentificationThread by signalling the stop event.

whileGo()

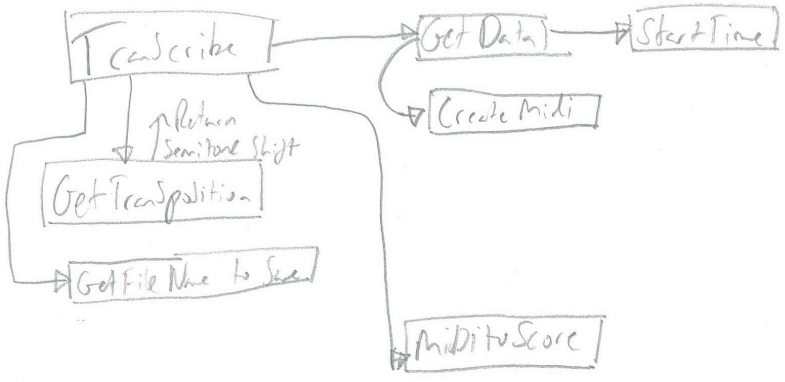
```
def whileGo():
        global liveAudioIdentification
        liveAudioIdentification = liveNoteThread(sensitivity=returnSensitivityData())
        liveAudioIdentification.start()
        while True:
                note = liveAudioIdentification.identifiedNoteQueue.get()
                if "/" in note:
                        note = note.split("/")[0].upper()
                        note =  (note[:-1])
                else:
                        note = (note[0].upper())
                updateNoteLabel(note)
```

The whileGo function checks the identifiedNoteQueue for new values and when it finds one it updates the Ui using updateNoteLabel. First, it starts the audioIdentification thread and then starts looking at the queue until values are added, when it gets a value it converts it to the form /([A-Z]){1}[#]?)/ so it can call the update function without any errors.

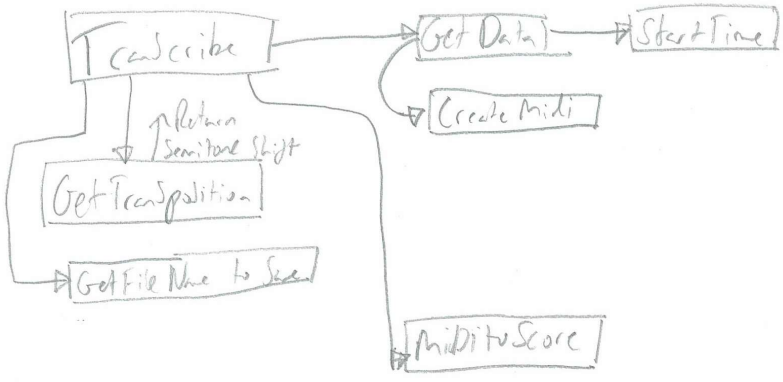


class AppContext()

```
class AppContext(ApplicationContext):
        def run(self):
                global NGROKPORT
                NGROKPORT = str(input("ENTER NGROK PORT :"))
                global app
                app = QtWidgets.QApplication(sys.argv)
                app.setWindowIcon(QIcon(LOGO_PATH))

                global MainWindow
                MainWindow = QtWidgets.QMainWindow()
                ##StyleSheet
                with open(locateResources.findFile("startUpWindow.stylesheet"),"r") as
ss:
                    MainWindow.setStyleSheet(ss.read())
                global mainUI
                mainUI = Ui_MainWindow()
                mainUI.setupUi(MainWindow)


                setupSlotsMain()

                global transposeWindow
                transposeWindow = QtWidgets.QDialog()
                transposeWindow.setWindowIcon(QIcon(LOGO_PATH))
                transposeWindow.setWindowTitle("Transpose") ##################
                global transUI
                transUI = Ui_Dialog()
                transUI.setupUi(transposeWindow)
                transUI.currentNoteImage.setPixmap(QtGui.QPixmap(LOGO_PATH))
                setupSlotsTrans()
                ##StyleSheet
                with open(locateResources.findFile("transWindow.stylesheet"),"r") as
ss:
                    transposeWindow.setStyleSheet(ss.read())

                global transcribeWindow
                transcribeWindow = QtWidgets.QMainWindow()
                transcribeWindow.setWindowIcon(QIcon(LOGO_PATH))
                transcribeWindow.setWindowTitle("Transcribe") ##################
                global scribeUI
                scribeUI = Ui_scribeWindow()
                scribeUI.setupUi(transcribeWindow)
                setupScribeSlots()
                ##StyleSheet
                with open(locateResources.findFile("transcribeWindow.stylesheet"),"r")
as ss:
                    transcribeWindow.setStyleSheet(ss.read())
```



```
                global settingsWindow
                settingsWindow=QtWidgets.QMainWindow()
                global settingsUI
                settingsUI = Ui_settingsWindow()
                settingsUI.setupUi(settingsWindow)
                setupSettingsSlots()
                ##StyleSheet
                with open(locateResources.findFile("settingsWindow.stylesheet"),"r") as
ss:
                    settingsWindow.setStyleSheet(ss.read())

                global errorWindow
                global errorUI
                errorWindow = QtWidgets.QMainWindow()
                errorUI = Ui_Error()
                errorUI.setupUi(errorWindow)
                setupSlotsError()
                ##StyleSheet
                with open(locateResources.findFile("errorWindow.stylesheet"),"r") as
ss:
                    errorWindow.setStyleSheet(ss.read())

                global sqlWindow
                global sqlWindowUI
                sqlWindow = QtWidgets.QMainWindow()
                sqlWindowUI = Ui_sqlWindow()
                sqlWindowUI.setupUi(sqlWindow)
                setupSqlSlots()
                ##StyleSheet
                with open(locateResources.findFile("sqlWindow.stylesheet"),"r") as ss:
                    sqlWindow.setStyleSheet(ss.read())
                ##    sqlWindowUI.refreshTable(df) ##populate with data from sql

                global downloadWindow
                global downloadWindowUI
                downloadWindow = QtWidgets.QDialog()
                downloadWindowUI = Ui_downloadWindow()
                downloadWindowUI.setupUi(downloadWindow)


                MainWindow.show()
                transposeWindow.hide()
                transcribeWindow.hide()
                errorWindow.hide()
                sqlWindow.hide()
                downloadWindow.hide()
                settingsWindow.hide()

                global transAudioTR
```



```
                transAudioTR = transAudioThread()
                mainGui = guiThread()
                return self.app.exec_()
```

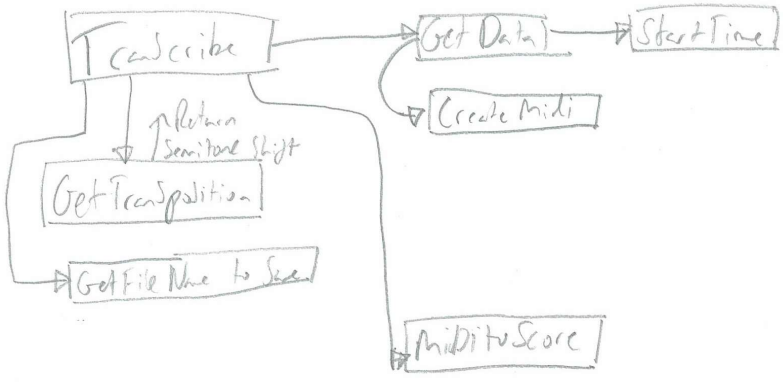
This class is used to run the PyQt5 application without it hanging or crashing. The run function is the code which will be executed on start. First it declares a global variable NGROKPORT and sets it to the value entered by the user. NGROKPORT is the port number for the tcp tunnel to get to the database. Next the parent app is created and its child windows follow. The MainWindow is first created as an object then its stylesheet is assigned to it using the locateResources function. Next the contents of the window are created and setup with the main window class. The final part for the window is to setup the slots for it which is done by calling the function setupSlotsMain. This is done for every window: transposeWindow, transcribeWindow, settingsWindow, errowWindow, sqlWindow and the download window. Next the only window the user should see is the main window so all the others are hidden. Finally the transcribe and the gui threads are created and the executing window is returned.

```
if __name__ == "__main__":
        global keys
        keys = ["C","C#","D","D#","E","F","F#","G","G#","A","A#","B"]
        global LOGO_PATH
        LOGO_PATH = locateResources.findFile("NSLogo.png")

        NGROKPORT = "0"###ngrokport ie 0.tcp.ngrok.io:PORT
        appctxt = AppContext()
        exit_code = appctxt.run()
        sys.exit(exit_code)
```

Finally more global variables are created, these being keys, a default value for NGROKPORT and the path for the logo file by locateResources. Then the AppContext object is created and ran.

All the Ui_ Classes contain a function called retranslateUi, this is for any text objects within the Window, it allows it to be easily translated into different languages.

notateWAVpostRecording.py



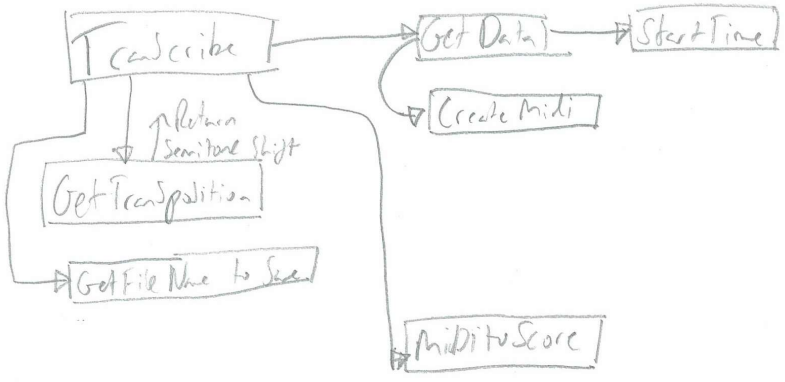
```
from pyknon.genmidi import Midi
from pyknon.music import NoteSeq
from pyknon.music import Note,Rest
import pyknon
import numpy as np
import pdb
import os
```

Imports the required modules for the program

***class** dataStore()*

```
class dataStore(): ### Stores all data needed for midi file
    def __init__(self):
        self.startTime=float()
        self.tempo = int()
        self.MidiData = list()
        self.duration = list()
        self.volume = list()
        self.NoteOnSet = list()
    def view(self):
        print(self.startTime)
        print(self.tempo)
        print(self.MidiData)
        print(self.duration)
        print(self.volume)
        print(self.NoteOnSet)
    def add(self,note,duration,onset):
        try:
            if len(note)>2:
                note = note[:3]
        except TypeError:
            pass
        self.MidiData.append(note)
        self.duration.append(duration)
        self.volume.append(100)
        self.NoteOnSet.append(onset)
    def setTempo(self,bpm):
        self.tempo = bpm
    def export(self):
        self.MidiData = self.MidiData[1:]
        self.duration = self.duration[1:]
        self.volume = self.volume[1:]
        self.NoteOnSet = self.NoteOnSet[1:]
```

This class creates a new data structure for storing the information needed to construct the MIDI file. The init function creates the class variables for later use, and the view function displays the contents of each variable. The add function takes a note name, note length and the note start time. It then makes sure the note is in the format /[A-Z]{1}['#']?[0-9]{1}/



(note with accidental followed by octave), after which it will append it to the class variables, if the note is a '-1' then it is denoting a rest instead of a note. The function setTempo will store the given bpm so it can be added to the MIDI file. Finally the export function removes the leading item in every class list, this is so the bars of silence until the played begins aren't notated.

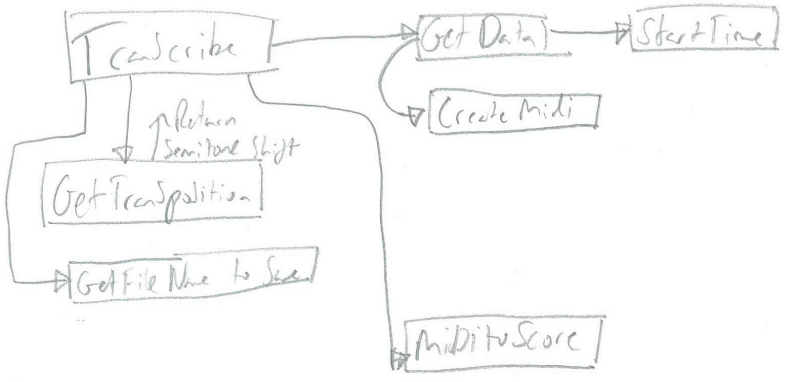
toValueOctave()

```
def toValueOctave(txt):
    print(txt)
    txt = txt.upper()
    octave = int(txt[-1])+1
    note = txt[:-1]
    if note == "Eb":
        note = "D#"
    elif note == "Ab":
        note = "G#"
    elif note =="Bb":
        note = "A#"
    elif note == "A":
        note = "A"
    value = ["A#","B","C","C#","D","D#","E","E#","F","F#","G","G#","A"].index(note)
    return int(value),int(octave)
```

This function takes a note in the form of a string, it then converts it to uppercase and strips the octave from it (the last character) and increments it as the pyknon middle c is C5. Then note is then converted to remove any flat symbols from it and replaces it with its relative sharp. Finally it returns the index of the Note and the Octave value.

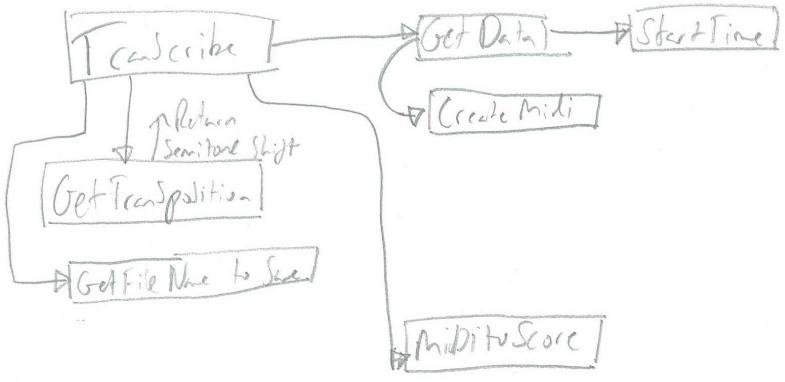
save()

```
def save(AudioData,name,transpositionValue):
    import aubio
    import os

    AudioData.export()
    INSTRUMENT = 0
    midiFile = Midi(1,AudioData.tempo,INSTRUMENT)

    noteSeqList=list()
    for i in range(len(AudioData.MidiData)):
        if AudioData.MidiData[i] == -1:
            rest = Rest(float(AudioData.duration[i]))
            noteSeqList.append(rest)
        else:
```



```
            note =
Note(toValueOctave(transposeNote(AudioData.MidiData[i],transpositionValue))[0],toValueO
ctave(transposeNote(AudioData.MidiData[i],transpositionValue))[1],dur=float(AudioData.d
uration[i])) ##edited most recently
            noteSeqList.append(note)

    midiFile.seq_notes(noteSeqList)
    ############USAGE##############
        ##NoteSeq([Note(note_num, dur=note_dur)])
    ##midi.seq_notes(notes1, track=0) ### Puts notes into file

    midiFile.write("{}_midiFile.mid".format(name)) ### Writes To file
    print("DONE")
    import locateResources as tempImport
    import time
    convertScriptLocation = tempImport.findFile("convert.bat")
    print ("[D] Entering Debug Mode")
    print ("[D] Before Exiting exectue cmd : PyQt5.QtCore.pyqtRestoreInputHook()")
    print ("[D] Convert.bat is held at {}".format(convertScriptLocation))
    print ('[D] OS Command to run is ["{0}" {1}_midiFile.mid
{1}_scoreFile]'.format(convertScriptLocation,name))
    os.system('"{0}" {1}_midiFile.mid
{1}_scoreFile'.format(convertScriptLocation,name)) ### Calls batch script to convert,
param1 is the input and param2 is the output file in the curdir
    print("[D] Converted")
    outputfiles = (os.getcwd(),(str("{}_midiFile.mid".format(name)),str(name)))
    print(outputfiles)
    print ("[D] Waiting until score file exists")
    while not (os.path.exists("{}_scoreFile.pdf".format(name))):
        print ("[D] PDF FILE NOT FOUND")
        time.sleep(1)
    print ("[D] PDF FILE FOUND")
```

The save function takes the information stored in the dataStore structure and creates a MIDI file with it. First it makes sure the required modules are available by reimporting them, then it runs the audioData's export function to remove the leading silence. A MIDI file is then initialised with one track, the tempo and the default instrument which is the keyboard. To construct the information into a MIDI file all the entries in the AudioData structure are cycled through in chronological order, we check if the note value is '-1' indicating a rest; if it is we will add a rest using the Rest function from pyknon at have it 'play' for the corresponding duration. If the note isn't a rest then we will need to add extra information. For a note we will use Pyknon again but the Note function, we will provide it with the note in the form [integer value, integer octave value] by using the toValueOctave Function, before this we will apply the transposition using the transposeNote function to change it into the desired key. Alongside the note values we will also pass in the duration it was held for. We then add this note or rest to the sequence of events. By doing this in chronological order we don't need to worry about onset times as we create a virtual



sequence of identical events. Finally we will add the sequence of events (noteSeqList) to the MIDI file we created and save it under the name (name)_mididFile.mid .

The next stage is to convert this midi file into a score file using the convert.bat script. We will first get the exact path of the conversion script using locateRescourses and then make a command line call to use convert.bat with the parameters (name)_midiFile.mid and (name). As this process will take a varying amount of time depending on the length of the MIDI file we will wait until the file name_scoreFile.pdf is created and then continue when we find it.

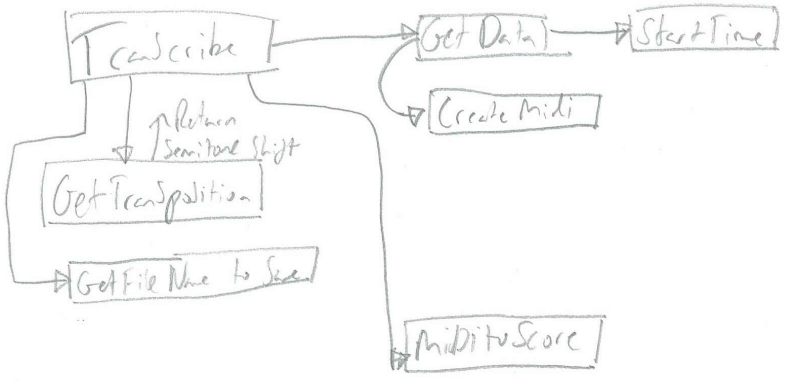
debugTrace()

```
def debugTrace():
    from PyQt5.QtCore import pyqtRemoveInputHook, pyqtRestoreInputHook
    import pdb
    import sys
    pyqtRemoveInputHook()
    try:
        debugger = pdb.Pdb()
        debugger.reset()
        debugger.do_next(None)
        user_frame = sys._getframe().f_back
        debugger.interaction(user_frame, None)
    finally:
        pyqtRestoreInputHook()
```

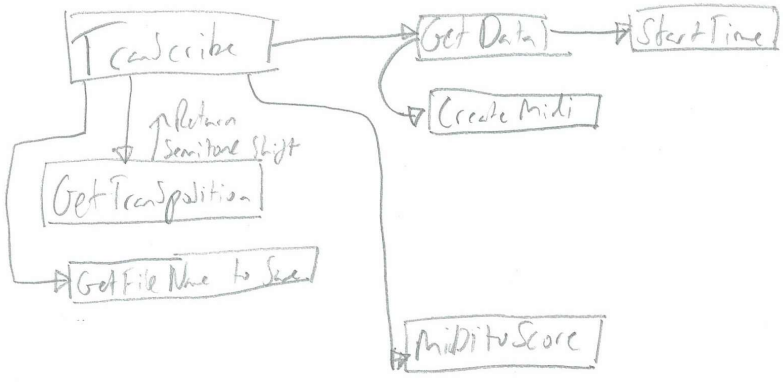
This function is used to debug with pdb alongside PyQt applications without an endless output from PyQt.
Source https://stackoverflow.com/questions/1736015/debugging-a-pyqt4-app

analyse()

```
def analyse(fileName,AudioData):
    from aubio import source,pitch,miditofreq,freq2note
    downsample = 1
    samplerate = 44100 // downsample

    win_s = 4096 // downsample # fft size
    hop_s = 512  // downsample # hop size

    s = source(fileName, samplerate, hop_s)
```

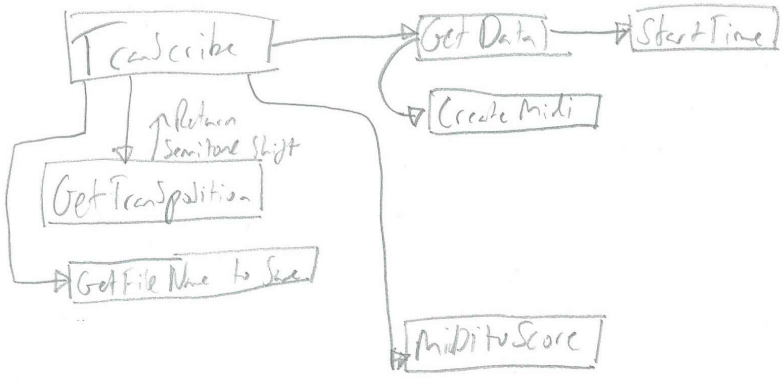


```
    samplerate = s.samplerate

    tolerance = 0.9

    pitch_o = pitch("yin", win_s, hop_s, samplerate)
    pitch_o.set_unit("midi")
    pitch_o.set_tolerance(tolerance)

    pitches = []
    confidences = []

    # total number of frames read
    total_frames = 0
    data = list()
    while True:
        samples, read = s()
        pitch = pitch_o(samples)[0]
        pitch = int(round(pitch))
        confidence = pitch_o.get_confidence()
        if confidence < 0.8: pitch = 0.
        pitch = miditofreq(pitch)
        #print("%f %f %f" % (total_frames / float(samplerate), pitch, confidence))
        time = (total_frames/float(samplerate))
        if pitch == miditofreq(0):
            subdata = [time,-1]
        else:
            subdata = [time,pitch]
        data.append(subdata)
        pitches += [pitch]
        confidences += [confidence]
        total_frames += read
        if read < hop_s: break


    tempdata = list()

    timeOffsets = 1.0/64
    noteOffsets = 0
    startTime = 0
    endTime = 0
    condensedData = list() #[note,startTime,endTime,Duration]
    for entry in data:
        ##print entry,
        idx = data.index(entry)
        try:
                if data[idx+1][1]-noteOffsets <= data[idx][1]
<=data[idx+1][1]+noteOffsets:
                #print("SAME NOTE")
                    endTime = data[idx+1][0]
```



```
                else:
                    ##print("SAME NOTE DOESNT CONTINUE THEREFORE ENDPOINT")
                    ##print("NOTE: {} STARTTIME: {} ENDTIME:
{}".format(entry[1],startTime,endTime))
                    duration = round(endTime-startTime,3)
                    if endTime == 0:
                        continue
                    elif abs(duration)<timeOffsets:
                        continue
                    else:
                        condensedData.append([entry[1],startTime,endTime,duration])
                    startTime = data[idx+1][0]
                    endTime = 0
        except IndexError:
            ##print("NOTE: {} STARTTIME: {} ENDTIME:
{}".format(entry[1],startTime,endTime))
            if endTime == 0:
                continue
            try:
                tempVar = abs(duration)
                if tempVar<timeOffsets:
                        continue
                else:
                    condensedData.append([entry[1],startTime,endTime,endTime-
startTime])
            except Exception as e:
                        print(e)

    for entry in condensedData:
        if entry[0] == -1:
            AudioData.add(-1,entry[3],entry[1])
        elif entry[3]==0:
            continue
        else:
            ##print(freq2note(entry[0]),entry[3])
            AudioData.add(freq2note(entry[0]),entry[3],entry[1])

    return AudioData
```

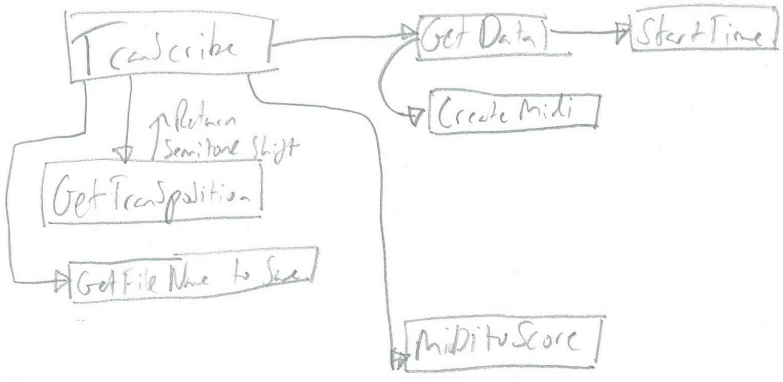
The analyse function will create a list of notes and their on and off timings, we do this by first reimporting the module requirements to make sure they are there, next we define some parameters for the analysis such as the downsample (1 as to not change the sample rate) , the sample rate (44100 as standard), the win_s (how large of a window to analyse at a time) and the hop_s rate (How big each frame should be). Next we need to open the waveform file we created with these settings so aubio can understand it. Next we will assign the sample rate we defined to the source file for aubio.
For the pitch detection as described earlier we will be using the YIN algorithm with a confidence rate of 0.9 (only select notes we are over 90% confident in). After all the



settings have been defined we will cycle through hops of our waveform file determining the pitch for each. For each hop we will extract the pitch from the YIN algorithm results and round it to the nearest integer value, then we will check it against another confidence level of 80%, if it falls below we will remove the pitch value and assign it the pitch of 0 (a rest) and append the data of [time, -1] (in the form onset, note) if there is a pitch we will append the same but with the note value. When we reach the end of the file the loop will stop.
Next we will condense the data so we have the note, its onset time, offset time and duration. The timeOffsets variable is the region where we decide that the note has been continued to be played without a rest. We will cycle through the data and check if the next value is the same note, if it is we will change the end time to the time when the following note ends, by doing this we can condense the data into single rest or note occurrences. We will also round the duration to 3 decimal places to avoid horrific notation. Finally we will add this information to the AudioData structure, we do this by cycling through the condensed data and using the add function to either add a rest or a note alongside its information, we also remove any accidental values of duration 0.

getBPM()

```
def getBPM(path):
    from aubio import source,tempo
    from numpy import median, diff
    downsample = 1
    samplerate = 44100 // downsample

    win_s = 4096 // downsample # fft size
    hop_s = 512  // downsample # hop size


    s = source(path, samplerate, hop_s)
    samplerate = s.samplerate
    o = tempo("specdiff", win_s, hop_s, samplerate)
    # List of beats, in samples
    beats = []
    # Total number of frames read
    total_frames = 0

    while True:
        samples, read = s()
        is_beat = o(samples)
        if is_beat:
            this_beat = o.get_last_s()
            beats.append(this_beat)
            #if o.get_confidence() > .2 and len(beats) > 2.:
            #    break
        total_frames += read
        if read < hop_s:
```

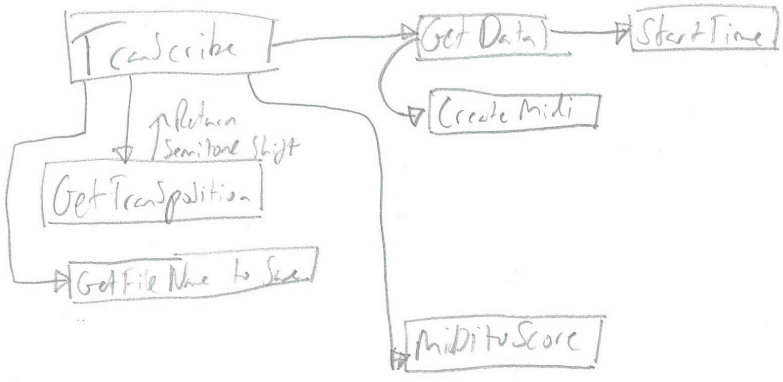


```
            break

    # Convert to periods and to bpm
    if len(beats) > 1:
        if len(beats) < 4:
            print("few beats found in {:s}".format(path))
        bpms = 60./diff(beats)
        b = median(bpms)
    else:
        b = 0
        print("not enough beats found in {:s}".format(path))
    return b
```

The getBPM function attempts to extract the tempo from the audio file. This code is taken from demo files from aubio. Source - https://github.com/aubio/aubio/blob/master/python/demos/demo_bpm_extract.py . First it defines the variables and opens the file for aubio like in the analyse function. It then uses spectral analysis to decide where the downbeats are in the file, it then takes the median difference between these beats and divides 60 by it. This calculates the BPM.

transposeNote()

```
def transposeNote(note,transpositionValue):
        keys = ["C","C#","D","D#","E","F","F#","G","G#","A","A#","B"]
        keys = [p+str(i) for i in range(0,9) for p in keys]
        startIdx = keys.index(note)
        if startIdx == None:
                raiseError("[e] starIdx is None")
        transpositionValue=transpositionValue+startIdx
        try:
                newNote = keys[transpositionValue]
        except:
                if transpositionValue>len(keys):
                        transpositionValue=transpositionValue-len(keys)
                elif transpositionValue*-1>0:
                        transpositionValue = len(keys)+transpositionValue

                try:
                        newNote = keys[transpositionValue]
                except Exception as e:
                        print ("THAT G#")
                        print (e)
                        print (len(keys))
                        print
(note,calculateTransposition(transUI,keys)[1],transpositionValue)
                        if transpositionValue == 12:
                                transpositionValue == 0
```



```
                print ("MANUALLY CHANGEG TRANSPO VALUE")


        try:
                return newNote
        except UnboundLocalError:
                return newNote
```

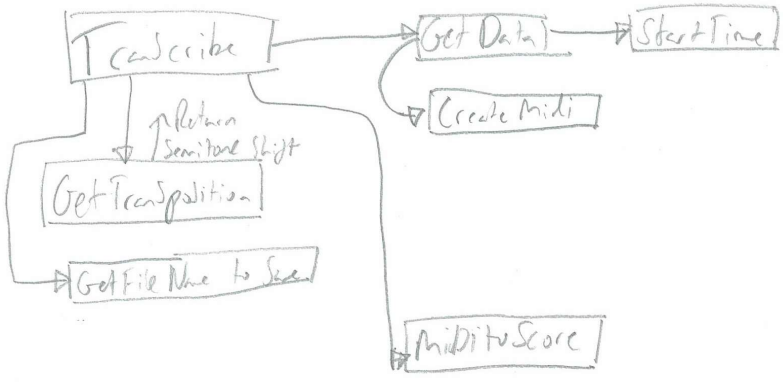
This transposeNote function is different to the one used in main.py as we require the octaves this time. We first create a new list of keys which contain the octaves 0 through 8 in the musical order. We then get the starting index of the note and shift it up or down by the transpositionValue. If the note is at the end or the start of the list we can run into indexing errors which is when an error would be raised as we would transpose it to a non-existent note.

postAnalyseAudio()

```
def postAnalyseAudio(wav_file,name,transpositionValue):
    AudioData=dataStore()
    AudioData = analyse(wav_file,AudioData)
    AudioData.setTempo(round(getBPM(wav_file)))
    save(AudioData,name,transpositionValue)
```

This function is what will be called by other programs, it runs the audio analysis of the waveform file then creates and converts the midi file into a score file. It first creates the AudioData structure then it runs the analyse function to apply the YIN algorithm, then it sets the tempo after identifying it using spectral analysis in the getBPM function. Finally this data is saved using the save function.

locateResources.py



```
import os
import difflib
cwd = os.getcwd()
possibleFiles,possiblePaths = list(),list()
```

Imports the necessary modules and sets up the variables needed later.

findMostSimilarFile()

```
def findMostSimilarFile(fileToFind,possible):
    similarFileFound = difflib.get_close_matches(fileToFind,possible,1,cutoff=0.3)[0]
    return possible.index(similarFileFound)
```

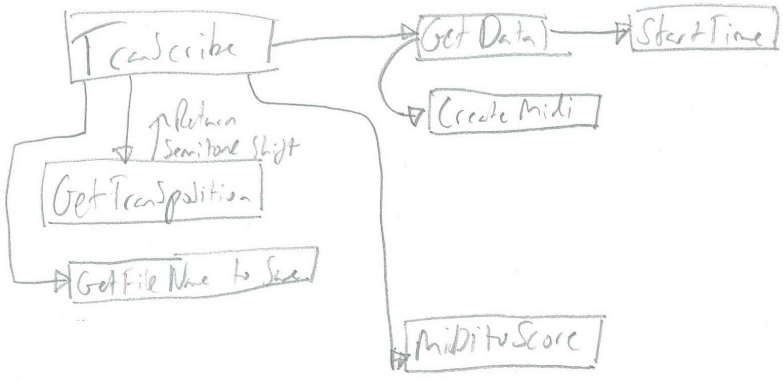
This function will find the closest file name in a given list. First it uses difflib to find the closest name and then it outputs the index of the file.

findFile()

```
def findFile(fileToFind):
    for root,dirs,files in os.walk(cwd):
        for file in files:
            if fileToFind in file:
                path = "{}\\{}".format(root,file)
                possiblePaths.append(path)
                possibleFiles.append(file)
    return possiblePaths[findMostSimilarFile(fileToFind,possibleFiles)]
```

This function first searches through all the files and directories where it is called from and creates a list of all the files. Then it used the findMostSimilarFile function on the list and outputs the file path of the file we were searching for.

findAudioDevice.py

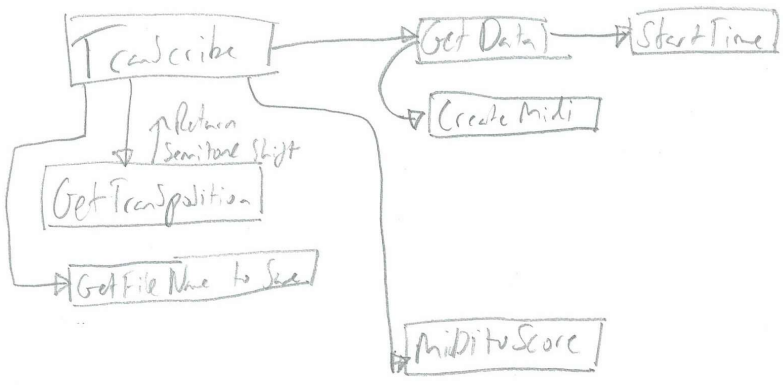


locate()

```
def locate():
        import pyaudio
        deviceList = list()
        p = pyaudio.PyAudio()
        info = p.get_host_api_info_by_index(0)
        numdevices = info.get('deviceCount')
        outputDevices = dict() ##deviceName, ID
        for i in range (0,numdevices):
                name = str(p.get_device_info_by_host_api_device_index(0,i).get('name'))
                if
p.get_device_info_by_host_api_device_index(0,i).get('maxInputChannels')>0:
                        if "microphone" in name.lower():
                                deviceList.append((name,i))
                                print ("*** MICROPHONE ***")
                                print ("Input Device id ", i, " - ", name)
                                print (p.get_device_info_by_host_api_device_index(0,i))
        return deviceList
```

This function searches the system for all available microphones. First it creates a pyaudio object and gets all the devices it can see, next it cycles through each device and determines if it is an input or an output, it does this by checking if it has any input channels, if it does it will then check if the word microphone is in the system devices name, if all checks are passed it returns the all the device information of that index.

dropboxUPLOAD.py



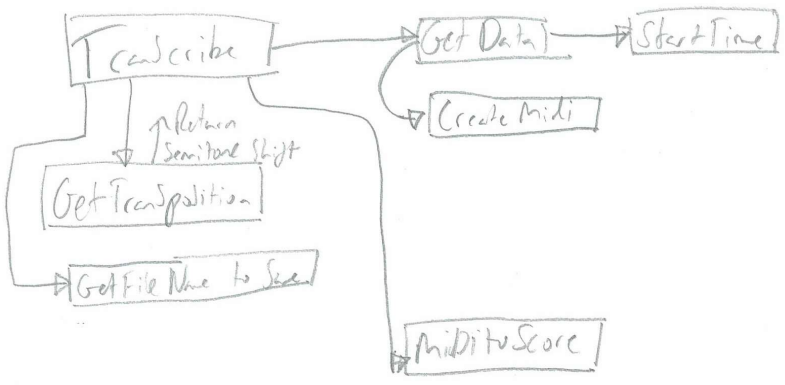
```
import dropbox
from dropbox.files import WriteMode
from dropbox.exceptions import AuthError
import pandas as pd
import os
import pyodbc
if __name__ == "__main__":
        midiFileName = 'out.mid'
        scoreFileName = 'uploadtest1.pdf'

        access_token = 'Iw2ShQdqqwAAAAAAAAAAFHPJuDGkEFBvA4vqaFqBes6W-YBZ_nzRwiVHfk7nDumj'
        fileName = ''
        filePathDropBox = "/"+'' #Name on dropbox

        connection = dropbox.Dropbox(access_token)
        try:
                connection.users_get_current_account()
        except AuthError as err:
           print ("[E] ERROR: Invalid access token; try re-generating an access token
from the app console on the web.")
```

The dropboxUPLOAD module is used to initialise database and file host connections and to execute uploads or queries. First the required modules are imported then a tunnel is created to the dropbox server using the access token to make sure there is a connection possible before allowing any other connections to be made.

connect_DB()

```
def connect_DB():
        import dropbox
        from dropbox.files import WriteMode
        from dropbox.exceptions import AuthError
        access_token = 'Iw2ShQdqqwAAAAAAAAAAFHPJuDGkEFBvA4vqaFqBes6W-
YBZ_nzRwiVHfk7nDumj'
        fileName = ''
        filePathDropBox = "/"+'' #Name on dropbox
        connection = dropbox.Dropbox(access_token)
        try:
                connection.users_get_current_account()
                return connection
        except AuthError as err:
                print ("[E] ERROR: Invalid access token; try re-generating an access
token from the app console on the web.")
```

The connect_DB function creates a connection to the dropbox server and checks it was successful. This connection is kept open in the connection variable.



upload()

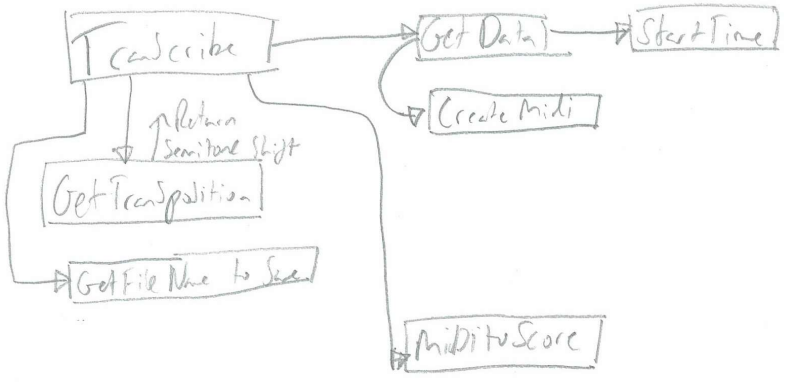
```
def upload(midiFileName,scoreFileName):
    connection = connect_DB()
    fileType = ("midi","score")
    files = [midiFileName,scoreFileName]
    links = list()
    for upload in files:
        path = ("/"+fileType[files.index(upload)]+"/"+str(upload))
        f = open(upload,"rb")
        connection.files_upload(f.read(),path,WriteMode("overwrite"))
        f.close()
        save = connection.files_get_temporary_link(path)
        link = save.link.encode('utf-8')
        ##print ("File: {} ||| Path: {} ||| Hosted: {}".format(upload,path,link))
        links.append(link)
    return links[0],links[1],connection
```

The upload function takes the paths of the files to upload and adds them to the file host. FIrst a connection is made using the connect_DB function and then for each file it will upload it to the correct directory (either /midi/midiFileName or /score/scoreFileName), finally both urls are returned to access these files alongside the connection if it is needed for later usage.

connect_SQL()

```
def connect_SQL(NGROKPORT):
        try:
                cnxn = pyodbc.connect('DRIVER={SQL Server Native Client
11.0};SERVER=tcp:0.tcp.ngrok.io,'+NGROKPORT+';UID=oscarUser;PWD=wps;database=mainDB')
##            cnxn = pyodbc.connect("Driver={SQL Server Native Client 11.0};"
##                                  "Server=127.0.0.1;"
##                                  "port=:1433;"
##                                  "Database=mainDB;"
##                                  "Trusted_Connection=yes;")#Server=ANDROID3
        except Exception as e:
                print (e)
        return cnxn
```

The connect_SQL takes the NGROKPORT variable and creates a connection to the SQL Server, it used pyodbc to do this, the url for the database is 0.tcp.ngrok.io:NGROKPORT and the credentials are also provided. This will return a cnxn to the mainDB table on the server. If the server fails to respond or connect an error will be displayed. This connection



is returned for use.

query()

```
def query(cmd,cnxn):
        try:
            df = pd.read_sql_query(cmd,cnxn) ##use single quotes for ntext
            cnxn.commit()
            return df

        except Exception as e:
                print ("[E] Error: {}".format(e))
```

This query function executes a SQL query on the server, it uses pandas to do this so that the result is formattable to display in the results table (df), if there was any error in the queries execution a suitable error is outputted.

checkHash()

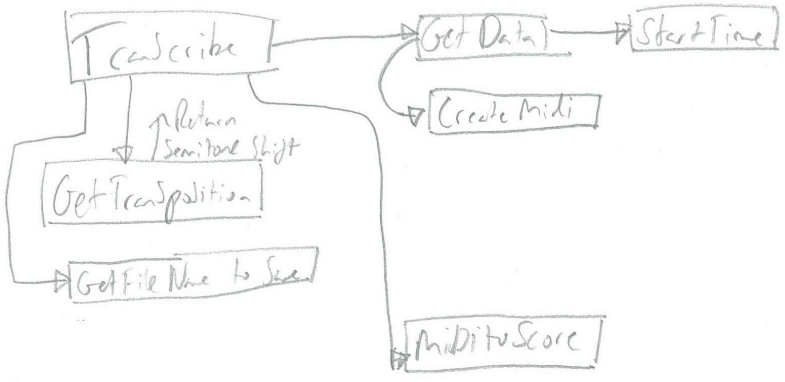
```
def checkHash(HASH,column,cnxn):
        unique = False
        df = query("SELECT {0} FROM info WHERE [{0}]=({1})".format(column,HASH),cnxn)
        if len(df) == 0:
                unique = True
        else:
                unique = False
        return unique
```

The checkHash function is used to see if a hash is unique or not, it does this by searching the 'info' table for the hash wanting to be checked, if there are any results then the hash isn't unique to the variable unique is toggled to False, otherwise unique is toggled to True and the hash is unique.

start()

```
def start(NGROKPORT):
        global cnxn
        cnxn = connect_SQL(NGROKPORT)
        return cnxn
```

The start function creates a cnxn to the sql server using the connect_SQL function, it then returns the connection to be used by other programs.

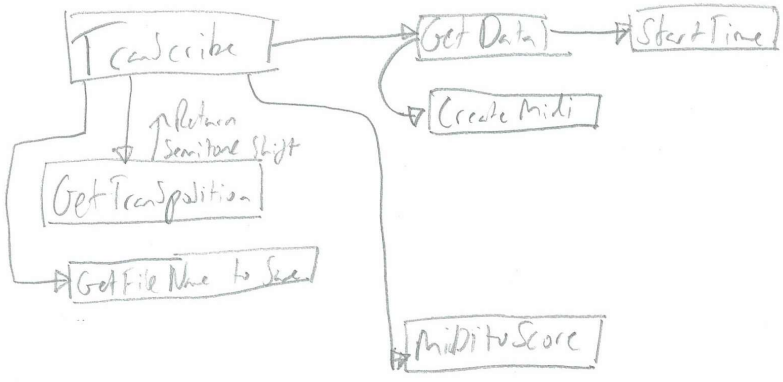


stop()

```
def stop():
        cnxn.close()
```

The stop function closes the connection to the sql server.

StyleSheet Files



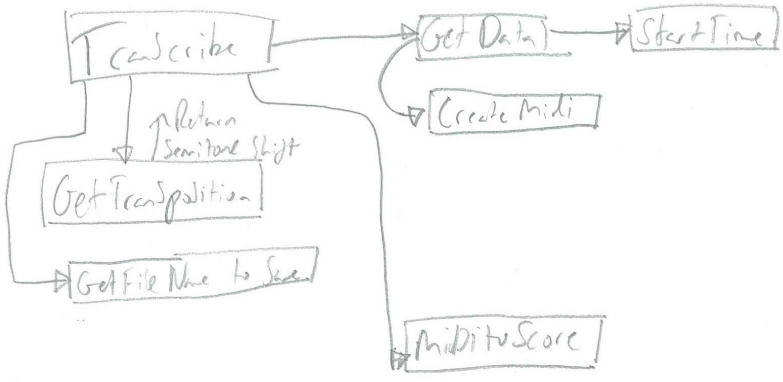
settingsWindow.stylesheet

```
QPushButton#applyAudioSettingsButton{
    background-color: green;
    border-color: black;
    font: bold;
}
QPushButton#refreshAudioDevicesBUtton{
    background-color: #939393;
    border-color: black;
    font: bold;
}

QWidget{
        background-color: #B3B3B3;
        }

QComboBox {
    border: 1px;
        border: black;
    border-radius: 1px;
        background-color: #E3E3E3;
}


QLabel#audioInputDeviceLabel{
        border: 0px;
        border-color: black;
        font: bold;
        }
```

This stylesheet is for the settingsWindow, it sets the applyAudioSettingsButton to have the background colour of green, a black border and its font bold.
The PushButton is set with the background colour of hex #939393 and the border colour of black and it's font bold. The ComboBox is set with a border of 1 pixel, and the colour black, its radius to 1 pixel and the background colour of hex #E3E3E3. The audioInputDeviceLabel has no border and its font in bold.

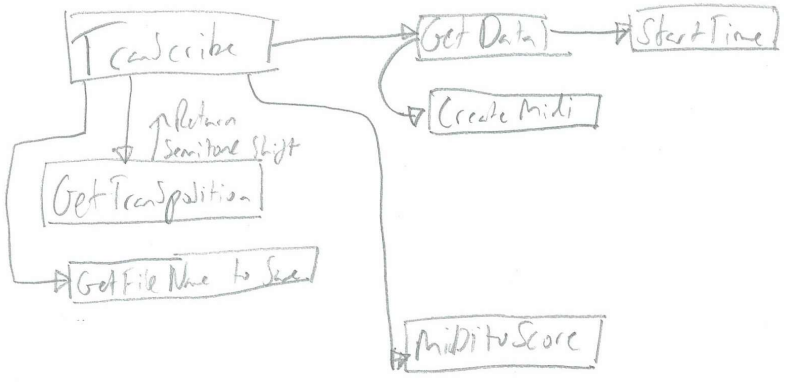
transcribeWindow.stylesheet

```
QLineEdit{
        border:2px;
        border-color:black;
        background-color:#E6E6E6;
```



```
        }

QDial {
        background-color: #E6E6E6;
        }
QLabel{
        font:bold;}

QComboBox {
    border: 0px;
        border: black;
    border-radius: 0px;
        background-color: #E6E6E6;
}

QSplitter::handle{
        image: None;}

QPushButton#startButton{
        border: 2px;
        border-color:black;
        background-color:green;
        border-radius: 5px;
        }
QPushButton#stopButton{
        border: 2px;
        border-color:black;
        background-color:red;
        border-radius: 5px;
        }
QPushButton#resetButton{
        border: 2px;
        border-color:black;
        background-color:#E6E6E6;
        border-radius: 5px;
        }
QMainWindow{
        background-color:#B3B3B3;
        }
```

The stylesheet sets all the input boxes (lineEdits) to have a border of 2px and the border colour of black, the background colour is also hex #E6E6E6. All Dials are set to have the background colour hex #E6E6E6 and all labels are set to bold. All dropdown boxes (ComboBox) are set to have no border and the background colour of hex #E6E6E6. The startButton is set to have a black border of 2px and a background colour of green and radius 5px. The stop button is the same as the start button except the background colour is red, the reset button is also the same but the background is hex #E6E6E6. Finally the Main window has the background colour hex #B3B3B3.



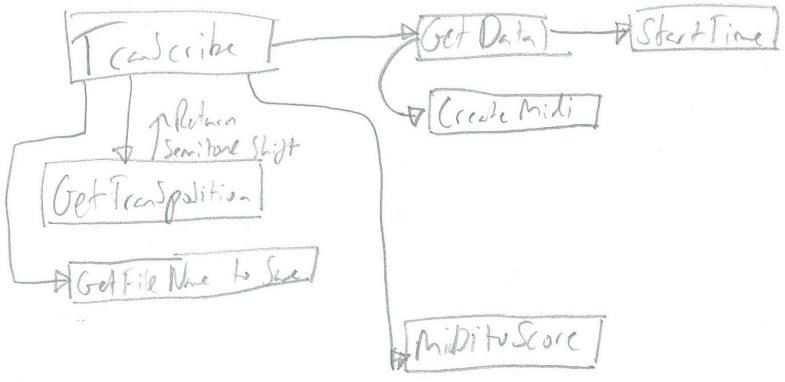
transWindow.stylesheet

```
QPushButton#startButton{
    background-color: green;
    border-color: black;
    font: bold 20px;
}
QPushButton#stopButton{
    background-color: red;
    border-color: black;
    font: bold 20px;
}

QDialog{
        background-color: #B3B3B3;
        }

QComboBox {
    border: 0px;
        border: black;
    border-radius: 0px;
        background-color: #E6E6E6;
}

QSplitter::handle{
        image: None;}

QDial {
        background-color: #E6E6E6;
        }

QLabel#currentNoteImage{
        border: 0px;
        border-color: black;
        }
```

This stylesheet is for the transposition window, it sets the push button to have the background colour green and the border colour black. The text inside is bold and has size 20px.The stop button is the same but it has the background colour red. The window colour is set to hex #B3B3B3. The dropdown box (ComboBox) doesn't have a border and has the background colour is hex #E6E6E6. The dials are set to have the background colour #E6E6E6. Finally the Label currentNoteImage doesn't have a border.

errorWindow.stylesheet

```
QMainWindow{
```

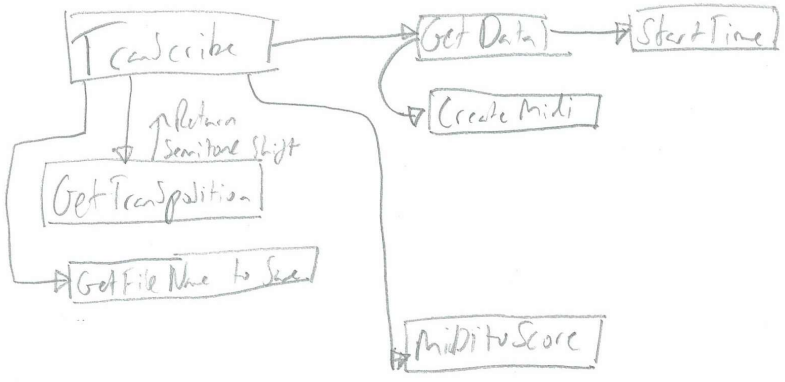


```
        background-color: #FF6961;
        }
QLabel{
        font: bold;
        }

QPushButton{
        border: 4px;
        border-color:#FF6961;
        background-color:#E6E6E6;
        border-radius: 5px;
        font:bold;
        }
```

This stylesheet is for the error WIndow, it sets the background colour of the window to hex #FF6961, the labels to have a bold font and the buttons to have border of 4px and colour #FF6961,radius 5px. Its background colour is set to hex #E6E6E6 and its font to bold.

sqlWindow.stylesheet

```
QMainWindow{
        background-color: #B3B3B3;
        }
QLabel{
        font: bold;
        }
QTableView{
        background-color:#E6E6E6;
        selection-background-color:#779ECB;
        border:5px;
        border-color:black;
        border-radius:5px;
        }

QHeaderView::section{
        background-color:#B3B3B3;
        font: bold;}

QSplitter::handle{
        image: None;}

QPushButton{
        border: 4px;
        border-color:black;
        background-color:#E6E6E6;
        border-radius: 5px;
        }
```



This is the stylesheet for the sql Window, it sets the windows background colour is hex #B3B3B3 and the labels to bold. It changes the results table to have background colour hex #E6E6E6 and the selected items background colour to hex #779ECB, it also gives the table a border of 5px ,radius of 5px and the border colour of black. The header of the table is also given the background colour of hex #B3B3B3 and a bold font. Finally the buttons to have border 4px, radius 5x and colour black, the background colour is set to hex #E6E6E6.

startUpWindow.stylesheet

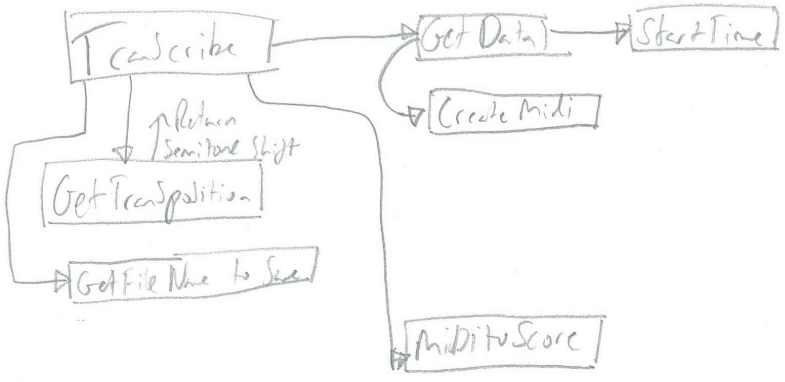
```
QPushButton {
    background-color: #E6E6E6;
    border-style: outset;
    border-width: 1px;
    border-radius: 5px;
    border-color: black;
    font: bold 20px;
    min-width: 10em;
    padding: 3px;
}

QMainWindow {
        background-color: #B3B3B3;
        }

QSplitter::handle{
        image: None;}
```

This is the stylesheet for the main startup window it sets all buttons to have the background colour hex #E6E6E6, and a black border of width 1px, radius 5px and style outset. It then sets the fonts to bold and size 20px with the width being 10em and padding of 3px. The background colour for the window is set to hex #B3B3B3.

Convert.bat

```
@echo off
start "CONVERT" "Musescore 3\bin\MuseScore3.exe" %~1 -o %~2%.pdf
EXIT /B 0
```

This MS-DOS script runs the Musescore3 executable with the parameters of the midi file to convert and the destined output file in pdf form. The script then exits, it is also set to have no visual output so the user can't see what is happening.



4097 Oscar Lindenbaum 62113 Wheatley Park School

Computer Science NEA - NoteSwitch

Video of Usage

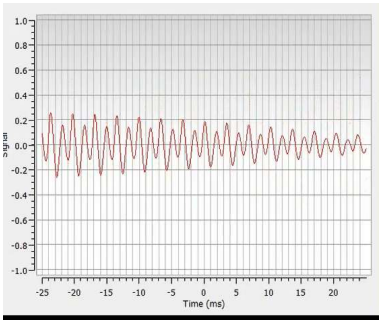
https://drive.google.com/file/d/1sL_0XPjI6AR8EHLoveQDJ4p4HTfJVsic/view?usp=sharing

Testing

Dry Run of Algorithms

Note Identification

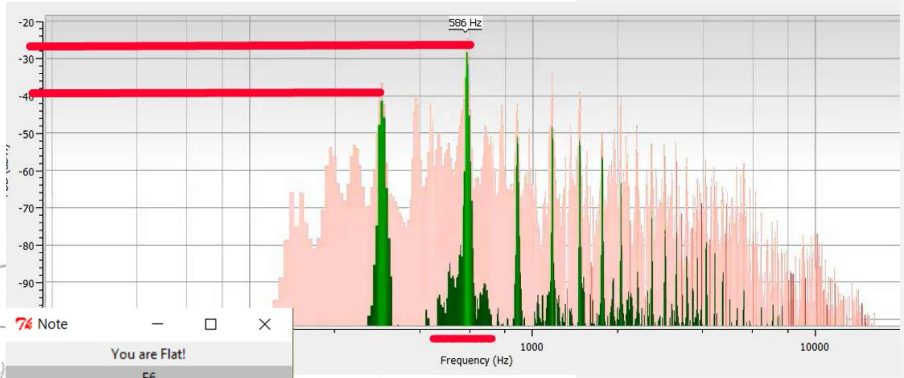
In order to test that the note identification work I will make a seperate program that Identifies the note and displays the output. By doing this I can make sure that the backbone functionality of the program works as planned.



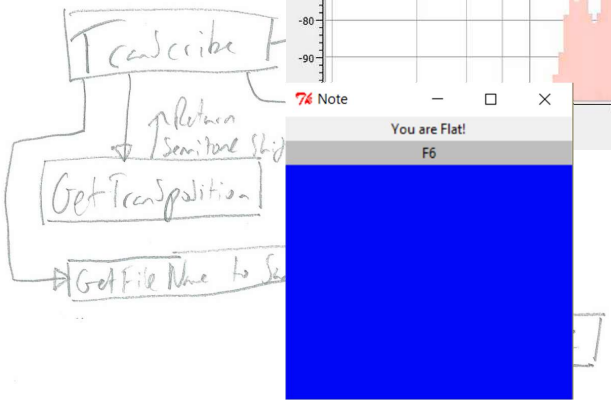
How will it work?

Shown to the left is the oscilloscope output when I play the note D in the key of C. What I will need to do is apply a Fourier Transform to separate this wave into its fundamental frequencies.

Below is the output after applying the transform. As you can see there are peaks at certain frequencies - the notes. To determine which note is being played, I will simply select the one with the highest Amplitude (tallest) as indicated in the diagram this is the green spike underlined in red. The note in the example would be a D (582Hz) which is the closest value to 596Hz.



The test program will display the note it thinks is being played and the colour indicates how flat or sharp you are, with green being perfect.

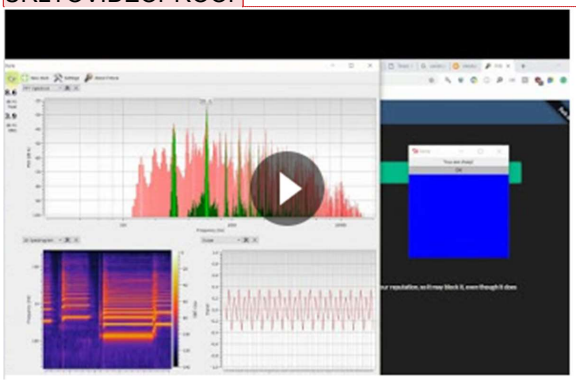


4097 Oscar Lindenbaum 62113 Wheatley Park School

Computer Science NEA - NoteSwitch

In order to see if this works I will use a tone generator and directly feed the audio into the program. Initially I played an A, then an F, then F up an octave and the program managed to successfully identify these notes with reasonable speed.

URLTOVIDEOPROOF



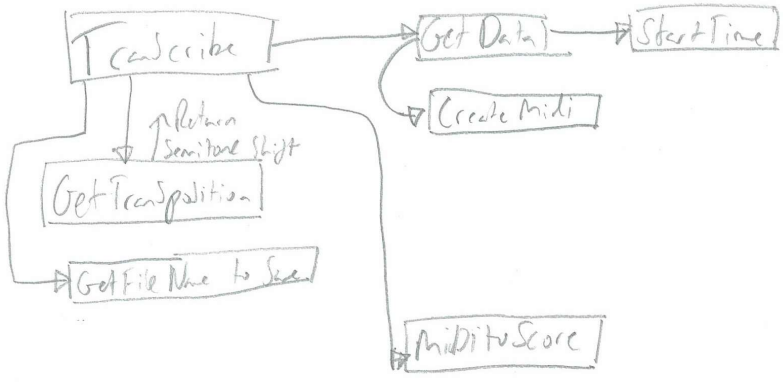
https://drive.google.com/open?id=1aWDbZVpEiakwap9nItqtSkVu_-qiNtly
https://drive.google.com/open?id=1j-o2F2ZwRxUeoWv5rKXWzP13g2EeTBZS

Uploading File to Dropbox

To check the functionality of the upload process I will create a working proof of concept that uploads test files onto the filehost.

URLTOVIDEOPROOF

https://drive.google.com/open?id=1wGFG-XnBddz3X2bXMkzWZr_Pu1XgsinQ



Commented [10]: need to upload video noteIdentificationDryRun

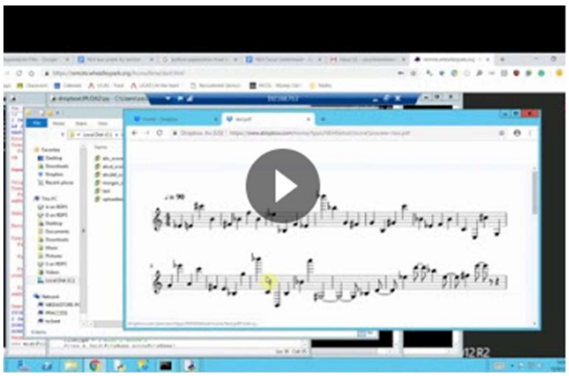
Commented [11]: noteidentificationDryRun2

Commented [12]: add video dropboxuploaddryrun

Commented [13]: dropboxupload2

4097 Oscar Lindenbaum 62113 Wheatley Park School

Computer Science NEA - NoteSwitch

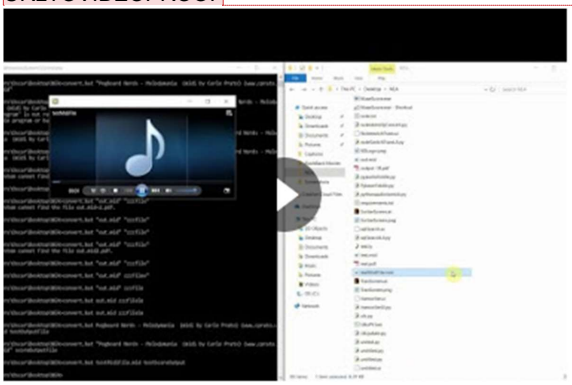


https://drive.google.com/open?id=1u4n2Je_272WwBRqZRjilg8o4qtt8czh5

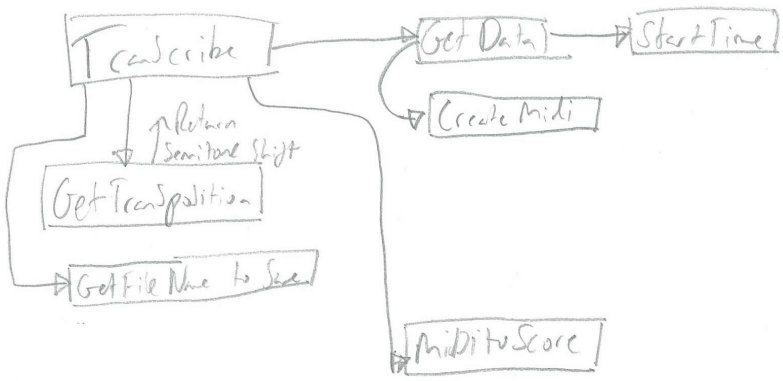
Converting MIDI File to Score File

URLTOVIDEOPROOF

Commented [14]: Add video convertToScoreDryRun



https://drive.google.com/open?id=1MitwYl06h-U9F_hqiluHsF2Zc_YzfSVt



Data Validation

For most of the application there is no need for data validation as the data is either computer generated or the user can only pick from predefined options. One of the places where they can is they can enter a filename and a username in the transcribe window, the validation that needs to go here is to make sure the filename only contains characters accepted for a typical file, or that the text box is empty. This is when the file is created, if successful the application continues otherwise the user is told that the file wasn't created and to try again. If the text box is empty then a suitable error is shown and the applications halts. A requirement of the filename is that is has to be unique which is checked when updating the database, depending if the new entry can be added.

The other window where text can be freely entered is the SQL Search window, this doesn't need to be filtered as if there are no files similar to the search queries then no results will be shown.

Are the objectives met?

1.1.1	**The user is presented with a selection of buttons, "Transcribe" and "Transpose" which opens the correct window.**

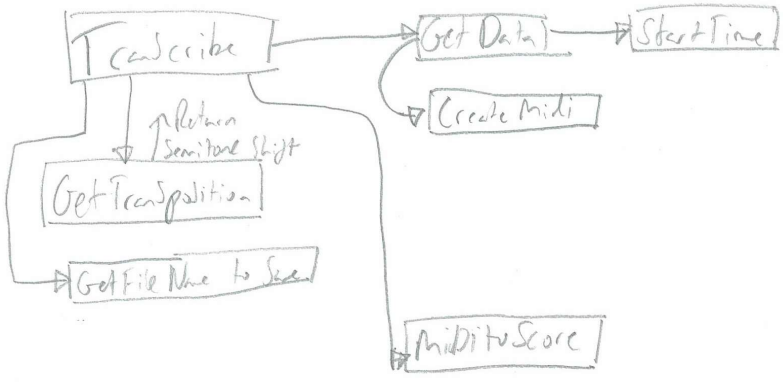
Criteria Met. See Video 1 - https://drive.google.com/file/d/14OmxAzsB6GzQJMS-vZYRXTqB7m-mLLHt/view?usp=sharing

2.1.1	**The program can successfully identify the frequency and the correct note so it can display the correct note.**

Criteria Met. See Video 2 - https://drive.google.com/file/d/16z-RrxvNQML2C519TawuSr9UQTMjQnar/view?usp=sharing

2.1.2	**The user can select the transposition value and this changes the transposition output of the program.**

Criteria Met. See Video 2 - https://drive.google.com/file/d/16z-RrxvNQML2C519TawuSr9UQTMjQnar/view?usp=sharing



2.1.3	**The correct transposed note is displayed for the user as text.**

Criteria Met. See Video 2 - https://drive.google.com/file/d/16z-RrxvNQML2C519TawuSr9UQTMjQnar/view?usp=sharing

3.1.1	**The user can view past scores and MIDI files.**

Criteria Met. See Video 1 - https://drive.google.com/file/d/14OmxAzsB6GzQJMS-vZYRXTqB7m-mLLHt/view?usp=sharing

3.1.2	**The user can download the score and MIDI files for future use.**

Criteria Met. See Video 1 - https://drive.google.com/file/d/14OmxAzsB6GzQJMS-vZYRXTqB7m-mLLHt/view?usp=sharing

4.1.1	**It can create a midi representation of an audio file or live audio stream.**

Criteria Met. See Video 1 - https://drive.google.com/file/d/14OmxAzsB6GzQJMS-vZYRXTqB7m-mLLHt/view?usp=sharing

4.1.2	**It can convert the generated midi file into a musical notation such as a score which is outputted for the user.**

Criteria Met. See Video 1 - https://drive.google.com/file/d/14OmxAzsB6GzQJMS-vZYRXTqB7m-mLLHt/view?usp=sharing

1.2.1	The user can select a “Search” button which opens the server and allows them to search through the database for past scores and recordings.

Criteria Met. See Video 1 - https://drive.google.com/file/d/14OmxAzsB6GzQJMS-vZYRXTqB7m-mLLHt/view?usp=sharing

1.3.1	The user can open the “Settings” window, this would allow the user to select the audio devices manually.

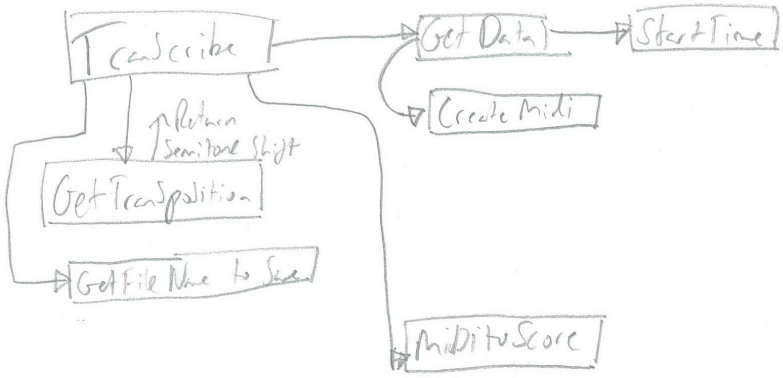
Criteria Met. See Video 1 - https://drive.google.com/file/d/14OmxAzsB6GzQJMS-vZYRXTqB7m-mLLHt/view?usp=sharing

1.3.2	The program has an additional function for tuning instruments.

Criteria Not Met - This was an optimal objective so it doesn’t affect the current application. It could be implemented in later versions.

2.2.1	The correct transposed note is displayed for the user as a musical representation.

Criteria Met. See Video 2 - https://drive.google.com/file/d/16z-RrxvNQML2C519TawuSr9UQTMjQnar/view?usp=sharing



2.2.2	The user can change the sensitivity of the note detection (The margin of *cents* determining which note is being played) *One semitone is equal to 100 cents.*

Criteria Met. See Video 2 - https://drive.google.com/file/d/16z-RrxvNQML2C519TawuSr9UQTMjQnar/view?usp=sharing

2.3.1	The user can pick from a list of instruments which has the key already paired with them so that they don't need to find out the key of their instruments.

Criteria Not Met - This was an optimal solution so it doesn't affect the current application. Depending on the feedback from the end user, this could be easily implemented.

3.2.1	The user can search through all of the scores with different parameters.

Criteria Met. See Video 1 - https://drive.google.com/file/d/14OmxAzsB6GzQJMS-vZYRXTqB7m-mLLHt/view?usp=sharing

3.3.1	The user can select an entry and 'preview' the files without downloading them.

Criteria Not Met - This was an optimal objective so it doesn't affect the current application, there is an alternative as the user can download and view the files, it could be implemented by temporarily saving the files and showing them in another window and then deleting the file when not in use.

4.2.1	The score has the correct quantized note lengths.

Criteria Met. See Video 1 - https://drive.google.com/file/d/14OmxAzsB6GzQJMS-vZYRXTqB7m-mLLHt/view?usp=sharing

4.2.2	The program can be changed so that the user enters the instruments starting key and the desired end key which then changes the transposition of the notes.

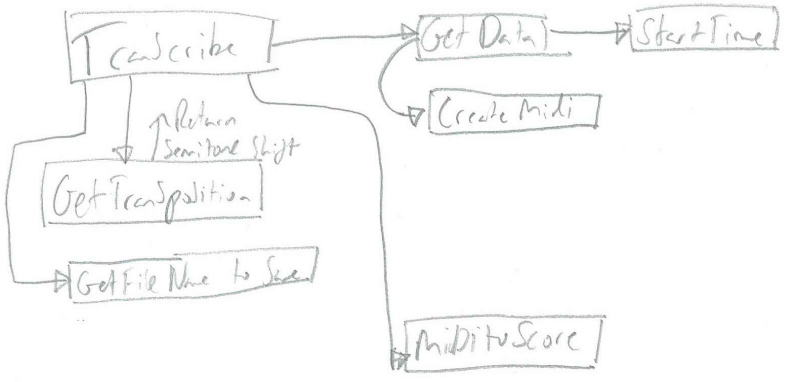
Criteria Met. See Video 1 - https://drive.google.com/file/d/14OmxAzsB6GzQJMS-vZYRXTqB7m-mLLHt/view?usp=sharing

4.2.3	The score has the best 'spellings' of notes displayed i.e replacing double sharps with a flat so it is easier to read.

Criteria Met. See Video 1 - https://drive.google.com/file/d/14OmxAzsB6GzQJMS-vZYRXTqB7m-mLLHt/view?usp=sharing

4.2.4	The program outputs both the score file and the midi file so it can then be edited in other programs such as MuseScore for corrections.

Criteria Met. See Video 3 - https://drive.google.com/file/d/1JR4EhcgiUH21I_SZCkDYND6bSeDnFPew/view?usp=sharing



4.2.5	The program can detect the Tempo automatically.

Criteria Met. See Video 1 - https://drive.google.com/file/d/14OmxAzsB6GzQJMS-vZYRXTqB7m-mLLHt/view?usp=sharing

4.2.6	The program can detect the time signature automatically.

Criteria Met. See Video 1 - https://drive.google.com/file/d/14OmxAzsB6GzQJMS-vZYRXTqB7m-mLLHt/view?usp=sharing

4.3.1	The user can pick from a list of instruments which has the key already paired with them so that they don't need to find out the key of their instruments.

Criteria Not Met - This was an Optimal objective so it doesn't affect the current application, this would be easily implemented by changing the list used for the keys.

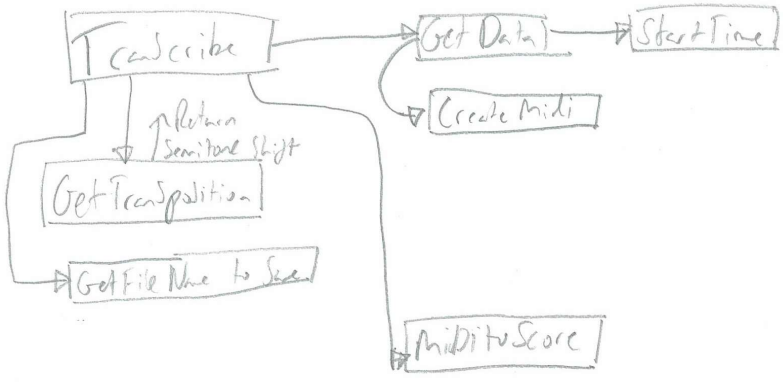
4.3.2	The program can record multiple parts for a larger composition and merge all of the scores together for a conductor's score.

Criteria Not Met - This was an Optimal objective so it doesn't affect the current application. This would be a different component as it would involve merging the created midi tracks post recording.

Individual Tests

Main Menu Testing - https://drive.google.com/file/d/1E_pu0m8-XQh4RWpdRYWf6l8glkSlPB2e/view?usp=sharing

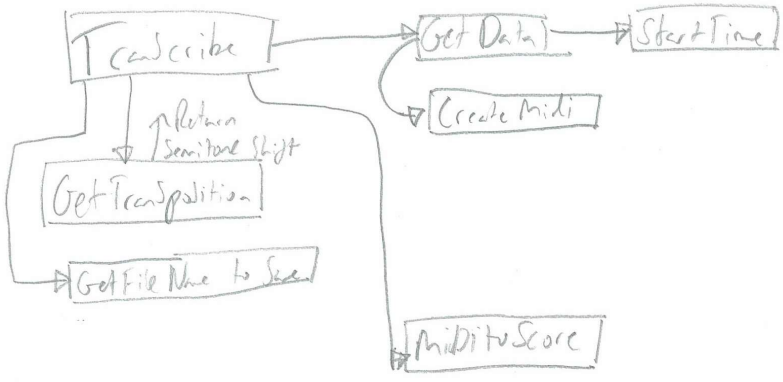
Test No.	Test Case	Expected Outcome	Actual Outcome
1	Clicking on the Transcribe button	The Transcribe window opens	The transcribe window opens successfully
2	Clicking on the Transpose button	The Transpose window opens	The transpose window opens successfully
3	Clicking on the Search button	The Search window opens	The search window opens successfully



4	Clicking on the Settings button	The Settings window opens	The settings window opens successfully
5	Clicking anywhere else	There is no response	No Response
6	Clicking the Exit button	The window closes	The window closes

Transpose Testing - https://drive.google.com/file/d/1FTS7Pk6TNr3VMu3F1Z-WLcS6RD9lZlfP/view?usp=sharing

Test No.	Test Case	Expected Outcome	Actual Outcome
1	Changing the start key using the dial.	The starting key is changed and so is the dropdown box to match	Both change accordingly
2	Changing the start key using the dropdown box	The starting key is changed and so is the dial box to match	Both change accordingly
3	Changing the end key using the dial.	The ending key is changed and so is the dropdown box to match	Both change accordingly
4	Changing the end key using the dropdown box	The ending key is changed and so is the dial box to match	**The dropdown box changes but the position of the dial doesn't**
5	Changing the sensitivity using the slider	The sensitivity value changes	The sensitivity value changes
6	Pressing the start button without the inputs defined	The live analysis doesn't start and an error message is shown	An error message is shown
7	Pressing the start button with the inputs defined	The live analysis starts and the user inputs are disabled	The live identification starts and the inputs are disabled



8	Pressing start with the audio device settings not set	The live analysis doesn't start and an error message is shown	An error saying 'adjust settings' is shown
8	Pressing the stop button	The live analysis stops and the user inputs and enabled	The analysis stops and the inputs are enabled
9	Clicking the Exit button	The window closes	The window closes

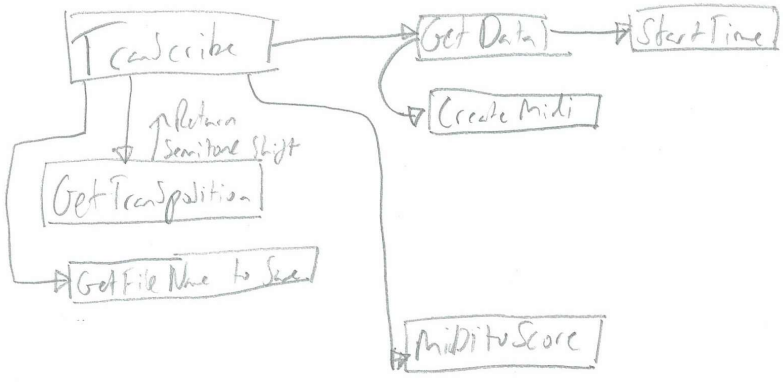
The transpose window passes all but one of its tests, test 4. This is where the ending key dial relative to the dropdown box is not updated when the user selects using the dropdown box. Although this doesn't impede the applications output it might be misleading to the user and can be fixed. **FIX TRS04**

Settings Window Testing -
https://drive.google.com/file/d/1DfsMkEhB03AmbMoZOr482mgl488C4tK_/view?usp=sharing

Test No.	Test Case	Expected Outcome	Actual Outcome
1	Clicking on the refresh button	The dropdown box is added with any new audio devices.	The audio devices are refreshed
2	Selecting an audio device	The device can be selected from the dropdown box	Audio devices can be selected
3	Pressing apply button	A file called audioSettingsFile is created containing the details and a success message is shown	A success message is displayed and the audioSettingsFile is created.
4	Clicking the Exit button	The window closes	The window closes

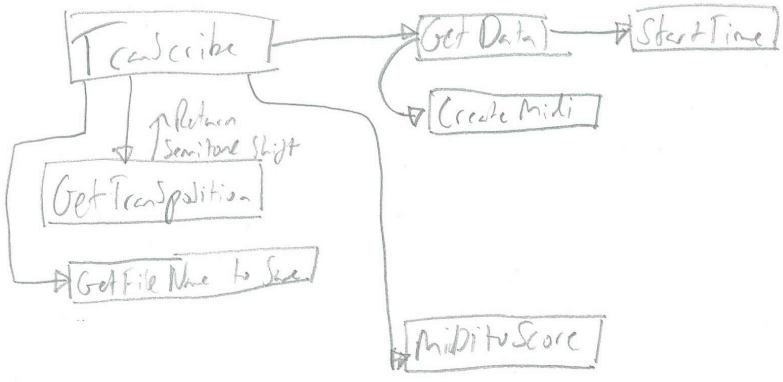
SQL Search Window Testing -
https://drive.google.com/file/d/1FgMAYeZMzEc09zdeyNmVlw38NfGkF53F/view?usp=sharing

Test No.	Test Case	Expected Outcome	Actual Outcome

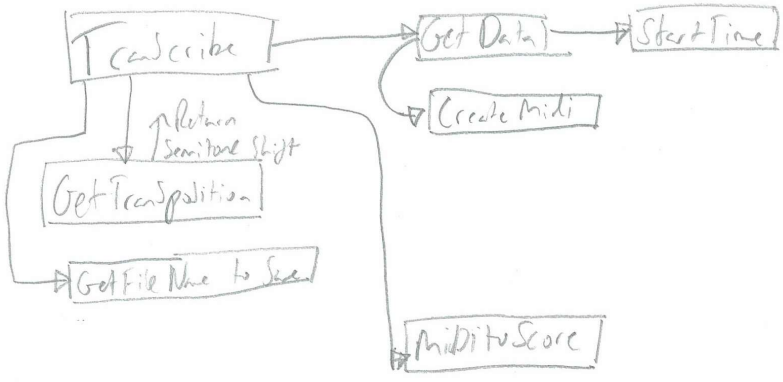


1	Clicking View All	All the entries from the database are shown in the table	All entries are shown in the database
2	Searching by Username	Entries matching the username will be shown in the table	Results similar to the username are shown
3	Searching by File Name	Entries matching the file name will be shown in the table	Results similar to the file name are shown
4	Searching by date	Entries matching the date will be shown in the table	Results similar to the date are shown
5	Pressing download button without selecting any options	An error is shown	An error is shown
6	Pressing download button selecting only the type	An error is shown	An error is shown
7	Pressing download button selecting only the item	An error is shown	An error is shown
8	Pressing download button with all inputs set	The files are downloaded to the current directory.	The files are successfully downloaded
9	Clicking on the exit button	The window closes	The window closes

Transcribe Window Testing
Video 1-
https://drive.google.com/file/d/1cWGWU7e4M1QosSRvTAnT22igbOlHXM8j/view?usp=sharing
Video 2-
https://drive.google.com/file/d/19esn8Pc46u3qr_GyWSvX5gMU1NEhuZYC/view?usp=sharing



Test No.	Test Case	Expected Outcome	Actual Outcome
1	Changing the start key using the dial.	The starting key is change and so is the dropdown box to match	Both change accordingly
2	Changing the start key using the dropdown box	The starting key is change and so is the dial box to match	**The value is changed but the dial isn't updated to match**
3	Changing the end key using the dial.	The ending key is change and so is the dropdown box to match	Both change accordingly
4	Changing the end key using the dropdown box	The ending key is change and so is the dial box to match	**The value is changed but the dial isn't updated to match**
5	Pressing start without entering any credentials	An error is shown	***See video 2*** **An error is shown but sometimes the timer and threads are started causing the program to crash**
6	Pressing start without setting the audio device	An error is shown	*See video 2* An error saying 'adjust settings' is shown
7	Pressing start with all inputs specified	The program starts recording the user and the timer starts. All inputs are disabled	The timer starts and the inputs are disabled
8	Pressing the stop button	The recording is stopped and analysed. The files are then created and uploaded to the server. All inputs are enabled	The timer stops and the inputs are enabled. The files are uploaded to the server and a success message is shown



9	Pressing the reset button	All inputs are set to their default value and enabled.	The inputs go back to their default values, they are also enabled
10	Pressing the exit button	The window closes	The window closes

The transcribe window also passed the majority of its tests apart from 2,4 and 5. Tests 2 and 4 is a replication of TRS04 but in a different window which can be fixed easily. - **FIX TRC02 TRC 04**

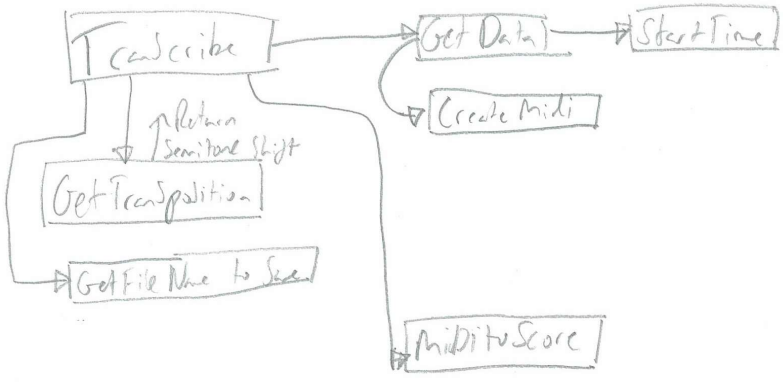
The other error is caused when the text credentials are not inputted before the start button is pressed. This causes a crash because the threads are started regardless when the start button is pressed and cannot be started twice. - **FIX TRC05**

Note Identification

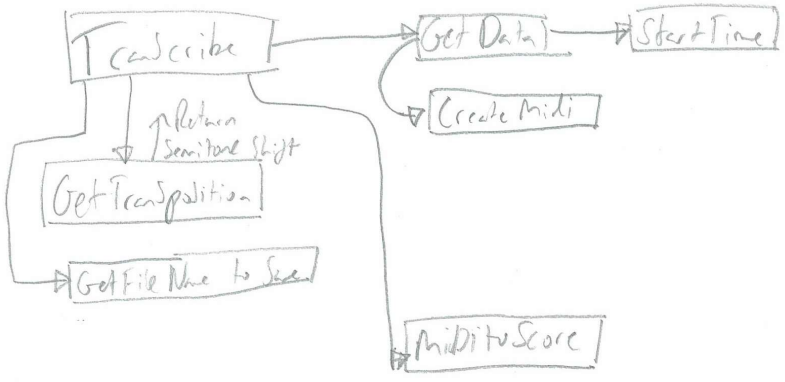
Using the smaller program from earlier which I used to identify the pitch

BOUNDARYVIDEO[1:215] EXACTVIDEO[216:]

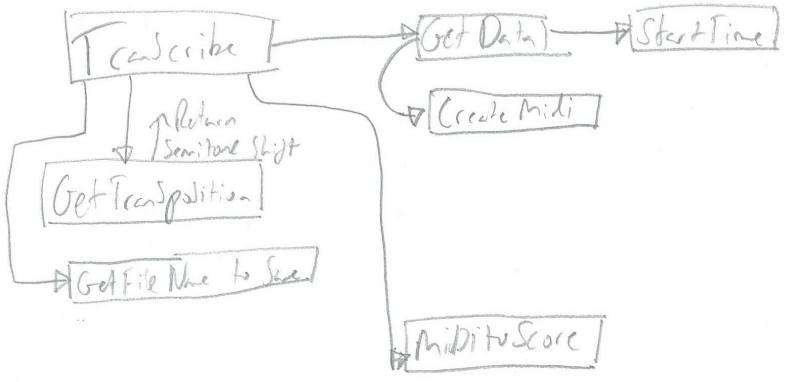
Test No.	Test Case	Expected Outcome	Actual Outcome
1	No Transposition (C to C) Play tone of 16.64 (LOWER BOUND)	The program should interpret this audio as a note of C0	Note identified as F1
2	No Transposition (C to C) Play tone of 17.04 (UPPER BOUND)	The program should interpret this audio as a note of C0	Note identified as F1
3	No Transposition (C to C) Play tone of 17.64 (LOWER BOUND)	The program should interpret this audio as a note of C#0/Db0	Note identified as F1
4	No Transposition (C to C) Play tone of 18.04 (UPPER BOUND)	The program should interpret this audio as a note of C#0/Db0	Note identified as F1
5	No Transposition (C to C) Play tone of 18.7 (LOWER BOUND)	The program should interpret this audio as a note of D0	Note identified as F1



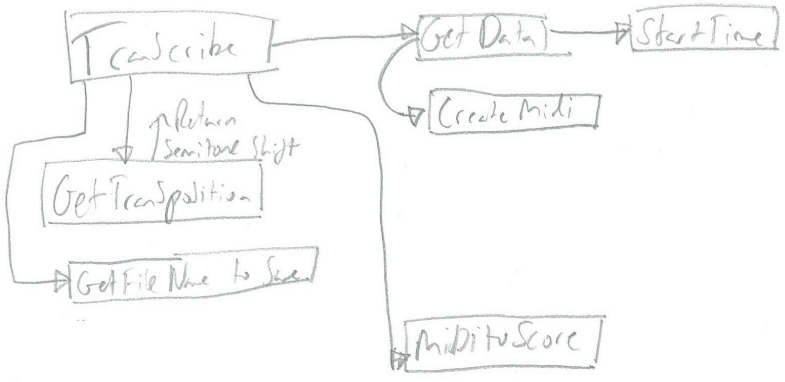
6	No Transposition (C to C) Play tone of 19.1 (UPPER BOUND)	The program should interpret this audio as a note of D0	Note identified as F1
7	No Transposition (C to C) Play tone of 19.82 (LOWER BOUND)	The program should interpret this audio as a note of D#0/Eb0	Note identified as F1
8	No Transposition (C to C) Play tone of 20.22 (UPPER BOUND)	The program should interpret this audio as a note of D#0/Eb0	Note identified as F1
9	No Transposition (C to C) Play tone of 21.01 (LOWER BOUND)	The program should interpret this audio as a note of E0	Note identified as F1
10	No Transposition (C to C) Play tone of 21.41 (UPPER BOUND)	The program should interpret this audio as a note of E0	Note identified as F1
11	No Transposition (C to C) Play tone of 22.28 (LOWER BOUND)	The program should interpret this audio as a note of F0	Note identified as F1
12	No Transposition (C to C) Play tone of 22.68 (UPPER BOUND)	The program should interpret this audio as a note of F0	Note identified as F1
13	No Transposition (C to C) Play tone of 23.61 (LOWER BOUND)	The program should interpret this audio as a note of F#0/Gb0	Note identified as F1
14	No Transposition (C to C) Play tone of 24.01 (UPPER BOUND)	The program should interpret this audio as a note of F#0/Gb0	Note identified as F1
15	No Transposition (C to C) Play tone of 25.03 (LOWER BOUND)	The program should interpret this audio as a note of G0	Note identified as F1
16	No Transposition (C to C) Play tone of 25.43 (UPPER BOUND)	The program should interpret this audio as a note of G0	Note identified as F1
17	No Transposition (C to C) Play tone of 26.53 (LOWER BOUND)	The program should interpret this audio as a note of G#0/Ab0	Note identified as F1



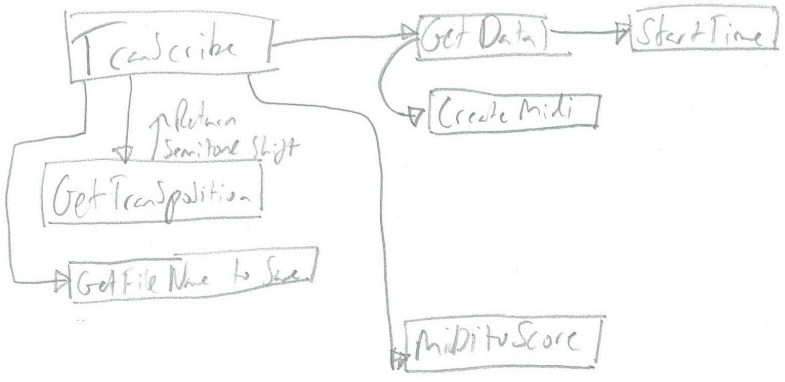
18	No Transposition (C to C) Play tone of 26.93 (UPPER BOUND)	The program should interpret this audio as a note of G#0/Ab0	Note identified as F1
19	No Transposition (C to C) Play tone of 28.12 (LOWER BOUND)	The program should interpret this audio as a note of A0	Note identified as F1
20	No Transposition (C to C) Play tone of 28.52 (UPPER BOUND)	The program should interpret this audio as a note of A0	Note identified as F1
21	No Transposition (C to C) Play tone of 29.81 (LOWER BOUND)	The program should interpret this audio as a note of A#0/Bb0	Note identified as F1
22	No Transposition (C to C) Play tone of 30.21 (UPPER BOUND)	The program should interpret this audio as a note of A#0/Bb0	Note identified as F1
23	No Transposition (C to C) Play tone of 31.59 (LOWER BOUND)	The program should interpret this audio as a note of B0	Note identified as F1
24	No Transposition (C to C) Play tone of 31.99 (UPPER BOUND)	The program should interpret this audio as a note of B0	Note identified as F1
25	No Transposition (C to C) Play tone of 33.47 (LOWER BOUND)	The program should interpret this audio as a note of C1	Note identified as F1
26	No Transposition (C to C) Play tone of 33.87 (UPPER BOUND)	The program should interpret this audio as a note of C1	Note identified as F1
27	No Transposition (C to C) Play tone of 35.48 (LOWER BOUND)	The program should interpret this audio as a note of C#1/Db1	Note identified as F1
28	No Transposition (C to C) Play tone of 35.88 (UPPER BOUND)	The program should interpret this audio as a note of C#1/Db1	Note identified as F1
29	No Transposition (C to C) Play tone of 37.6 (LOWER BOUND)	The program should interpret this audio as a note of D1	Note identified as F#1



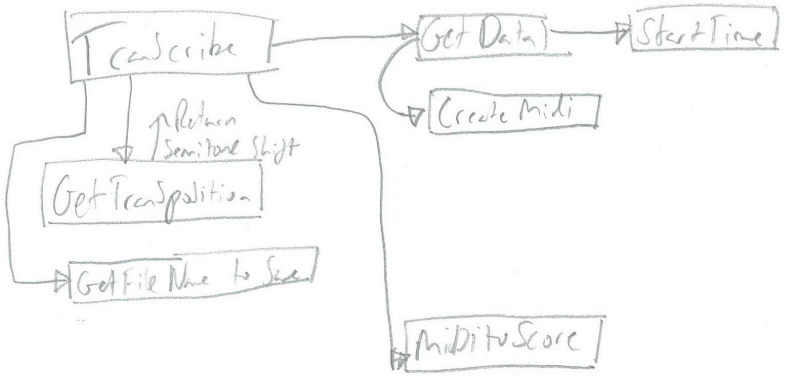
30	No Transposition (C to C) Play tone of 38.0 (UPPER BOUND)	The program should interpret this audio as a note of D1	Note identified as G1
31	No Transposition (C to C) Play tone of 39.85 (LOWER BOUND)	The program should interpret this audio as a note of D#1/Eb1	Note identified as G#1
32	No Transposition (C to C) Play tone of 40.25 (UPPER BOUND)	The program should interpret this audio as a note of D#1/Eb1	Note identified as A1
33	No Transposition (C to C) Play tone of 42.22 (LOWER BOUND)	The program should interpret this audio as a note of E1	Note identified as A#1
34	No Transposition (C to C) Play tone of 42.62 (UPPER BOUND)	The program should interpret this audio as a note of E1	Note identified as A#1
35	No Transposition (C to C) Play tone of 44.75 (LOWER BOUND)	The program should interpret this audio as a note of F1	Note identified as A#1
36	No Transposition (C to C) Play tone of 45.15 (UPPER BOUND)	The program should interpret this audio as a note of F1	Note identified as A#1
37	No Transposition (C to C) Play tone of 47.42 (LOWER BOUND)	The program should interpret this audio as a note of F#1/Gb1	Note identified as B1
38	No Transposition (C to C) Play tone of 47.82 (UPPER BOUND)	The program should interpret this audio as a note of F#1/Gb1	Note identified as B1
39	No Transposition (C to C) Play tone of 50.25 (LOWER BOUND)	The program should interpret this audio as a note of G1	Note identified as B1
40	No Transposition (C to C) Play tone of 50.65 (UPPER BOUND)	The program should interpret this audio as a note of G1	Note identified as B1
41	No Transposition (C to C) Play tone of 53.25 (LOWER BOUND)	The program should interpret this audio as a note of G#1/Ab1	Note identified as C2



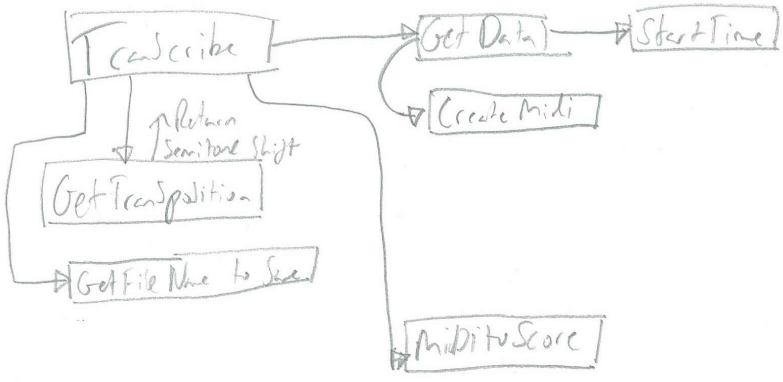
42	No Transposition (C to C) Play tone of 53.65 (UPPER BOUND)	The program should interpret this audio as a note of G#1/Ab1	Note identified as C2
43	No Transposition (C to C) Play tone of 56.44 (LOWER BOUND)	The program should interpret this audio as a note of A1	Note identified as C2
44	No Transposition (C to C) Play tone of 56.84 (UPPER BOUND)	The program should interpret this audio as a note of A1	Note identified as C2
45	No Transposition (C to C) Play tone of 59.81 (LOWER BOUND)	The program should interpret this audio as a note of A#1/Bb1	Note identified as C#2
46	No Transposition (C to C) Play tone of 60.21 (UPPER BOUND)	The program should interpret this audio as a note of A#1/Bb1	Note identified as C#2
47	No Transposition (C to C) Play tone of 63.38 (LOWER BOUND)	The program should interpret this audio as a note of B1	Note identified as D2
48	No Transposition (C to C) Play tone of 63.78 (UPPER BOUND)	The program should interpret this audio as a note of B1	Note identified as D2
49	No Transposition (C to C) Play tone of 67.15 (LOWER BOUND)	The program should interpret this audio as a note of C2	Note identified as D#2
50	No Transposition (C to C) Play tone of 67.55 (UPPER BOUND)	The program should interpret this audio as a note of C2	Note identified as D#2
51	No Transposition (C to C) Play tone of 71.16 (LOWER BOUND)	The program should interpret this audio as a note of C#2/Db2	Note identified as D#2
52	No Transposition (C to C) Play tone of 71.56 (UPPER BOUND)	The program should interpret this audio as a note of C#2/Db2	Note identified as D#2
53	No Transposition (C to C) Play tone of 75.4 (LOWER BOUND)	The program should interpret this audio as a note of D2	Note identified as E2



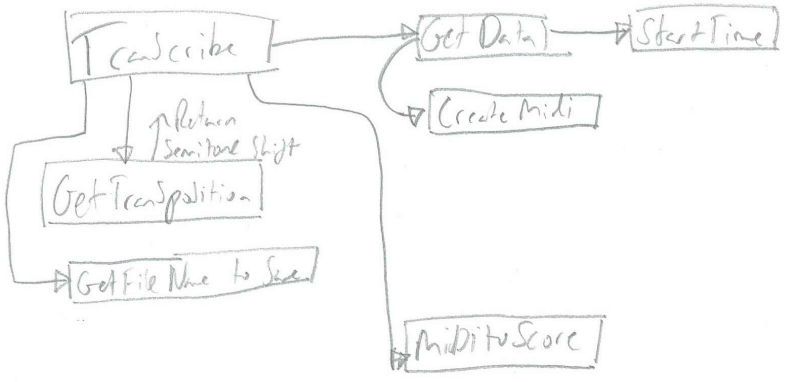
54	No Transposition (C to C) Play tone of 75.8 (UPPER BOUND)	The program should interpret this audio as a note of D2	Note identified as E2
55	No Transposition (C to C) Play tone of 79.89 (LOWER BOUND)	The program should interpret this audio as a note of D#2/Eb2	Note identified as F2
56	No Transposition (C to C) Play tone of 80.29 (UPPER BOUND)	The program should interpret this audio as a note of D#2/Eb2	Note identified as F2
57	No Transposition (C to C) Play tone of 84.66 (LOWER BOUND)	The program should interpret this audio as a note of E2	Note identified as F#2
58	No Transposition (C to C) Play tone of 85.06 (UPPER BOUND)	The program should interpret this audio as a note of E2	Note identified as F#2
59	No Transposition (C to C) Play tone of 89.71 (LOWER BOUND)	The program should interpret this audio as a note of F2	Note identified as G2
60	No Transposition (C to C) Play tone of 90.11 (UPPER BOUND)	The program should interpret this audio as a note of F2	Note identified as G2
61	No Transposition (C to C) Play tone of 95.05 (LOWER BOUND)	The program should interpret this audio as a note of F#2/Gb2	Note identified as G#2
62	No Transposition (C to C) Play tone of 95.45 (UPPER BOUND)	The program should interpret this audio as a note of F#2/Gb2	Note identified as G#2
63	No Transposition (C to C) Play tone of 100.71 (LOWER BOUND)	The program should interpret this audio as a note of G2	Note identified as G#2
64	No Transposition (C to C) Play tone of 101.11 (UPPER BOUND)	The program should interpret this audio as a note of G2	Note identified as A2
65	No Transposition (C to C) Play tone of 106.71 (LOWER BOUND)	The program should interpret this audio as a note of G#2/Ab2	Note identified as A2



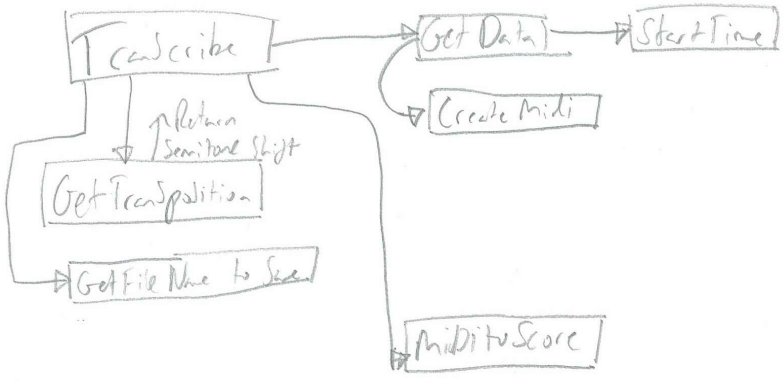
66	No Transposition (C to C) Play tone of 107.11 (UPPER BOUND)	The program should interpret this audio as a note of G#2/Ab2	Note identified as A2
67	No Transposition (C to C) Play tone of 113.07 (LOWER BOUND)	The program should interpret this audio as a note of A2	Note identified as A#2
68	No Transposition (C to C) Play tone of 113.47 (UPPER BOUND)	The program should interpret this audio as a note of A2	Note identified as A#2
69	No Transposition (C to C) Play tone of 119.8 (LOWER BOUND)	The program should interpret this audio as a note of A#2/Bb2	Note identified as B2
70	No Transposition (C to C) Play tone of 120.2 (UPPER BOUND)	The program should interpret this audio as a note of A#2/Bb2	Note identified as B2
71	No Transposition (C to C) Play tone of 126.94 (LOWER BOUND)	The program should interpret this audio as a note of B2	Note identified as C3
72	No Transposition (C to C) Play tone of 127.34 (UPPER BOUND)	The program should interpret this audio as a note of B2	Note identified as C3
73	No Transposition (C to C) Play tone of 134.5 (LOWER BOUND)	The program should interpret this audio as a note of C3	Note identified as C#3
74	No Transposition (C to C) Play tone of 134.9 (UPPER BOUND)	The program should interpret this audio as a note of C3	Note identified as C#3
75	No Transposition (C to C) Play tone of 142.51 (LOWER BOUND)	The program should interpret this audio as a note of C#3/Db3	Note identified as D3
76	No Transposition (C to C) Play tone of 142.91 (UPPER BOUND)	The program should interpret this audio as a note of C#3/Db3	Note identified as D3
77	No Transposition (C to C) Play tone of 150.99 (LOWER BOUND)	The program should interpret this audio as a note of D3	Note identified as D#3



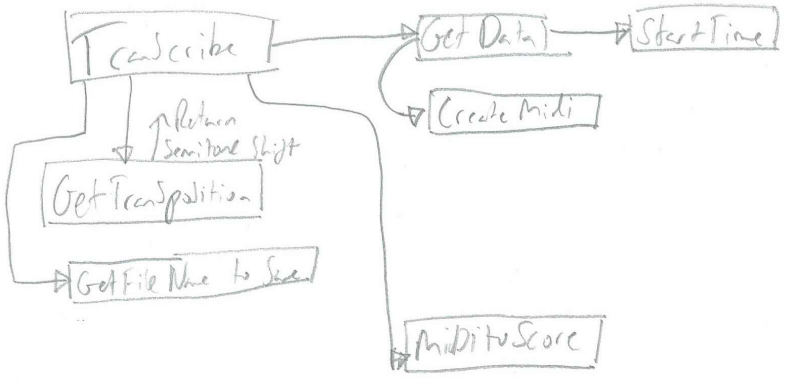
78	No Transposition (C to C) Play tone of 151.39 (UPPER BOUND)	The program should interpret this audio as a note of D3	Note identified as D#3
79	No Transposition (C to C) Play tone of 159.99 (LOWER BOUND)	The program should interpret this audio as a note of D#3/Eb3	Note identified as E3
80	No Transposition (C to C) Play tone of 160.39 (UPPER BOUND)	The program should interpret this audio as a note of D#3/Eb3	Note identified as E3
81	No Transposition (C to C) Play tone of 169.51 (LOWER BOUND)	The program should interpret this audio as a note of E3	Note identified as F3
82	No Transposition (C to C) Play tone of 169.91 (UPPER BOUND)	The program should interpret this audio as a note of E3	Note identified as F3
83	No Transposition (C to C) Play tone of 179.61 (LOWER BOUND)	The program should interpret this audio as a note of F3	Note identified as F#3
84	No Transposition (C to C) Play tone of 180.01 (UPPER BOUND)	The program should interpret this audio as a note of F3	Note identified as F#3
85	No Transposition (C to C) Play tone of 190.3 (LOWER BOUND)	The program should interpret this audio as a note of F#3/Gb3	Note identified as G3
86	No Transposition (C to C) Play tone of 190.7 (UPPER BOUND)	The program should interpret this audio as a note of F#3/Gb3	Note identified as G3
87	No Transposition (C to C) Play tone of 201.62 (LOWER BOUND)	The program should interpret this audio as a note of G3	Note identified as G#3
88	No Transposition (C to C) Play tone of 202.02 (UPPER BOUND)	The program should interpret this audio as a note of G3	Note identified as G#3
89	No Transposition (C to C) Play tone of 213.62 (LOWER BOUND)	The program should interpret this audio as a note of G#3/Ab3	Note identified as A3



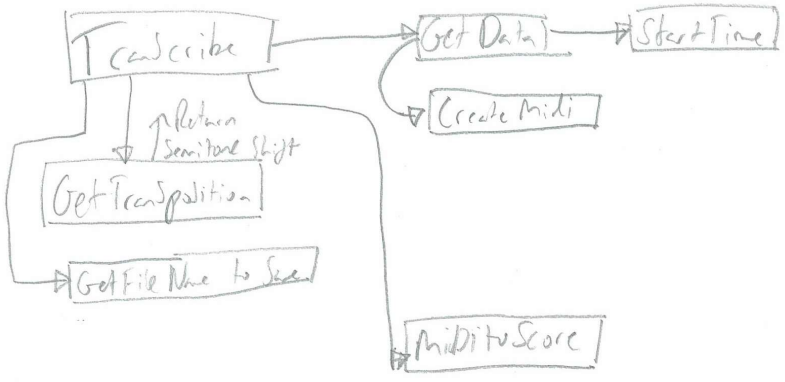
90	No Transposition (C to C) Play tone of 214.02 (UPPER BOUND)	The program should interpret this audio as a note of G#3/Ab3	Note identified as A3
91	No Transposition (C to C) Play tone of 226.34 (LOWER BOUND)	The program should interpret this audio as a note of A3	Note identified as A#3
92	No Transposition (C to C) Play tone of 226.74 (UPPER BOUND)	The program should interpret this audio as a note of A3	Note identified as A#3
93	No Transposition (C to C) Play tone of 239.81 (LOWER BOUND)	The program should interpret this audio as a note of A#3/Bb3	Note identified as B3
94	No Transposition (C to C) Play tone of 240.21 (UPPER BOUND)	The program should interpret this audio as a note of A#3/Bb3	Note identified as B3
95	No Transposition (C to C) Play tone of 254.08 (LOWER BOUND)	The program should interpret this audio as a note of B3	Note identified as C4
96	No Transposition (C to C) Play tone of 254.48 (UPPER BOUND)	The program should interpret this audio as a note of B3	Note identified as C4
97	No Transposition (C to C) Play tone of 269.2 (LOWER BOUND)	The program should interpret this audio as a note of C4	Note identified as C#4
98	No Transposition (C to C) Play tone of 269.6 (UPPER BOUND)	The program should interpret this audio as a note of C4	Note identified as C#4
99	No Transposition (C to C) Play tone of 285.22 (LOWER BOUND)	The program should interpret this audio as a note of C#4/Db4	Note identified as D4
100	No Transposition (C to C) Play tone of 285.62 (UPPER BOUND)	The program should interpret this audio as a note of C#4/Db4	Note identified as D4
101	No Transposition (C to C) Play tone of 302.19 (LOWER BOUND)	The program should interpret this audio as a note of D4	Note identified as D#4



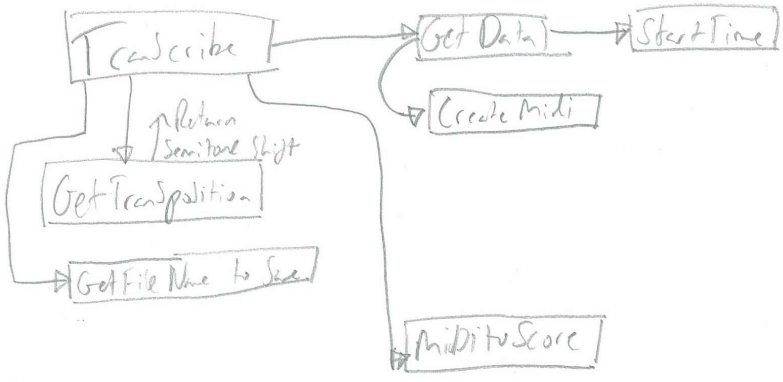
102	No Transposition (C to C) Play tone of 302.59 (UPPER BOUND)	The program should interpret this audio as a note of D4	Note identified as D#4
103	No Transposition (C to C) Play tone of 320.18 (LOWER BOUND)	The program should interpret this audio as a note of D#4/Eb4	Note identified as E4
104	No Transposition (C to C) Play tone of 320.58 (UPPER BOUND)	The program should interpret this audio as a note of D#4/Eb4	Note identified as E4
105	No Transposition (C to C) Play tone of 339.23 (LOWER BOUND)	The program should interpret this audio as a note of E4	Note identified as F4
106	No Transposition (C to C) Play tone of 339.63 (UPPER BOUND)	The program should interpret this audio as a note of E4	Note identified as F4
107	No Transposition (C to C) Play tone of 359.41 (LOWER BOUND)	The program should interpret this audio as a note of F4	Note identified as F#4
108	No Transposition (C to C) Play tone of 359.81 (UPPER BOUND)	The program should interpret this audio as a note of F4	Note identified as F#4
109	No Transposition (C to C) Play tone of 380.8 (LOWER BOUND)	The program should interpret this audio as a note of F#4/Gb4	Note identified as G4
110	No Transposition (C to C) Play tone of 381.2 (UPPER BOUND)	The program should interpret this audio as a note of F#4/Gb4	Note identified as G4
111	No Transposition (C to C) Play tone of 403.45 (LOWER BOUND)	The program should interpret this audio as a note of G4	Note identified as G#4
112	No Transposition (C to C) Play tone of 403.85 (UPPER BOUND)	The program should interpret this audio as a note of G4	Note identified as G#4
113	No Transposition (C to C) Play tone of 427.45 (LOWER BOUND)	The program should interpret this audio as a note of G#4/Ab4	Note identified as A4



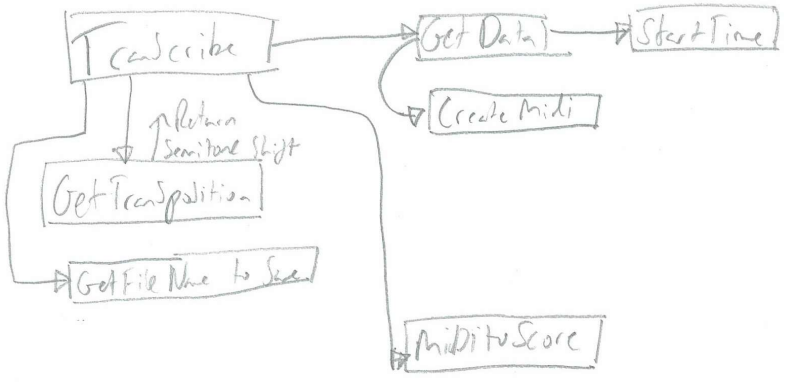
114	No Transposition (C to C) Play tone of 427.85 (UPPER BOUND)	The program should interpret this audio as a note of G#4/Ab4	Note identified as A4
115	No Transposition (C to C) Play tone of 452.88 (LOWER BOUND)	The program should interpret this audio as a note of A4	Note identified as A#4
116	No Transposition (C to C) Play tone of 453.28 (UPPER BOUND)	The program should interpret this audio as a note of A4	Note identified as A#4
117	No Transposition (C to C) Play tone of 479.82 (LOWER BOUND)	The program should interpret this audio as a note of A#4/Bb4	Note identified as B4
118	No Transposition (C to C) Play tone of 480.22 (UPPER BOUND)	The program should interpret this audio as a note of A#4/Bb4	Note identified as B4
119	No Transposition (C to C) Play tone of 508.36 (LOWER BOUND)	The program should interpret this audio as a note of B4	Note identified as C5
120	No Transposition (C to C) Play tone of 508.76 (UPPER BOUND)	The program should interpret this audio as a note of B4	Note identified as C5
121	No Transposition (C to C) Play tone of 538.61 (LOWER BOUND)	The program should interpret this audio as a note of C5	Note identified as C#5
122	No Transposition (C to C) Play tone of 539.01 (UPPER BOUND)	The program should interpret this audio as a note of C5	Note identified as C#5
123	No Transposition (C to C) Play tone of 570.65 (LOWER BOUND)	The program should interpret this audio as a note of C#5/Db5	Note identified as D5
124	No Transposition (C to C) Play tone of 571.05 (UPPER BOUND)	The program should interpret this audio as a note of C#5/Db5	Note identified as D5
125	No Transposition (C to C) Play tone of 604.59 (LOWER BOUND)	The program should interpret this audio as a note of D5	Note identified as D#5



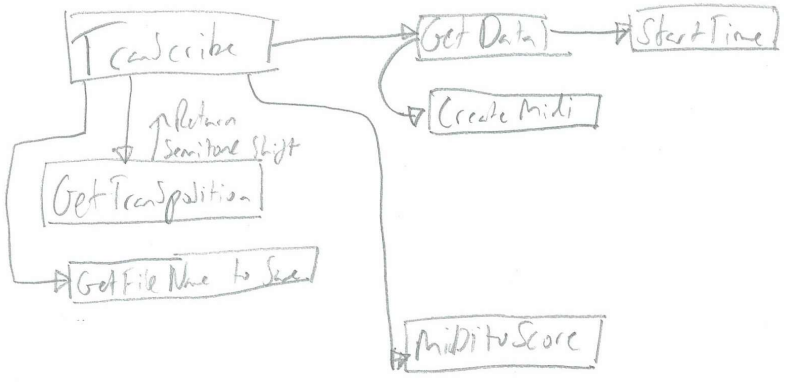
126	No Transposition (C to C) Play tone of 604.99 (UPPER BOUND)	The program should interpret this audio as a note of D5	Note identified as D#5
127	No Transposition (C to C) Play tone of 640.55 (LOWER BOUND)	The program should interpret this audio as a note of D#5/Eb5	Note identified as E5
128	No Transposition (C to C) Play tone of 640.95 (UPPER BOUND)	The program should interpret this audio as a note of D#5/Eb5	Note identified as E5
129	No Transposition (C to C) Play tone of 678.66 (LOWER BOUND)	The program should interpret this audio as a note of E5	Note identified as F5
130	No Transposition (C to C) Play tone of 679.06 (UPPER BOUND)	The program should interpret this audio as a note of E5	Note identified as F5
131	No Transposition (C to C) Play tone of 719.03 (LOWER BOUND)	The program should interpret this audio as a note of F5	Note identified as F5
132	No Transposition (C to C) Play tone of 719.43 (UPPER BOUND)	The program should interpret this audio as a note of F5	Note identified as F#5
133	No Transposition (C to C) Play tone of 761.79 (LOWER BOUND)	The program should interpret this audio as a note of F#5/Gb5	Note identified as G5
134	No Transposition (C to C) Play tone of 762.19 (UPPER BOUND)	The program should interpret this audio as a note of F#5/Gb5	Note identified as G5
135	No Transposition (C to C) Play tone of 807.1 (LOWER BOUND)	The program should interpret this audio as a note of G5	Note identified as G5
136	No Transposition (C to C) Play tone of 807.5 (UPPER BOUND)	The program should interpret this audio as a note of G5	Note identified as G#5
137	No Transposition (C to C) Play tone of 855.11 (LOWER BOUND)	The program should interpret this audio as a note of G#5/Ab5	Note identified as G#5



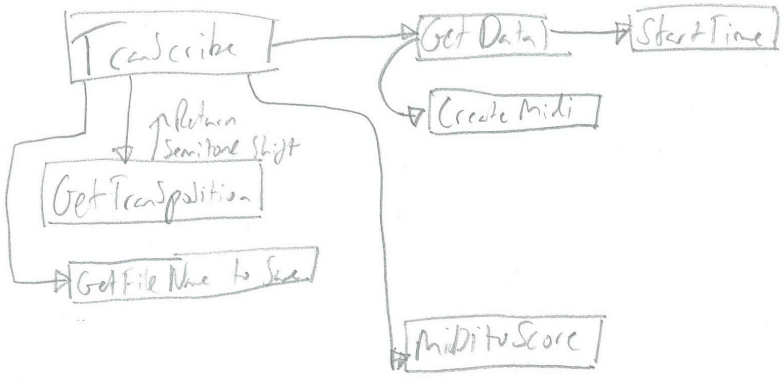
138	No Transposition (C to C) Play tone of 855.51 (UPPER BOUND)	The program should interpret this audio as a note of G#5/Ab5	Note identified as A5
139	No Transposition (C to C) Play tone of 905.96 (LOWER BOUND)	The program should interpret this audio as a note of A5	Note identified as A5
140	No Transposition (C to C) Play tone of 906.36 (UPPER BOUND)	The program should interpret this audio as a note of A5	Note identified as A#5
141	No Transposition (C to C) Play tone of 959.85 (LOWER BOUND)	The program should interpret this audio as a note of A#5/Bb5	Note identified as A#5
142	No Transposition (C to C) Play tone of 960.25 (UPPER BOUND)	The program should interpret this audio as a note of A#5/Bb5	Note identified as B5
143	No Transposition (C to C) Play tone of 1016.93 (LOWER BOUND)	The program should interpret this audio as a note of B5	Note identified as B5
144	No Transposition (C to C) Play tone of 1017.33 (UPPER BOUND)	The program should interpret this audio as a note of B5	Note identified as C6
145	No Transposition (C to C) Play tone of 1077.42 (LOWER BOUND)	The program should interpret this audio as a note of C6	Note identified as C#6
146	No Transposition (C to C) Play tone of 1077.82 (UPPER BOUND)	The program should interpret this audio as a note of C6	Note identified as C#6
147	No Transposition (C to C) Play tone of 1141.5 (LOWER BOUND)	The program should interpret this audio as a note of C#6/Db6	Note identified as D6
148	No Transposition (C to C) Play tone of 1141.9 (UPPER BOUND)	The program should interpret this audio as a note of C#6/Db6	Note identified as D6
149	No Transposition (C to C) Play tone of 1209.39 (LOWER BOUND)	The program should interpret this audio as a note of D6	Note identified as D#6



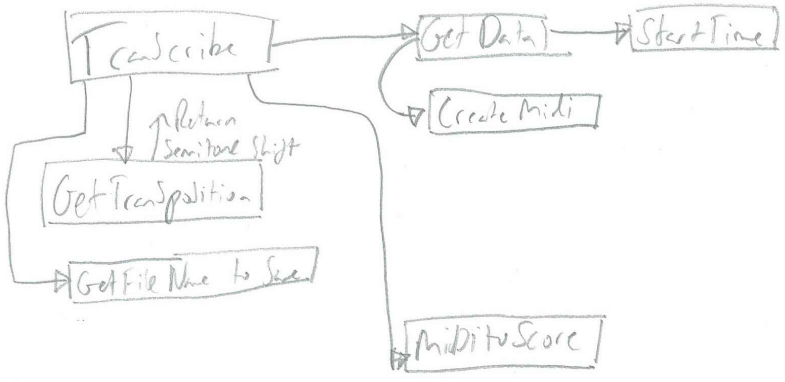
150	No Transposition (C to C) Play tone of 1209.79 (UPPER BOUND)	The program should interpret this audio as a note of D6	Note identified as D#6
151	No Transposition (C to C) Play tone of 1281.31 (LOWER BOUND)	The program should interpret this audio as a note of D#6/Eb6	Note identified as E6
152	No Transposition (C to C) Play tone of 1281.71 (UPPER BOUND)	The program should interpret this audio as a note of D#6/Eb6	Note identified as E6
153	No Transposition (C to C) Play tone of 1357.51 (LOWER BOUND)	The program should interpret this audio as a note of E6	Note identified as F6
154	No Transposition (C to C) Play tone of 1357.91 (UPPER BOUND)	The program should interpret this audio as a note of E6	Note identified as F6
155	No Transposition (C to C) Play tone of 1438.25 (LOWER BOUND)	The program should interpret this audio as a note of F6	Note identified as F#6
156	No Transposition (C to C) Play tone of 1438.65 (UPPER BOUND)	The program should interpret this audio as a note of F6	Note identified as F#6
157	No Transposition (C to C) Play tone of 1523.78 (LOWER BOUND)	The program should interpret this audio as a note of F#6/Gb6	Note identified as G6
158	No Transposition (C to C) Play tone of 1524.18 (UPPER BOUND)	The program should interpret this audio as a note of F#6/Gb6	Note identified as G6
159	No Transposition (C to C) Play tone of 1614.4 (LOWER BOUND)	The program should interpret this audio as a note of G6	Note identified as G#6
160	No Transposition (C to C) Play tone of 1614.8 (UPPER BOUND)	The program should interpret this audio as a note of G6	Note identified as G#6
161	No Transposition (C to C) Play tone of 1710.41 (LOWER BOUND)	The program should interpret this audio as a note of G#6/Ab6	Note identified as A6



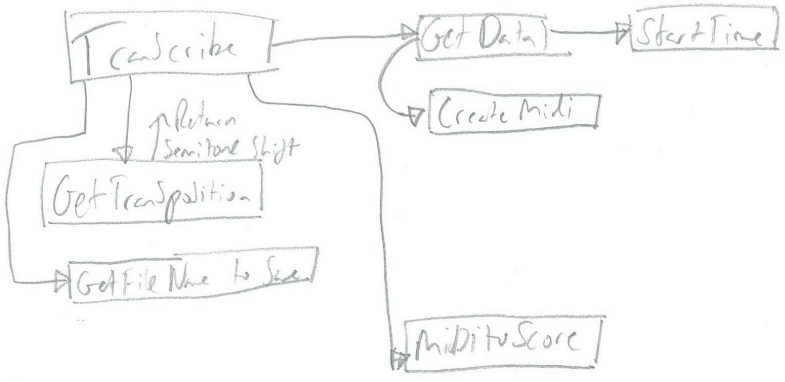
162	No Transposition (C to C) Play tone of 1710.81 (UPPER BOUND)	The program should interpret this audio as a note of G#6/Ab6	Note identified as A6
163	No Transposition (C to C) Play tone of 1812.13 (LOWER BOUND)	The program should interpret this audio as a note of A6	Note identified as A#6
164	No Transposition (C to C) Play tone of 1812.53 (UPPER BOUND)	The program should interpret this audio as a note of A6	Note identified as A#6
165	No Transposition (C to C) Play tone of 1919.9 (LOWER BOUND)	The program should interpret this audio as a note of A#6/Bb6	Note identified as B6
166	No Transposition (C to C) Play tone of 1920.3 (UPPER BOUND)	The program should interpret this audio as a note of A#6/Bb6	Note identified as B6
167	No Transposition (C to C) Play tone of 2034.06 (LOWER BOUND)	The program should interpret this audio as a note of B6	Note identified as C7
168	No Transposition (C to C) Play tone of 2034.46 (UPPER BOUND)	The program should interpret this audio as a note of B6	Note identified as C7
169	No Transposition (C to C) Play tone of 2155.03 (LOWER BOUND)	The program should interpret this audio as a note of C7	Note identified as C#6
170	No Transposition (C to C) Play tone of 2155.43 (UPPER BOUND)	The program should interpret this audio as a note of C7	Note identified as C#6
171	No Transposition (C to C) Play tone of 2283.19 (LOWER BOUND)	The program should interpret this audio as a note of C#7/Db7	Note identified as F#5
172	No Transposition (C to C) Play tone of 2283.59 (UPPER BOUND)	The program should interpret this audio as a note of C#7/Db7	Note identified as F#5
173	No Transposition (C to C) Play tone of 2418.97 (LOWER BOUND)	The program should interpret this audio as a note of D7	Note identified as D#7



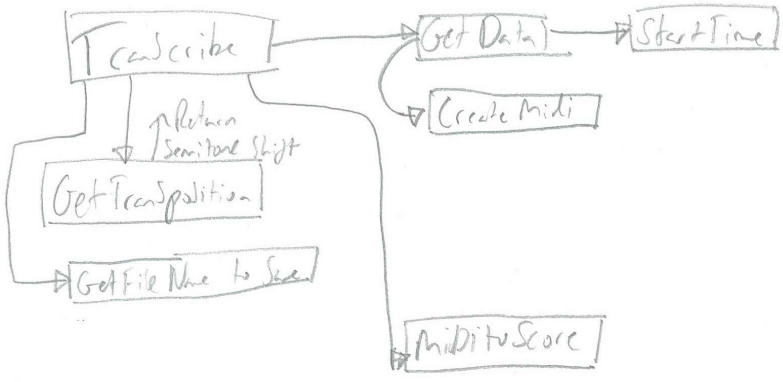
174	No Transposition (C to C) Play tone of 2419.37 (UPPER BOUND)	The program should interpret this audio as a note of D7	Note identified as D#7
175	No Transposition (C to C) Play tone of 2562.82 (LOWER BOUND)	The program should interpret this audio as a note of D#7/Eb7	Note identified as E7
176	No Transposition (C to C) Play tone of 2563.22 (UPPER BOUND)	The program should interpret this audio as a note of D#7/Eb7	Note identified as E7
177	No Transposition (C to C) Play tone of 2715.23 (LOWER BOUND)	The program should interpret this audio as a note of E7	Note identified as F7
178	No Transposition (C to C) Play tone of 2715.63 (UPPER BOUND)	The program should interpret this audio as a note of E7	Note identified as F7
179	No Transposition (C to C) Play tone of 2876.69 (LOWER BOUND)	The program should interpret this audio as a note of F7	Note identified as A#5
180	No Transposition (C to C) Play tone of 2877.09 (UPPER BOUND)	The program should interpret this audio as a note of F7	Note identified as A#5
181	No Transposition (C to C) Play tone of 3047.76 (LOWER BOUND)	The program should interpret this audio as a note of F#7/Gb7	Note identified as G7
182	No Transposition (C to C) Play tone of 3048.16 (UPPER BOUND)	The program should interpret this audio as a note of F#7/Gb7	Note identified as G7
183	No Transposition (C to C) Play tone of 3229.0 (LOWER BOUND)	The program should interpret this audio as a note of G7	Note identified as C6
184	No Transposition (C to C) Play tone of 3229.4 (UPPER BOUND)	The program should interpret this audio as a note of G7	Note identified as C6
185	No Transposition (C to C) Play tone of 3421.02 (LOWER BOUND)	The program should interpret this audio as a note of G#7/Ab7	Note identified as A7



186	No Transposition (C to C) Play tone of 3421.42 (UPPER BOUND)	The program should interpret this audio as a note of G#7/Ab7	Note identified as A7
187	No Transposition (C to C) Play tone of 3624.45 (LOWER BOUND)	The program should interpret this audio as a note of A7	Note identified as A#7
188	No Transposition (C to C) Play tone of 3624.85 (UPPER BOUND)	The program should interpret this audio as a note of A7	Note identified as A#7
189	No Transposition (C to C) Play tone of 3839.99 (LOWER BOUND)	The program should interpret this audio as a note of A#7/Bb7	Note identified as B7
190	No Transposition (C to C) Play tone of 3840.39 (UPPER BOUND)	The program should interpret this audio as a note of A#7/Bb7	Note identified as B7
191	No Transposition (C to C) Play tone of 4068.34 (LOWER BOUND)	The program should interpret this audio as a note of B7	Note identified as C8
192	No Transposition (C to C) Play tone of 4068.74 (UPPER BOUND)	The program should interpret this audio as a note of B7	Note identified as C8
193	No Transposition (C to C) Play tone of 4310.27 (LOWER BOUND)	The program should interpret this audio as a note of C8	Note identified as C#6
194	No Transposition (C to C) Play tone of 4310.67 (UPPER BOUND)	The program should interpret this audio as a note of C8	Note identified as C#6
195	No Transposition (C to C) Play tone of 4566.57 (LOWER BOUND)	The program should interpret this audio as a note of C#8/Db8	Note identified as F#6
196	No Transposition (C to C) Play tone of 4566.97 (UPPER BOUND)	The program should interpret this audio as a note of C#8/Db8	Note identified as F#6
197	No Transposition (C to C) Play tone of 4838.13 (LOWER BOUND)	The program should interpret this audio as a note of D8	Note identified as D#8



198	No Transposition (C to C) Play tone of 4838.53 (UPPER BOUND)	The program should interpret this audio as a note of D8	Note identified as D#8
199	No Transposition (C to C) Play tone of 5125.83 (LOWER BOUND)	The program should interpret this audio as a note of D#8/Eb8	Note identified as C6
200	No Transposition (C to C) Play tone of 5126.23 (UPPER BOUND)	The program should interpret this audio as a note of D#8/Eb8	Note identified as C6
201	No Transposition (C to C) Play tone of 5430.64 (LOWER BOUND)	The program should interpret this audio as a note of E8	Note identified as F8
202	No Transposition (C to C) Play tone of 5431.04 (UPPER BOUND)	The program should interpret this audio as a note of E8	Note identified as F8
203	No Transposition (C to C) Play tone of 5753.58 (LOWER BOUND)	The program should interpret this audio as a note of F8	Note identified as A#6
204	No Transposition (C to C) Play tone of 5753.98 (UPPER BOUND)	The program should interpret this audio as a note of F8	Note identified as A#6
205	No Transposition (C to C) Play tone of 6095.72 (LOWER BOUND)	The program should interpret this audio as a note of F#8/Gb8	Note identified as G7
206	No Transposition (C to C) Play tone of 6096.12 (UPPER BOUND)	The program should interpret this audio as a note of F#8/Gb8	Note identified as G7
207	No Transposition (C to C) Play tone of 6458.21 (LOWER BOUND)	The program should interpret this audio as a note of G8	Note identified as C6
208	No Transposition (C to C) Play tone of 6458.61 (UPPER BOUND)	The program should interpret this audio as a note of G8	Note identified as C6
209	No Transposition (C to C) Play tone of 6842.24 (LOWER BOUND)	The program should interpret this audio as a note of G#8/Ab8	Note identified as F#5

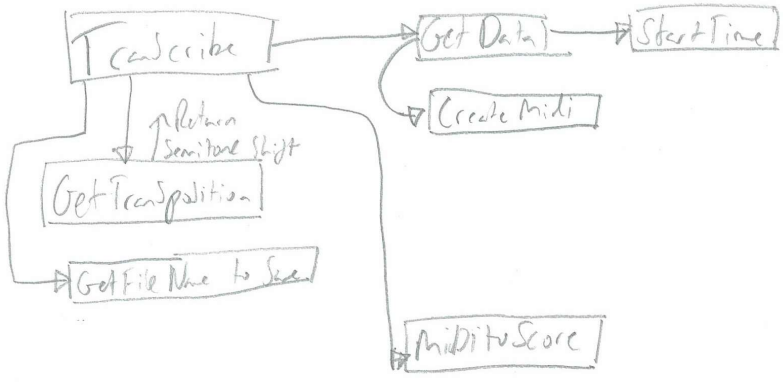


210	No Transposition (C to C) Play tone of 6842.64 (UPPER BOUND)	The program should interpret this audio as a note of G#8/Ab8	Note identified as F#5
211	No Transposition (C to C) Play tone of 7249.11 (LOWER BOUND)	The program should interpret this audio as a note of A8	Note identified as A#8
212	No Transposition (C to C) Play tone of 7249.51 (UPPER BOUND)	The program should interpret this audio as a note of A8	Note identified as A#8
213	No Transposition (C to C) Play tone of 7680.18 (LOWER BOUND)	The program should interpret this audio as a note of A#8/Bb8	Note identified as B7
214	No Transposition (C to C) Play tone of 7680.58 (UPPER BOUND)	The program should interpret this audio as a note of A#8/Bb8	Note identified as B7

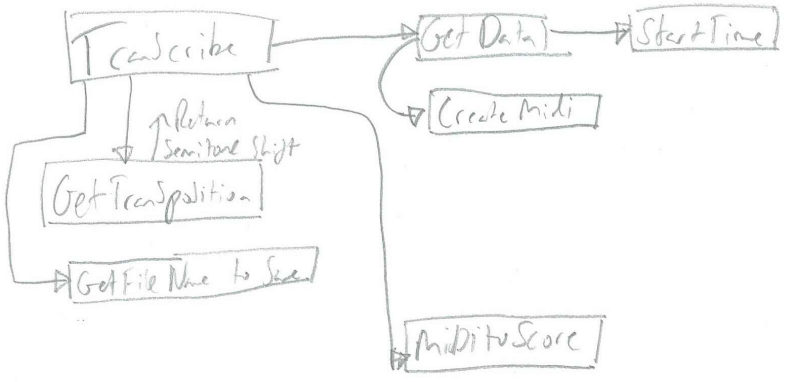
Summary of boundary tests

The Upper and Lower Bound frequencies are decided by ±0.2Hz of the midpoint between any two frequencies, this equates to roughly to the borders of each semitones frequency. In the boundary tests the program fails to identify any notes successfully until around 37.6Hz, where it detects changes in notes but not the note played. At 75.4Hz the program can figure out what the notes are, it is still incorrect but it is correct within one or two semitones and increases its accuracy as the frequencies exist. The program continues to be a semitone out until we get our first successful identification at 719.03Hz, or an F5, this is equivalent to the first octave above middle C. Once again the pattern continues, getting the correct frequency every tone or so. When we reach 2155.03Hz the program starts getting difficulties successfully identifying the octave, indicating ±12 tones to the actual frequency. The same problem occurs with progressively unstable guesses until the end of the test, particularly in the range of 4310.27Hz-7680.58Hz.

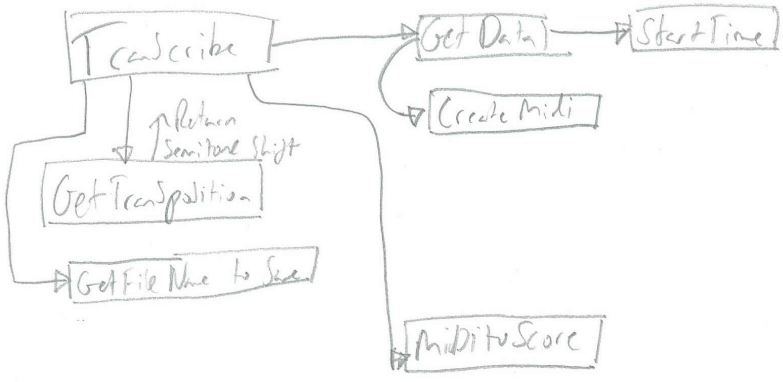
215	No Transposition (C to C) Play tone of 16.35 (EXACT)	The program should interpret this audio as a note of C0	Note Identified as F1
216	No Transposition (C to C) Play tone of 17.32 (EXACT)	The program should interpret this audio as a note of C#0/Db0	Note Identified as F1



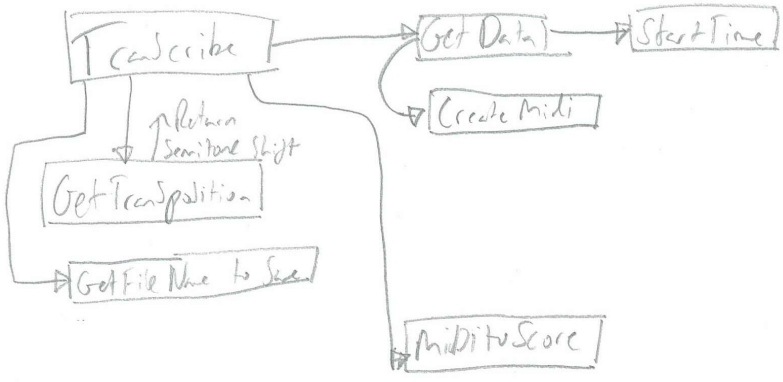
217	No Transposition (C to C) Play tone of 18.35 (EXACT)	The program should interpret this audio as a note of D0	Note Identified as F1
218	No Transposition (C to C) Play tone of 19.45 (EXACT)	The program should interpret this audio as a note of D#0/Eb0	Note Identified as F1
219	No Transposition (C to C) Play tone of 20.6 (EXACT)	The program should interpret this audio as a note of E0	Note Identified as F1
220	No Transposition (C to C) Play tone of 21.83 (EXACT)	The program should interpret this audio as a note of F0	Note Identified as F1
221	No Transposition (C to C) Play tone of 23.12 (EXACT)	The program should interpret this audio as a note of F#0/Gb0	Note Identified as F1
222	No Transposition (C to C) Play tone of 24.5 (EXACT)	The program should interpret this audio as a note of G0	Note Identified as F1
223	No Transposition (C to C) Play tone of 25.96 (EXACT)	The program should interpret this audio as a note of G#0/Ab0	Note Identified as F1
224	No Transposition (C to C) Play tone of 27.5 (EXACT)	The program should interpret this audio as a note of A0	Note Identified as F1
225	No Transposition (C to C) Play tone of 29.14 (EXACT)	The program should interpret this audio as a note of A#0/Bb0	Note Identified as F1
226	No Transposition (C to C) Play tone of 30.87 (EXACT)	The program should interpret this audio as a note of B0	Note Identified as F1
227	No Transposition (C to C) Play tone of 32.7 (EXACT)	The program should interpret this audio as a note of C1	Note Identified as F1
228	No Transposition (C to C) Play tone of 34.65 (EXACT)	The program should interpret this audio as a note of C#1/Db1	Note Identified as F1



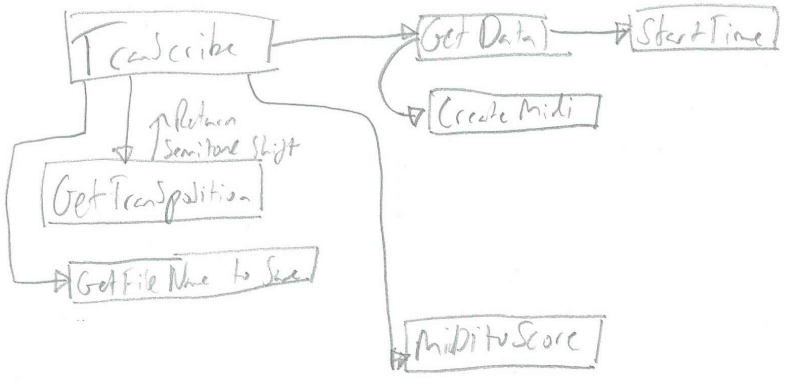
229	No Transposition (C to C) Play tone of 36.71 (EXACT)	The program should interpret this audio as a note of D1	Note Identified as F#1
230	No Transposition (C to C) Play tone of 38.89 (EXACT)	The program should interpret this audio as a note of D#1/Eb1	Note Identified as G#1
231	No Transposition (C to C) Play tone of 41.2 (EXACT)	The program should interpret this audio as a note of E1	Note Identified as A1
232	No Transposition (C to C) Play tone of 43.65 (EXACT)	The program should interpret this audio as a note of F1	Note Identified as A#1
233	No Transposition (C to C) Play tone of 46.25 (EXACT)	The program should interpret this audio as a note of F#1/Gb1	Note Identified as E1
234	No Transposition (C to C) Play tone of 49 (EXACT)	The program should interpret this audio as a note of G1	Note Identified as B1
235	No Transposition (C to C) Play tone of 51.91 (EXACT)	The program should interpret this audio as a note of G#1/Ab1	Note Identified as C2
236	No Transposition (C to C) Play tone of 55 (EXACT)	The program should interpret this audio as a note of A1	Note Identified as C2
237	No Transposition (C to C) Play tone of 58.27 (EXACT)	The program should interpret this audio as a note of A#1/Bb1	Note Identified as C2
238	No Transposition (C to C) Play tone of 61.74 (EXACT)	The program should interpret this audio as a note of B1	Note Identified as C#2
239	No Transposition (C to C) Play tone of 65.41 (EXACT)	The program should interpret this audio as a note of C2	Note Identified as D2
240	No Transposition (C to C) Play tone of 69.3 (EXACT)	The program should interpret this audio as a note of C#2/Db2	Note Identified as D#2



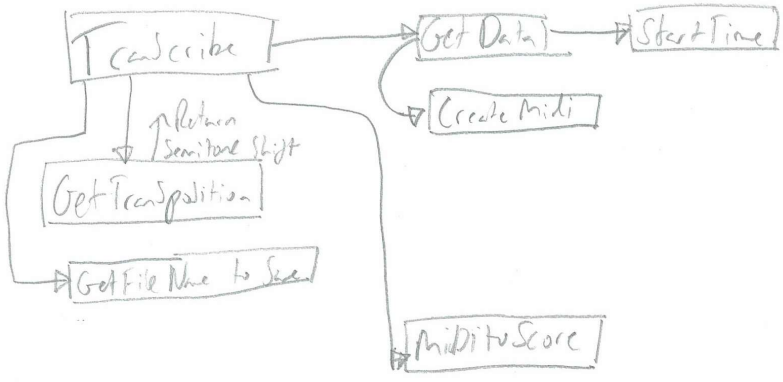
241	No Transposition (C to C) Play tone of 73.42 (EXACT)	The program should interpret this audio as a note of D2	Note Identified as E2
242	No Transposition (C to C) Play tone of 77.78 (EXACT)	The program should interpret this audio as a note of D#2/Eb2	Note Identified as F2
243	No Transposition (C to C) Play tone of 82.41 (EXACT)	The program should interpret this audio as a note of E2	Note Identified as F2
244	No Transposition (C to C) Play tone of 87.31 (EXACT)	The program should interpret this audio as a note of F2	Note Identified as F#2
245	No Transposition (C to C) Play tone of 92.5 (EXACT)	The program should interpret this audio as a note of F#2/Gb2	Note Identified as G2
246	No Transposition (C to C) Play tone of 98 (EXACT)	The program should interpret this audio as a note of G2	Note Identified as G#2
247	No Transposition (C to C) Play tone of 103.83 (EXACT)	The program should interpret this audio as a note of G#2/Ab2	Note Identified as A2
248	No Transposition (C to C) Play tone of 110 (EXACT)	The program should interpret this audio as a note of A2	Note Identified as A#2
249	No Transposition (C to C) Play tone of 116.54 (EXACT)	The program should interpret this audio as a note of A#2/Bb2	Note Identified as B2
250	No Transposition (C to C) Play tone of 123.47 (EXACT)	The program should interpret this audio as a note of B2	Note Identified as C3
251	No Transposition (C to C) Play tone of 130.81 (EXACT)	The program should interpret this audio as a note of C3	Note Identified as C#3
252	No Transposition (C to C) Play tone of 138.59 (EXACT)	The program should interpret this audio as a note of C#3/Db3	Note Identified as C#3



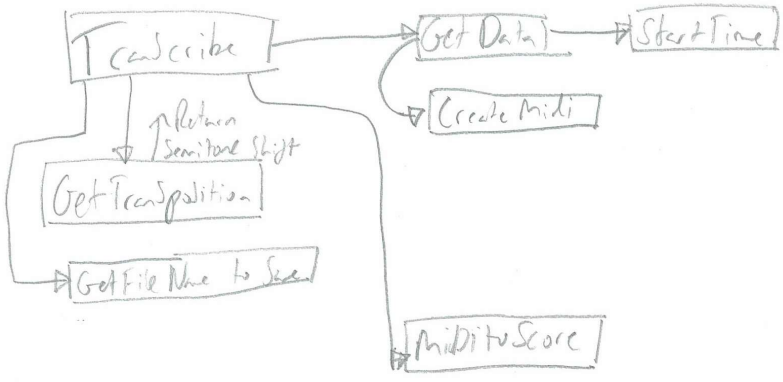
253	No Transposition (C to C) Play tone of 146.83 (EXACT)	The program should interpret this audio as a note of D3	Note Identified as D3
254	No Transposition (C to C) Play tone of 155.56 (EXACT)	The program should interpret this audio as a note of D#3/Eb3	Note Identified as D#3
255	No Transposition (C to C) Play tone of 164.81 (EXACT)	The program should interpret this audio as a note of E3	Note Identified as E3
256	No Transposition (C to C) Play tone of 174.61 (EXACT)	The program should interpret this audio as a note of F3	Note Identified as F3
257	No Transposition (C to C) Play tone of 185 (EXACT)	The program should interpret this audio as a note of F#3/Gb3	Note Identified as F#3
258	No Transposition (C to C) Play tone of 196 (EXACT)	The program should interpret this audio as a note of G3	Note Identified as G3
259	No Transposition (C to C) Play tone of 207.65 (EXACT)	The program should interpret this audio as a note of G#3/Ab3	Note Identified as G#3
260	No Transposition (C to C) Play tone of 220 (EXACT)	The program should interpret this audio as a note of A3	Note Identified as A3
261	No Transposition (C to C) Play tone of 233.08 (EXACT)	The program should interpret this audio as a note of A#3/Bb3	Note Identified as A#3
262	No Transposition (C to C) Play tone of 246.94 (EXACT)	The program should interpret this audio as a note of B3	Note Identified as B3
263	No Transposition (C to C) Play tone of 261.63 (EXACT)	The program should interpret this audio as a note of C4	Note Identified as C4
264	No Transposition (C to C) Play tone of 277.18 (EXACT)	The program should interpret this audio as a note of C#4/Db4	Note Identified as C#4



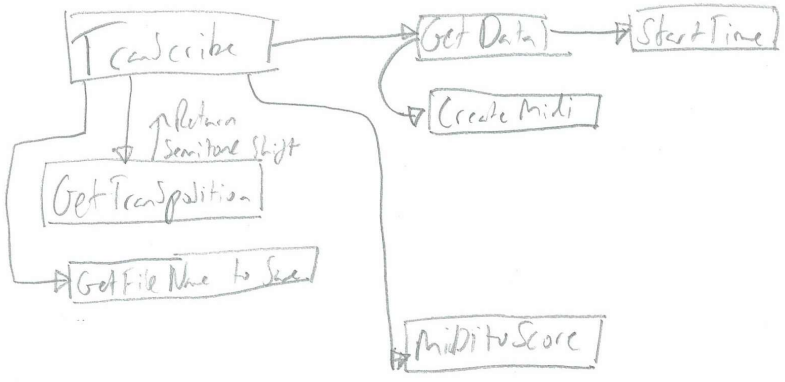
265	No Transposition (C to C) Play tone of 293.66 (EXACT)	The program should interpret this audio as a note of D4	Note Identified as D4
266	No Transposition (C to C) Play tone of 311.13 (EXACT)	The program should interpret this audio as a note of D#4/Eb4	Note Identified as D#4
267	No Transposition (C to C) Play tone of 329.63 (EXACT)	The program should interpret this audio as a note of E4	Note Identified as E3
268	No Transposition (C to C) Play tone of 349.23 (EXACT)	The program should interpret this audio as a note of F4	Note Identified as F4
269	No Transposition (C to C) Play tone of 369.99 (EXACT)	The program should interpret this audio as a note of F#4/Gb4	Note Identified as F#4
270	No Transposition (C to C) Play tone of 392 (EXACT)	The program should interpret this audio as a note of G4	Note Identified as G4
271	No Transposition (C to C) Play tone of 415.3 (EXACT)	The program should interpret this audio as a note of G#4/Ab4	Note Identified as G#4
272	No Transposition (C to C) Play tone of 440 (EXACT)	The program should interpret this audio as a note of A4	Note Identified as A4
273	No Transposition (C to C) Play tone of 466.16 (EXACT)	The program should interpret this audio as a note of A#4/Bb4	Note Identified as A#4
274	No Transposition (C to C) Play tone of 493.88 (EXACT)	The program should interpret this audio as a note of B4	Note Identified as B4
275	No Transposition (C to C) Play tone of 523.25 (EXACT)	The program should interpret this audio as a note of C5	Note Identified as C5
276	No Transposition (C to C) Play tone of 554.37 (EXACT)	The program should interpret this audio as a note of C#5/Db5	Note Identified as C#5



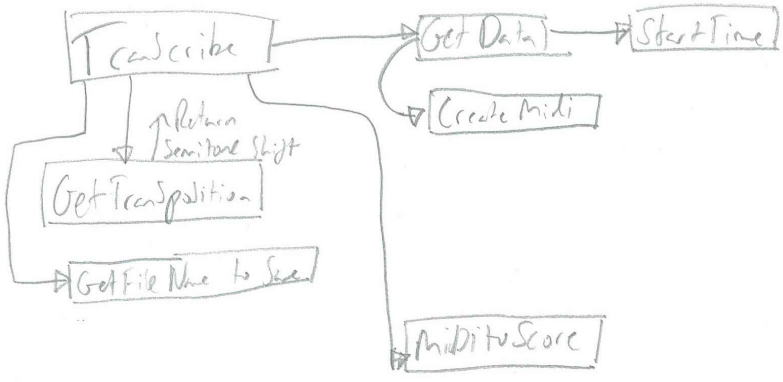
277	No Transposition (C to C) Play tone of 587.33 (EXACT)	The program should interpret this audio as a note of D5	Note Identified as D5
278	No Transposition (C to C) Play tone of 622.25 (EXACT)	The program should interpret this audio as a note of D#5/Eb5	Note Identified as D#5
279	No Transposition (C to C) Play tone of 659.25 (EXACT)	The program should interpret this audio as a note of E5	Note Identified as E5
280	No Transposition (C to C) Play tone of 698.46 (EXACT)	The program should interpret this audio as a note of F5	Note Identified as F5
281	No Transposition (C to C) Play tone of 739.99 (EXACT)	The program should interpret this audio as a note of F#5/Gb5	Note Identified as F#5
282	No Transposition (C to C) Play tone of 783.99 (EXACT)	The program should interpret this audio as a note of G5	Note Identified as G5
283	No Transposition (C to C) Play tone of 830.61 (EXACT)	The program should interpret this audio as a note of G#5/Ab5	Note Identified as G#5
284	No Transposition (C to C) Play tone of 880 (EXACT)	The program should interpret this audio as a note of A5	Note Identified as A5
285	No Transposition (C to C) Play tone of 932.33 (EXACT)	The program should interpret this audio as a note of A#5/Bb5	Note Identified as A#5
286	No Transposition (C to C) Play tone of 987.77 (EXACT)	The program should interpret this audio as a note of B5	Note Identified as B5
287	No Transposition (C to C) Play tone of 1046.5 (EXACT)	The program should interpret this audio as a note of C6	Note Identified as C6
288	No Transposition (C to C) Play tone of 1108.73 (EXACT)	The program should interpret this audio as a note of C#6/Db6	Note Identified as C#6



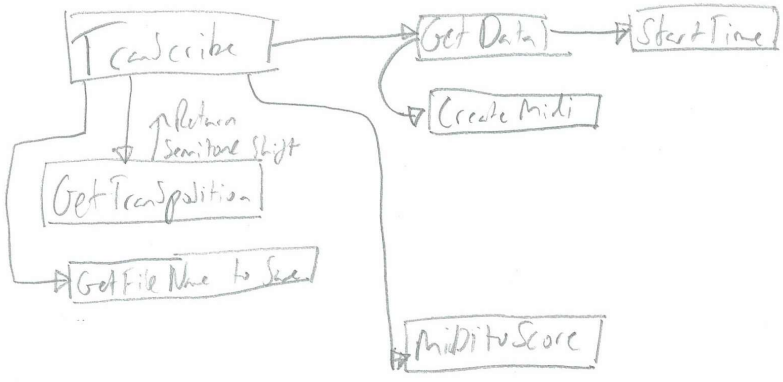
289	No Transposition (C to C) Play tone of 1174.66 (EXACT)	The program should interpret this audio as a note of D6	Note Identified as D6
290	No Transposition (C to C) Play tone of 1244.51 (EXACT)	The program should interpret this audio as a note of D#6/Eb6	Note Identified as D#6
291	No Transposition (C to C) Play tone of 1318.51 (EXACT)	The program should interpret this audio as a note of E6	Note Identified as E6
292	No Transposition (C to C) Play tone of 1396.91 (EXACT)	The program should interpret this audio as a note of F6	Note Identified as F6
293	No Transposition (C to C) Play tone of 1479.98 (EXACT)	The program should interpret this audio as a note of F#6/Gb6	Note Identified as F#6
294	No Transposition (C to C) Play tone of 1567.98 (EXACT)	The program should interpret this audio as a note of G6	Note Identified as G6
295	No Transposition (C to C) Play tone of 1661.22 (EXACT)	The program should interpret this audio as a note of G#6/Ab6	Note Identified as G#6
296	No Transposition (C to C) Play tone of 1760 (EXACT)	The program should interpret this audio as a note of A6	Note Identified as A6
297	No Transposition (C to C) Play tone of 1864.66 (EXACT)	The program should interpret this audio as a note of A#6/Bb6	Note Identified as A#6
298	No Transposition (C to C) Play tone of 1975.53 (EXACT)	The program should interpret this audio as a note of B6	Note Identified as B6
299	No Transposition (C to C) Play tone of 2093 (EXACT)	The program should interpret this audio as a note of C7	Note Identified as C7
300	No Transposition (C to C) Play tone of 2217.46 (EXACT)	The program should interpret this audio as a note of C#7/Db7	Note Identified as C#7



301	No Transposition (C to C) Play tone of 2349.32 (EXACT)	The program should interpret this audio as a note of D7	Note Identified as D7
302	No Transposition (C to C) Play tone of 2489.02 (EXACT)	The program should interpret this audio as a note of D#7/Eb7	Note Identified as D#7
303	No Transposition (C to C) Play tone of 2637.02 (EXACT)	The program should interpret this audio as a note of E7	Note Identified as A5
304	No Transposition (C to C) Play tone of 2793.83 (EXACT)	The program should interpret this audio as a note of F7	Note Identified as F7
305	No Transposition (C to C) Play tone of 2959.96 (EXACT)	The program should interpret this audio as a note of F#7/Gb7	Note Identified as F#7
306	No Transposition (C to C) Play tone of 3135.96 (EXACT)	The program should interpret this audio as a note of G7	Note Identified as G7
307	No Transposition (C to C) Play tone of 3322.44 (EXACT)	The program should interpret this audio as a note of G#7/Ab7	Note Identified as G#5
308	No Transposition (C to C) Play tone of 3520 (EXACT)	The program should interpret this audio as a note of A7	Note Identified as A7
309	No Transposition (C to C) Play tone of 3729.31 (EXACT)	The program should interpret this audio as a note of A#7/Bb7	Note Identified as A#7
310	No Transposition (C to C) Play tone of 3951.07 (EXACT)	The program should interpret this audio as a note of B7	Note Identified as B7
311	No Transposition (C to C) Play tone of 4186.01 (EXACT)	The program should interpret this audio as a note of C8	Note Identified as C8
312	No Transposition (C to C) Play tone of 4434.92 (EXACT)	The program should interpret this audio as a note of C#8/Db8	Note Identified as C#8



313	No Transposition (C to C) Play tone of 4698.63 (EXACT)	The program should interpret this audio as a note of D8	Note Identified as A#5
314	No Transposition (C to C) Play tone of 4978.03 (EXACT)	The program should interpret this audio as a note of D#8/Eb8	Note Identified as D#8
315	No Transposition (C to C) Play tone of 5274.04 (EXACT)	The program should interpret this audio as a note of E8	Note Identified as A6
316	No Transposition (C to C) Play tone of 5587.65 (EXACT)	The program should interpret this audio as a note of F8	Note Identified as F8
317	No Transposition (C to C) Play tone of 5919.91 (EXACT)	The program should interpret this audio as a note of F#8/Gb8	Note Identified as F#8
318	No Transposition (C to C) Play tone of 6271.93 (EXACT)	The program should interpret this audio as a note of G8	Note Identified as G8
319	No Transposition (C to C) Play tone of 6644.88 (EXACT)	The program should interpret this audio as a note of G#8/Ab8	Note Identified as C#7
320	No Transposition (C to C) Play tone of 7040 (EXACT)	The program should interpret this audio as a note of A8	Note Identified as A7
321	No Transposition (C to C) Play tone of 7458.62 (EXACT)	The program should interpret this audio as a note of A#8/Bb8	Note Identified as A#8
322	No Transposition (C to C) Play tone of 7902.13 (EXACT)	The program should interpret this audio as a note of B8	Note Identified as E5



4097 Oscar Lindenbaum 62113 Wheatley Park School

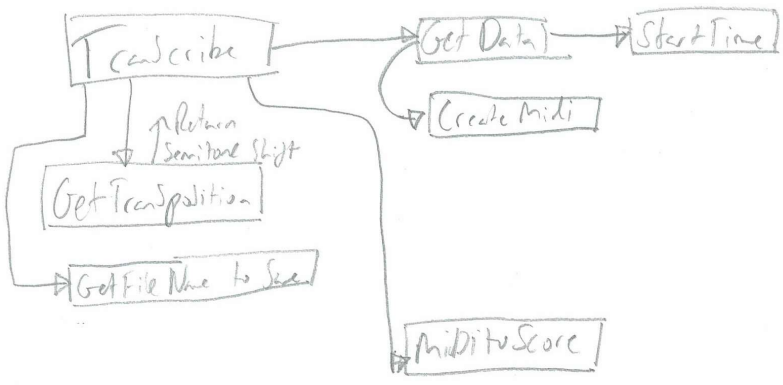
Computer Science NEA - NoteSwitch

INSTRUMENT	WRITTEN RANGE (C4=middle C)	INSTRUMENT	WRITTEN RANGE (C4=middle C)
Piccolo	D4-C7	Xylophone	F3-C7
Flute(in C) orQuerfloete	C4-D7	Marimba	(C2 to A2)-C7
Alto Flute(in G)	C4-C7	Orchestra Bellsalso Glockenspiel	G3-C6
Oboe	Bb3-A6	Vibraphone	F3-F6
Oboe d'amore	Bb3-E6	Chimes	C4-F5
English Horn	B3-G6	Guitar	E3-E6
Heckelphone /Bass Oboe	A3-G6	Harp	Cb1-F#7
Clarinets (Bb-Eb-A)	E3-C7	Piano	A0-C8
Basset Horn	C3-G6	Celesta	C3-C7
Bass Clarinet in Bb	Eb3(or C3)-G6	Harpsichord	F1-F6
Bassoon	Bb1-Eb5	Harmonium	F1-F6
Contrabassoon (Sarrusophone)	Bb1-Bb4	Pipe Organ console	C2-C7 (on tracker organs) Pedals C2-G4 (F4 on German organs)
Saxophones	Bb3-G6	Violin	G3-A7
Horn in F (double horn)	F#2-C6	Viola	C3-E6
Tuben, Wagner tubas (double tuben shown here	Bb: C3-G5 F: F2-D5	Cello	C2-C6
		Double Bass	C2-C5
Trumpet	F#3-D6		
Piccolo Trumpet	F#3-G5	**Accuracy From Keys on Piano**	**65%**
Alto Trombone	A2-G5		
Trombone (Tenor Trombone) Trombone (no valve, straight)	E2-F5	**Accuracy From All Results**	**58%**
Bass Trombone	Bb1-Bb4		
Contrabass Trombone	Ab0-C5	**Average Range of instruments**	**A#2-B5**
Tuba	D1-F4	**Accuracy In Range**	**90%**
Tenor tuba, Euphonium	Bb1-Bb4 in bass		

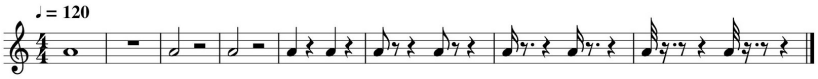
Rhythm Detection

To test the accuracy of the rhythm notation I will create a recording of all the different note lengths and then compare them to the score output.

Audio 1
Expected

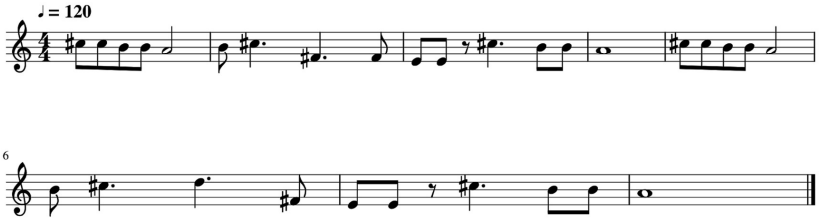


Rhythm Test Expected Output



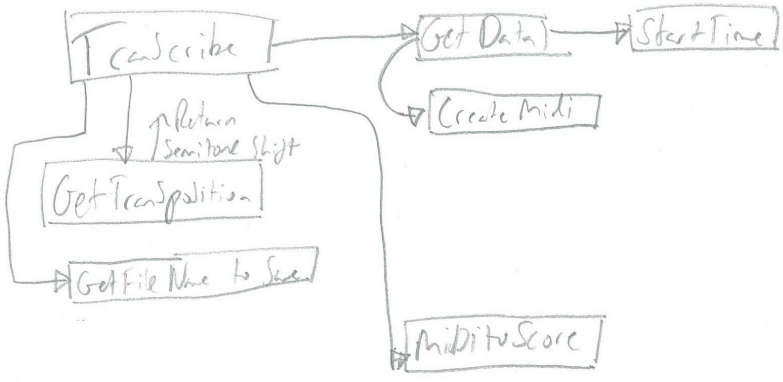
Audio 2

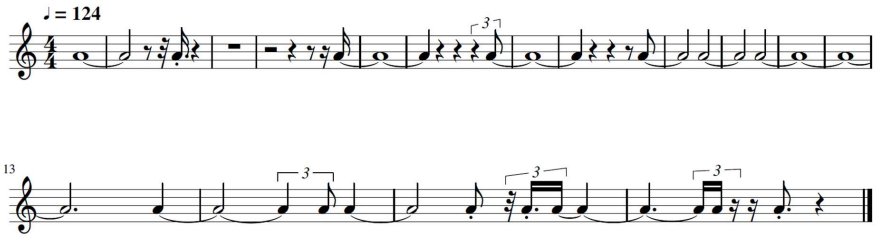
Rhythm Test 2 Expected Output



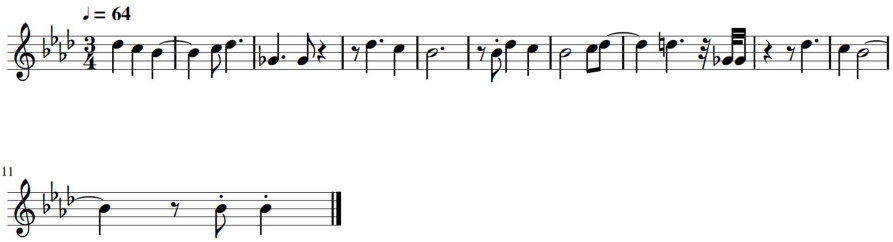
Expected

Rhythm Test 1 Actual Output





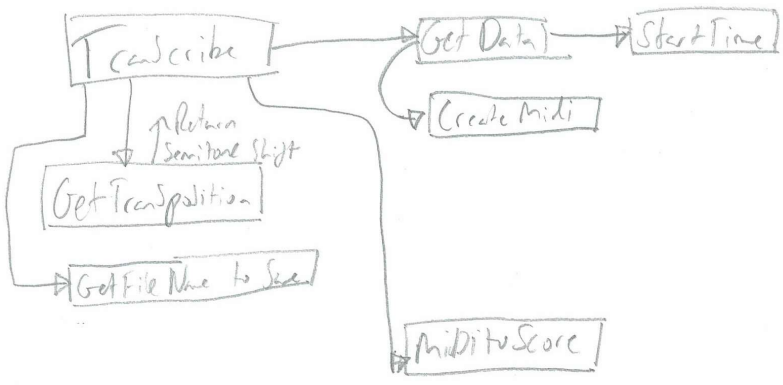
Rhythm Test 2 Actual Output



In both rhythm tests the program successfully identifies the correct note but it displays it as a flat for example a Db instead of a C# in test 2 bar one. This is still correct as it is the same with a different key signature but can sometimes make the music harder to read for the user. Also prominent in both tests is the different length notation, this will be because of either the time signature or the tempo which is selected by the program will offset the rhythms. While still being technically correct this can result in the scenario where it is harder for the user to read. Overall the program passes this test as all the information is correct, in later versions there could be an option for the user to manually input the tempo or time signature but this would defy the point of the program which is to make it easier for students who lack music knowledge to transcribe the solo.

Error Patches

Below are the errors found during the testing phase, each is supplied with the changes to the code and a video of the patch working.



TRC02, TRC04

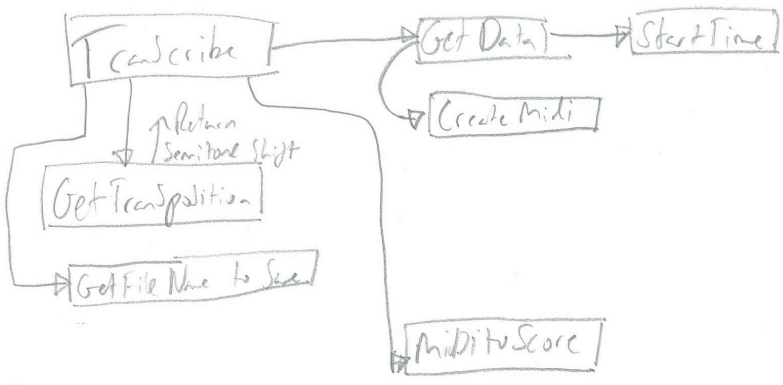
Video Proof https://drive.google.com/file/d/1c62IVMP7zOY-iMqnZPlkgxkRCf6fPfXN/view?usp=sharing

```
        ##//changes
        ##FIX TRC02
        self.startKeyCombo.currentIndexChanged.connect(self.startComboToDial)
        ##FIX TRC04
        self.endKeyCombo.currentIndexChanged.connect(self.endComboToDial)

        QtCore.QMetaObject.connectSlotsByName(MainWindow)

        self.thread = None ##add
        self.worker = None ##add

        self.startButton.clicked.connect(self.start_loop)  ##add

    @pyqtSlot()
    def startComboToDial(self):
            self.startKeyDial.setValue(self.startKeyCombo.currentIndex())
    @pyqtSlot()
    def endComboToDial(self):
            self.endKeyDial.setValue(self.endKeyCombo.currentIndex())
```

TRS04

Video Proof
https://drive.google.com/file/d/1KJvSa48BWiXy3u80EirSezK48KmD8uRY/view?usp=sharing

```
    ##FIX TRS04
    self.endKeyCombo.currentIndexChanged['int'].connect(self.endKeyDial.setValue)
```

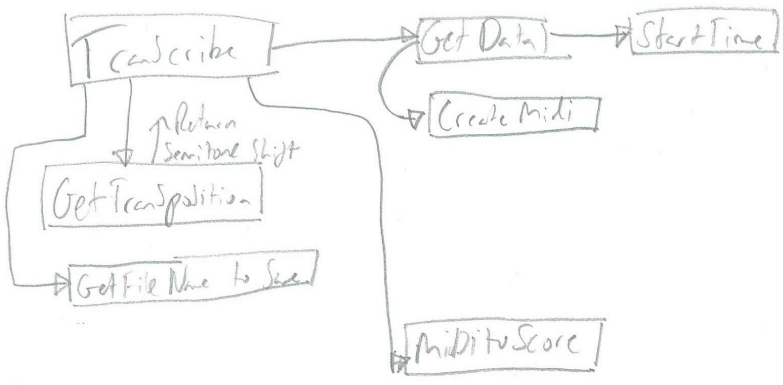


4097 Oscar Lindenbaum 62113 Wheatley Park School

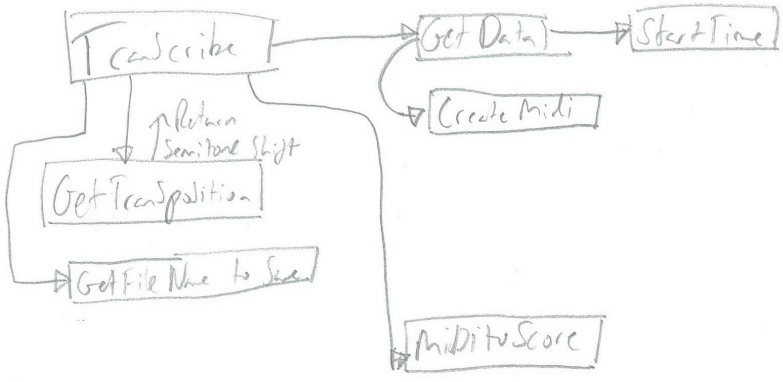
Computer Science NEA - NoteSwitch

TRC05

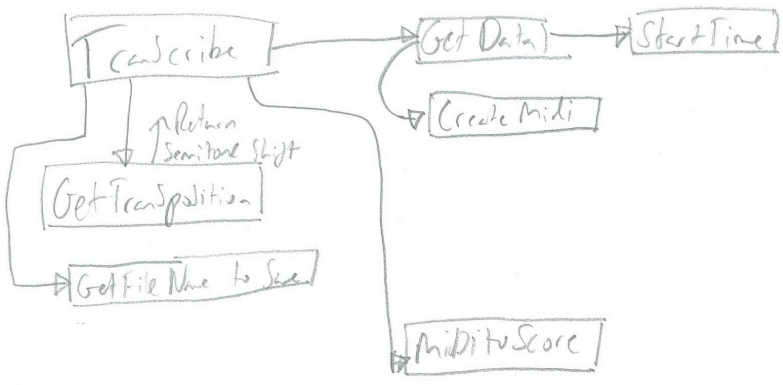
Video Proof
https://drive.google.com/file/d/14dtbiNCDeBRxBBWWUa4_YHPHmUFJsNAv/view?usp=sharing



In this patch the reason the timer is not starting is because the code is being executed on a

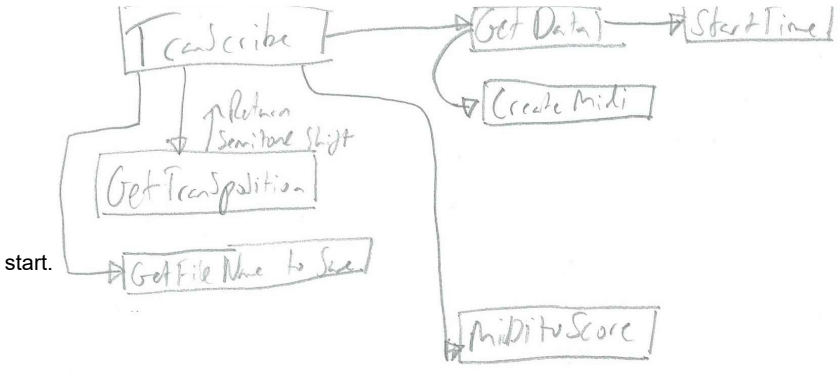


remote machine which has no audio devices attached, this means that the timer will not



```
@pyqtSlot()
def transcribeStartButton_clicked():
    if calculateTransposition(scribeUI,keys)[0]:
            if scribeUI.userNameTextBox.text() != "":
                    if scribeUI.fileNameTextBox.text() != "":
                            scribeUI.startButton.setEnabled(False)
                            scribeUI.stopButton.setEnabled(True)
                            scribeUI.resetButton.setEnabled(False)
                            scribeUI.startKeyDial.setEnabled(False)
                            scribeUI.endKeyDial.setEnabled(False)
                            scribeUI.startKeyCombo.setEnabled(False)
                            scribeUI.endKeyCombo.setEnabled(False)
                            scribeUI.fileNameTextBox.setEnabled(False)
                            scribeUI.userNameTextBox.setEnabled(False)
                            ##FIX TRC05
                            scribeUI.startPossible = True
                            print ("scribeUI.startPossible : {}".format(scribeUI.startPossible))
                            scribeUI.start_loop()
                    else:
                            raiseError("File Name Required")
                            scribeUI.startPossible == False
                            print ("scribeUI.startPossible : {}".format(scribeUI.startPossible))
            else:
                    if scribeUI.fileNameTextBox.text() =="":
                            raiseError("File Name Required")
                            scribeUI.startPossible == False
                            print ("scribeUI.startPossible : {}".format(scribeUI.startPossible))
                    else:
                            raiseError("Username Required")
                            scribeUI.startPossible == False
                            print ("scribeUI.startPossible : {}".format(scribeUI.startPossible))
```

```
    def start_loop(self):
            self.thread = QThread()  # a new thread to run our background tasks in
            self.worker = Worker()  # a new worker to perform those tasks
            self.worker.moveToThread(self.thread)  # move the worker into the thread, do this

            self.thread.started.connect(self.worker.work)  # begin our worker object's loop wh
            self.stopButton.clicked.connect(self.stop_loop)  # stop the loop on the stop butto
            self.worker.finished.connect(self.loop_finished)  # do something in the gui when t
            self.worker.finished.connect(self.thread.quit)  # tell the thread it's time to sto
            self.worker.finished.connect(self.worker.deleteLater)  # have worker mark itself f
            self.thread.finished.connect(self.thread.deleteLater)  # have thread mark itself f
            try:
                    idx = open("audioSettingsFile.txt","r").readlines()[1].split(",")[1]
                    if idx == "":
                            raiseError("Please set Audio Device in settings")
                    elif self.startPossible==True: ##FIX TRC05
                            print ("LOOP SHOULD START")
                            scribeUI.TimerStart()
                            self.thread.start()

                    else:
                            raiseError("Inputs not successful, please set all inputs")
                            print ("self.startPossible : {}".format(self.startPossible))
            except FileNotFoundError:
                    raiseError("Please open settings")
#FIX TRC05
    def __init__(self):
            self.startPossible = False
```



start.

End User Testing

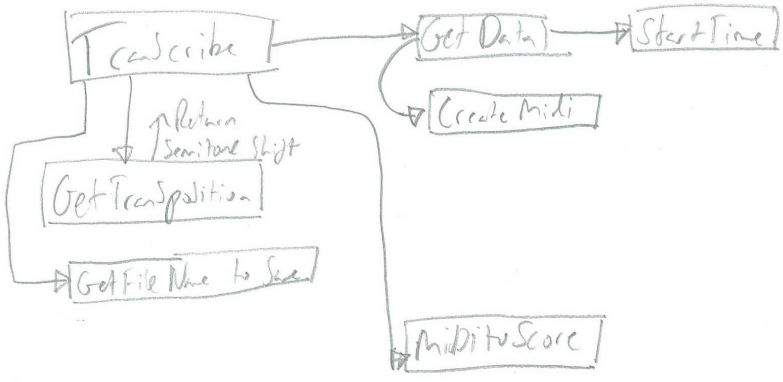
In order to test my application in a real life scenario I invited 3 students, including some who aren't music students, who had answered the initial form to participate in a hands on test. During the test I explained the application in more detail to them and how to use it, for all of the tests no errors occurred and the user was happy how it worked. I then got them to fill in a form (see appendix-viii for a blank copy), the results are shown in appendix-ix. All the users were happy with app layout and said it was simple and easy to navigate and use. Each user also said they could see a use for it in their current life, for "creating scores of their musical ideas with ease" or "transposing music for a band they're in".

As the users were happy with how to use the application they all said it would encourage them to create their own compositions as it would be easier and quicker as it allows you to 'plug' in your instrument which doesn't have any electrical inputs.

Two users commented on additional features, one of which being adding in multi-line scores, this was optimal objective 4.3.2. This reinforces the use for it in future deployments and could be easily implemented. The other additional feature mentioned extra images as to what the application is doing, this could be showing the live audio wave as the program is identifying the note, much like the last addition I could easily implement this using matplotlib in another window.

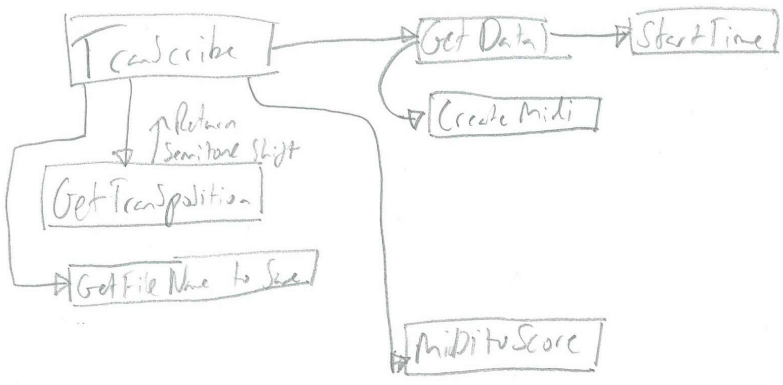
Finally all users mentioned how useful it could be for them, especially with general use.

Videos To Use
https://drive.google.com/file/d/1Q0tDk1SbCBsi-4aWDLSjAHNqwPkyykO4/view?usp=sharing
https://drive.google.com/file/d/1zdX2GIyXRSZKVT8o9vn-BIHhcH7-sWgP/view?usp=sharing
https://drive.google.com/file/d/1jaivB4vWQGCZxow7LAcoBw5S8TEiNPFN/view?usp=sharing



Evaluation

Appendices



Appendix i

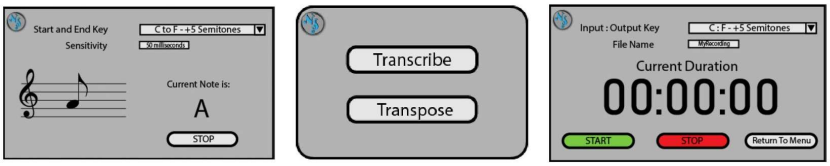
NoteSwitch Research
- Oscar Lindenbaum

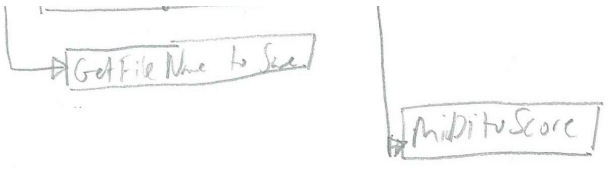
Currently, it is complicated to either transpose or identify the notes being played on any musical instrument without years of learning by ear and perfect pitch. As a musician myself I have found on multiple occasions where either I create a melody or tune and play it but when I want to switch to my other saxophone I would have to transpose all of the notes and remember where they all are on both instruments or pass my sheet music to another player and because their instrument is in a different key so therefore the notes are different. What the application would do was make it easier to switch between instruments with either written music or single notes without a lot of thinking and working out the notes one by one.

The Application

The separate functions that the app would do to solve the problem would be:

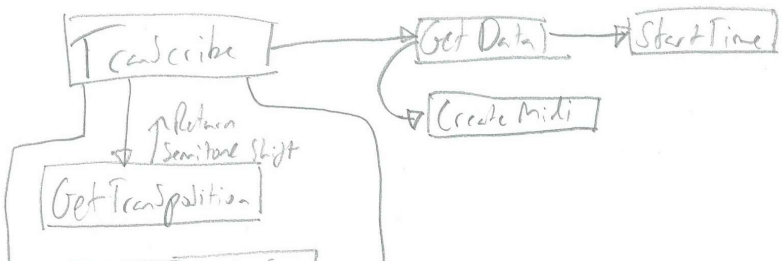
- ➔ Identify Single Notes being played by the given starting instrument and displaying their transposed counterpart.
- ➔ Identify multiple notes being played into the application and notate the rhythms and the different transposed notes onto a score. This file would then be outputted for the user.
  - ◆ If possible an option the app could have is that it will output both the midi file used to create the score file and the score file itself,what this would allow the user to put the recorded piece into another 3rd party software such as MuseScore which would then allow corrections or alterations to be made to the sheet music. What this would additionally do would be to record multiple tracks for example for an arrangement produced for a larger band.
- ➔ One additional feature which would not be a part of my essential solution would be to be able to tune your instrument with it. This would be done by analysing the pitch and deciding the 'level' of accuracy for the tuning which is needed to be attained.





4097 Oscar Lindenbaum 62113 Wheatley Park School

Computer Science NEA - NoteSwitch



End User Questions **COMMENTS IN BOLD**

Name:

Date:

- *<u>If applies:</u> What do you find are the difficulties of playing multiple instruments?*

- *What do you feel are the most important things to get correct when notating music?*

Appendix ii

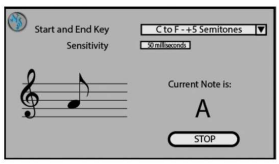
NoteSwitch Research
- Oscar Lindenbaum

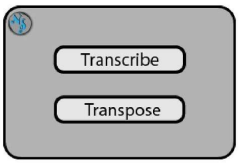
Currently, it is complicated to either transpose or identify the notes being played on any musical instrument without years of learning by ear and perfect pitch. As a musician myself I have found on multiple occasions where either I create a melody or tune and play it but when I want to switch to my other saxophone I would have to transpose all of the notes and remember where they all are on both instruments or pass my sheet music to another player and because their instrument is in a different key so therefore the notes are different. What the application would do was make it easier to switch between instruments with either written music or single notes without a lot of thinking and working out the notes one by one.

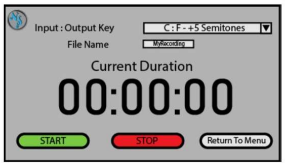
The Application

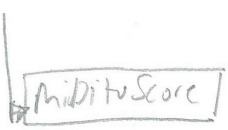
The separate functions that the app would do to solve the problem would be:

- ➔ Identify Single Notes being played by the given starting instrument and displaying their transposed counterpart.
- ➔ Identify multiple notes being played into the application and notate the rhythms and the different transposed notes onto a score. This file would then be outputted for the user.
    - ◆ If possible an option the app could have is that it will output both the midi file used to create the score file and the score file itself,what this would allow the user to put the recorded piece into another 3rd party software such as MuseScore which would then allow corrections or alterations to be made to the sheet music. What this would additionally do would be to record multiple tracks for example for an arrangement produced for a larger band.
- ➔ One additional feature which would not be a part of my essential solution would be to be able to tune your instrument with it. This would be done by analysing the pitch and deciding the 'level' of accuracy for the tuning which is needed to be attained.



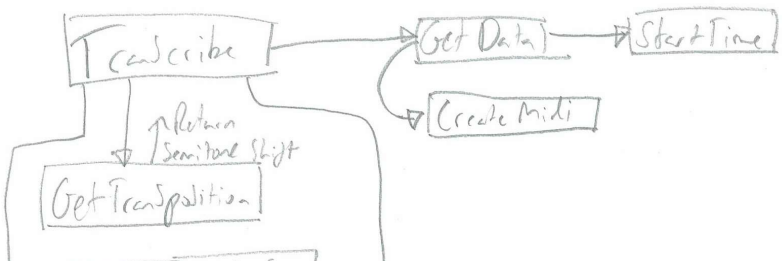






4097 Oscar Lindenbaum 62113 Wheatley Park School

Computer Science NEA - NoteSwitch



Client Questions **COMMENTS IN BOLD**

Name: Michael Ahmad

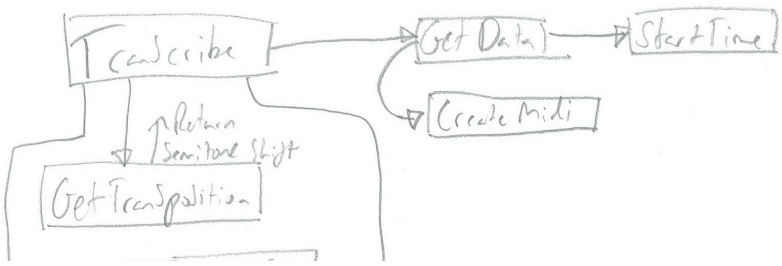
Date:

- *On average how many students per class play multiple instruments?*

- *What do you feel are the most important things to get correct when notating music?*

4097 Oscar Lindenbaum 62113 Wheatley Park School

Computer Science NEA - NoteSwitch

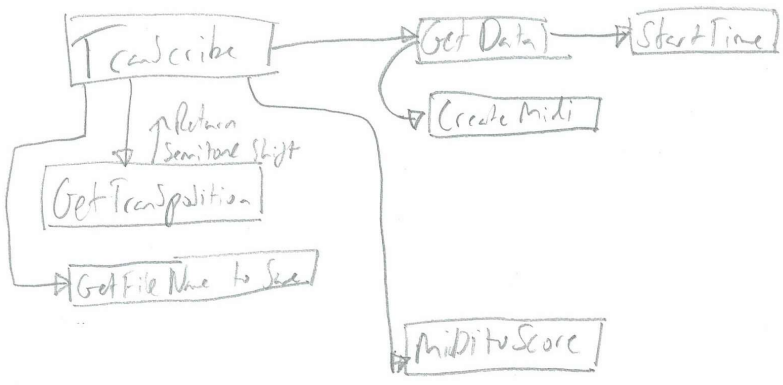


- *What other forms of data would be useful?*

- How will the program benefit students in lessons?

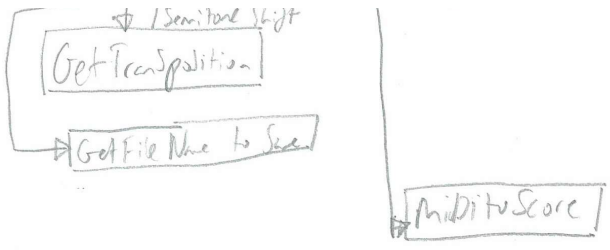
- Should the students be able to upload and share their work with each other and yourself?

- How explicit should the music theory behind application be to the user?

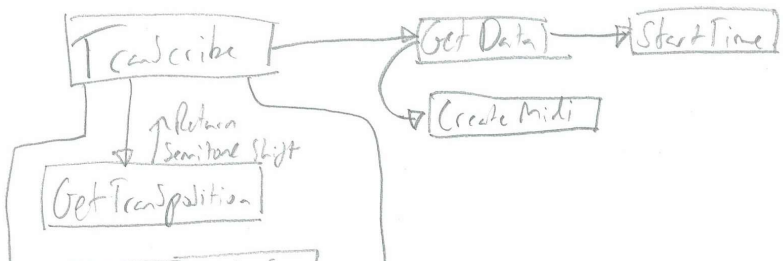

Transcribe
Get Data
Start Time
Create Midi
Return Semitone Shift
Get Transposition
Get File Name to Scan
MidiToScore

Appendix iii

C0
C#0/Db0
D0
D#0/Eb0
E0
F0
F#0/Gb0
G0
G#0/Ab0
A0
A#0/Bb0
B0
C1
C#1/Db1
D1
D#1/Eb1
E1
F1
F#1/Gb1
G1
G#1/Ab1
A1
A#1/Bb1
B1
C2
C#2/Db2
D2
D#2/Eb2
E2
F2
F#2/Gb2
G2
G#2/Ab2
A2
A#2/Bb2
B2
C3
C#3/Db3
D3
D#3/Eb3
E3
F3
F#3/Gb3



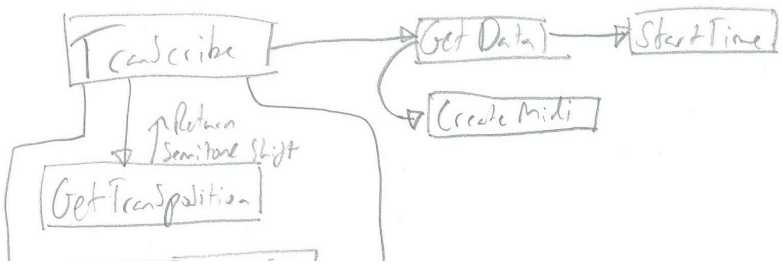
Computer Science NEA - NoteSwitch



G3
G#3/Ab3
A3
A#3/Bb3
B3
C4
C#4/Db4
D4
D#4/Eb4
E4
F4
F#4/Gb4
G4
G#4/Ab4
A4

4097 Oscar Lindenbaum 62113 Wheatley Park School

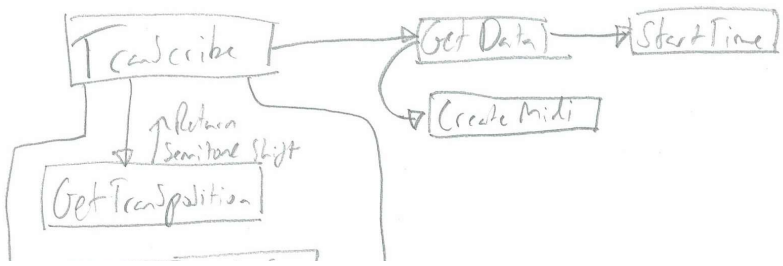
Computer Science NEA - NoteSwitch



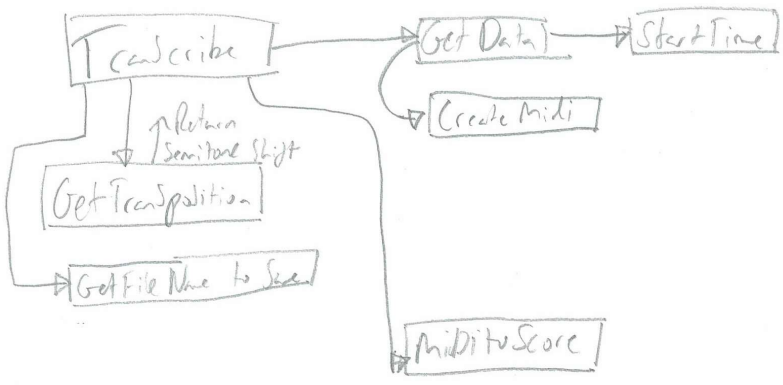
D7
D#7/Eb7
E7
F7
F#7/Gb7
G7
G#7/Ab7
A7
A#7/Bb7
B7
C8
C#8/Db8
D8
D#8/Eb8
E8

Appendix iv

196
207.65
220
233.08
246.94
261.63
277.18
293.66
311.13
329.63
349.23
369.99
392
415.3
440
466.16
493.88
523.25
554.37
587.33
622.25
659.25
698.46
739.99
783.99
830.61
880
932.33
987.77
1046.5
1108.73
1174.66
1244.51
1318.51
1396.91
1479.98
1567.98
1661.22
1760
1864.66
1975.53
2093
2217.46



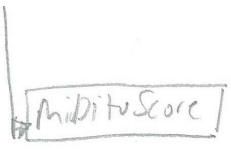
2349.32
2489.02
2637.02
2793.83
2959.96
3135.96
3322.44
3520
3729.31
3951.07
4186.01
4434.92
4698.63
4978.03
5274.04

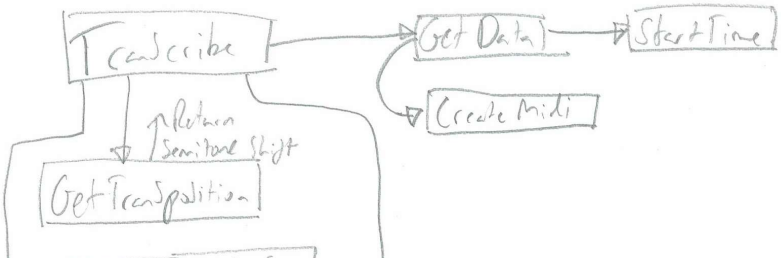

Transcribe
Get Data
Start Time
Create Midi
Return Semitone Shift
Get Transposition
Get File Name to Scan
MidiToScore

Appendix v

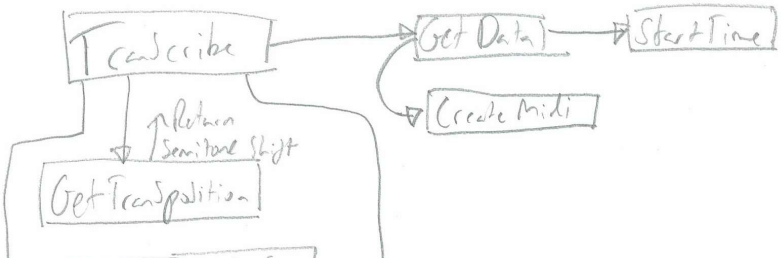
Main.py

```
from fbs_runtime.application_context import ApplicationContext
from PyQt5 import QtCore, QtGui, QtWidgets
import sys,os
from PyQt5.QtCore import pyqtSlot,QThread,pyqtSignal,QObject
from PyQt5.QtGui import *
import time, random , datetime
import dropboxUPLOAD as sql
import locateResources
import requests
global df
import pdb
import signal
import threading
import queue
import pyaudio
import aubio
import numpy as np
import notateWAVpostRecording

class guiThread(QThread):
       def __init__(self):
              QThread.__init__(self)
       def __del__(self):
              self.wait()
       def run(self):
              app.exec_()

class downloadWindowStart(QThread):
        def __init__(self):
               QThread.__init__(self)
        def __del__(self):
               self.wait()
        def run(self):
               downloadSelected_clicked()
class transAudioThread(QThread):
    def __init__(self):
        QThread.__init__(self)
    def __del__(self):
        self.wait()
    def run(self):
        whileGo()
        ##audio function start()

class timerThread(QThread): ###STILL NOT REALTIME
    def __init__(self):
        QThread.__init__(self)
    def __del__(self):
        self.wait()
    def run(self):
        scribeUI.timer.start(1)


##MainWindow
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.setFixedSize(331, 295)
        MainWindow.setTabShape(QtWidgets.QTabWidget.Rounded)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
```

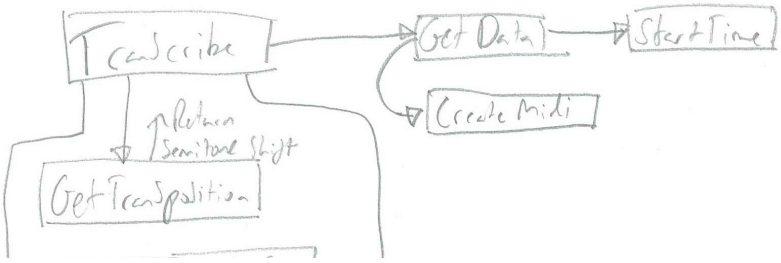




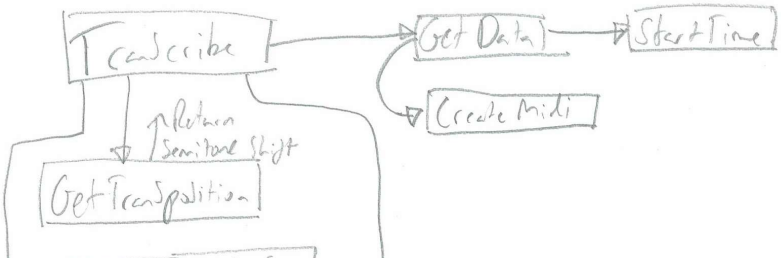

```
        self.splitter = QtWidgets.QSplitter(self.centralwidget)
        self.splitter.setGeometry(QtCore.QRect(30, 16, 271, 251))
        self.splitter.setOrientation(QtCore.Qt.Vertical)
        self.splitter.setObjectName("splitter")
        self.transposeButton = QtWidgets.QPushButton(self.splitter)
        font = QtGui.QFont()
        font.setPointSize(15)
        self.transposeButton.setFont(font)
        self.transposeButton.setObjectName("transposeButton")
        self.transcribeButton = QtWidgets.QPushButton(self.splitter)
        font = QtGui.QFont()
        font.setPointSize(15)
        self.transcribeButton.setFont(font)
        self.transcribeButton.setObjectName("transcribeButton")
        self.sqlSearchButton = QtWidgets.QPushButton(self.splitter)
        font = QtGui.QFont()
        font.setPointSize(15)
        self.sqlSearchButton.setFont(font)
        self.sqlSearchButton.setObjectName("sqlSearchButton")
        self.settingsButton = QtWidgets.QPushButton(self.splitter)
```



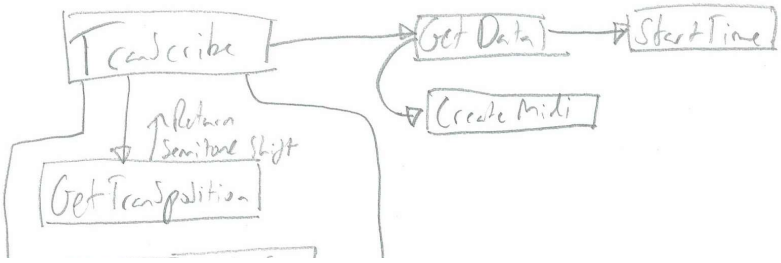

```
        font.setPointSize(12)
        self.label.setFont(font)
        self.label.setAlignment(QtCore.Qt.AlignCenter)
        self.label.setWordWrap(True)
        self.label.setObjectName("label")
        self.ignoreButton = QtWidgets.QPushButton(self.centralwidget)
        self.ignoreButton.setGeometry(QtCore.QRect(130, 200, 111, 31))
        self.ignoreButton.setObjectName("ignoreButton")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 390, 26))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        MainWindow.setWindowIcon(QIcon(LOGO_PATH))
        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)
```



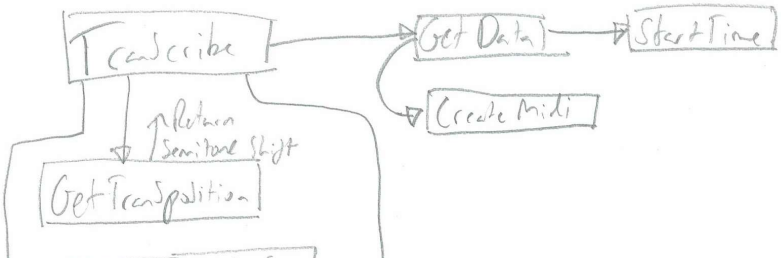

```
        self.midiDownloadCheckbox.setObjectName("midiDownloadCheckbox")
        self.scoreDownloadCheckbox = QtWidgets.QCheckBox(self.splitter)
        self.scoreDownloadCheckbox.setObjectName("scoreDownloadCheckbox")
        self.searchButton = QtWidgets.QPushButton(self.centralwidget)
        self.searchButton.setGeometry(QtCore.QRect(70, 140, 63, 28))#80, 140, 111, 28
        self.searchButton.setObjectName("searchButton")
        self.viewAllButton = QtWidgets.QPushButton(self.centralwidget)
        self.viewAllButton.setGeometry(QtCore.QRect(137, 140, 63, 28))
        self.viewAllButton.setObjectName("viewAllButton")
        self.sqlResultTable = QtWidgets.QTableView(MainWindow)
        self.sqlResultTable.setStyle(QtWidgets.QStyleFactory.create('Fusion'))
        self.sqlResultTable.setSelectionBehavior(self.sqlResultTable.SelectRows)

self.sqlResultTable.setSelectionMode(QtWidgets.QAbstractItemView.SingleSelection)
        self.sqlResultTable.setGeometry(QtCore.QRect(300, 30, 310, 550))
        self.sqlResultTable.setObjectName("sqlResultTable")
        self.sqlResultTable.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)

##self.sqlResultTable.setHorizontalHeaderLabels(["ID","User","UploadName","UploadDate"])
        MainWindow.setCentralWidget(self.centralwidget)
```



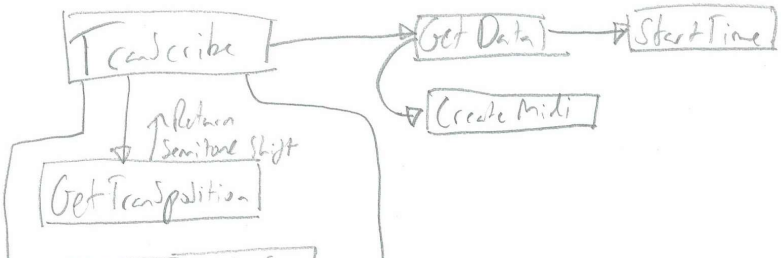

```
class PandasModel(QtCore.QAbstractTableModel):
    def __init__(self, data, parent=None):
        QtCore.QAbstractTableModel.__init__(self, parent)
        self._data = data

    def rowCount(self, parent=None):
        return len(self._data.values)

    def columnCount(self, parent=None):
        return self._data.columns.size

    def data(self, index, role=QtCore.Qt.DisplayRole):
        if index.isValid():
            if role == QtCore.Qt.DisplayRole:
                return QtCore.QVariant(str(
                    self._data.values[index.row()][index.column()]))
        return QtCore.QVariant()
```



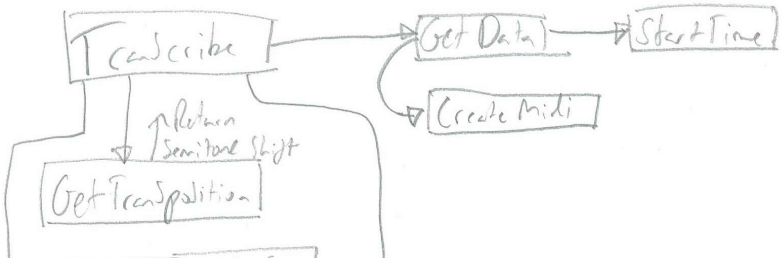

```
def generateSQL(searchOptions,searchType,data):
    query = "SELECT * FROM info WHERE "
    idxs = [i for i, x in enumerate(searchOptions) if x == True]
    for idx in idxs:
        current = "cast([{}] as nvarchar(max)) LIKE '%{}%' AND
".format(searchType[idx],data[idx])
        query += current
    if query[-5:]==" AND ":
        query = query[0:len(query)-5]
    sqlWindowUI.shellTextBoxSQL.setText(query)
    return query



def excepthook(excType, excValue, tracebackobj):
    """
    Global function to catch unhandled exceptions.

    @param excType exception type
    @param excValue exception value
```



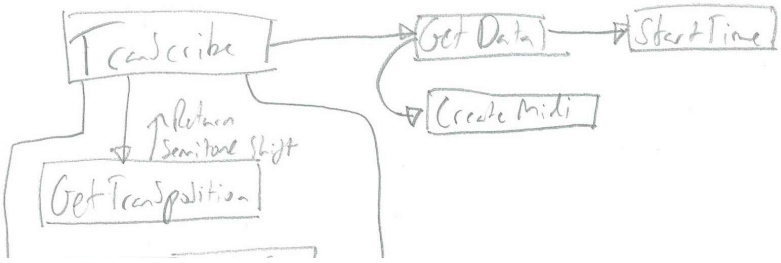

```
    if midiCheck == False and scoreCheck == False:
        raiseError("No File Type Selected to download")
        out =  ("ERROR: NOT SELECTED")
    elif len(idxSelected)==0:
        raiseError("No Entry Selected")
        out =  ("ERROR: NO ENTRY SELECTED")
    else:
        idx = idxSelected[0].row()
        out = ("""MIDIFILE CHECKED {}
SCOREFILE CHECKED {}
INDEX SELECTED {}""".format(midiCheck,scoreCheck,idx))
        sqlWindowUI.shellTextBoxDownload.setText(out)
        dropboxID = sqlWindowUI.df.dropboxID[idx]
        midiurl,scoreurl,urlList = getDownloadLinks(dropboxID)
        if midiCheck:
            out += "\n MIDI URL: {}".format(midiurl)
        else:
            urlList.remove(midiurl)
```



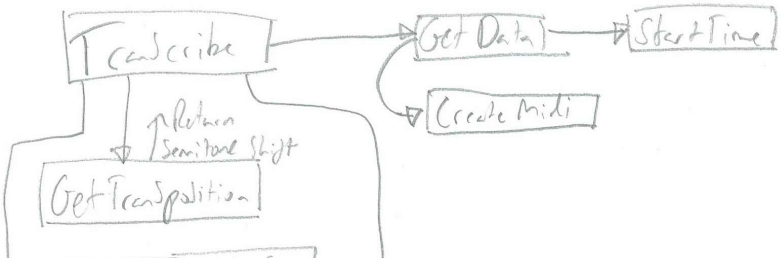

```
        return midiurl,scoreurl,urlList


def setupSqlSlots():
    sqlWindowUI.downloadButton.clicked.connect(downloadSelected_clicked)
    sqlWindowUI.searchButton.clicked.connect(searchButton_clicked)
    sqlWindowUI.viewAllButton.clicked.connect(viewAllButton_clicked)
##Recording Worker
class Worker(QObject):
    finished = pyqtSignal()  # our signal out to the main thread to alert it we've
completed our work
    def __init__(self):
        super(Worker, self).__init__()
        self.working = True  # this is our flag to control our loop

    def work(self):
        import pyaudio
        import wave
        global pa
        pa = pyaudio.PyAudio()
```



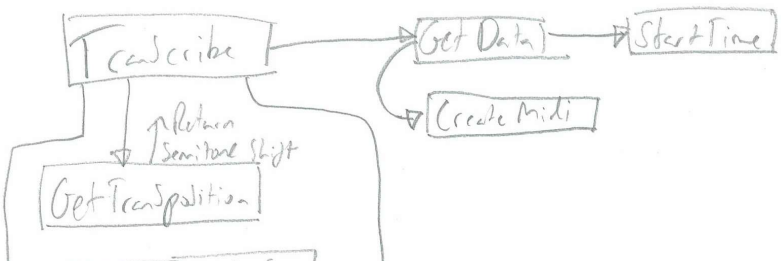

```
        self.userNameTextBox.setGeometry(QtCore.QRect(340, 90, 113, 22))
        self.userNameTextBox.setObjectName("userNameTextBox")
        self.label_3 = QtWidgets.QLabel(self.centralwidget)
        self.label_3.setGeometry(QtCore.QRect(260, 170, 181, 16))
        self.label_3.setObjectName("label_3")
        self.splitter = QtWidgets.QSplitter(self.centralwidget)
        self.splitter.setGeometry(QtCore.QRect(20, 10, 100, 143))
        self.splitter.setOrientation(QtCore.Qt.Vertical)
        self.splitter.setObjectName("splitter")
        self.startKeyLabel = QtWidgets.QLabel(self.splitter)
        font = QtGui.QFont()
        font.setPointSize(10)
        font.setBold(True)
        font.setWeight(75)
        self.startKeyLabel.setFont(font)
        self.startKeyLabel.setCursor(QtGui.QCursor(QtCore.Qt.ArrowCursor))
        self.startKeyLabel.setObjectName("startKeyLabel")
        self.startKeyDial = QtWidgets.QDial(self.splitter)
        self.startKeyDial.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
        self.startKeyDial.setMinimum(0)
```




```
        self.endKeyDial.setSingleStep(1)
        self.endKeyDial.setPageStep(2)
        self.endKeyDial.setProperty("value", 0)
        self.endKeyDial.setSliderPosition(0)
        self.endKeyDial.setOrientation(QtCore.Qt.Horizontal)
        self.endKeyDial.setWrapping(True)
        self.endKeyDial.setNotchTarget(12.0)
        self.endKeyDial.setNotchesVisible(True)
        self.endKeyDial.setObjectName("endKeyDial")
        self.endKeyCombo = QtWidgets.QComboBox(self.splitter_2)
        self.endKeyCombo.setEditable(False)
        self.endKeyCombo.setMaxVisibleItems(13)
        self.endKeyCombo.setMinimumContentsLength(1)
        self.endKeyCombo.setDuplicatesEnabled(True)
        self.endKeyCombo.setObjectName("endKeyCombo")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
```

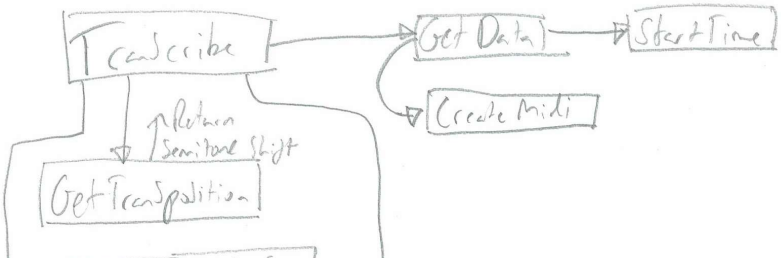



```
            self.worker.moveToThread(self.thread)  # move the worker into the thread, do
this first before connecting the signals

            self.thread.started.connect(self.worker.work)  # begin our worker object's
loop when the thread starts running
            self.stopButton.clicked.connect(self.stop_loop)  # stop the loop on the stop
button click
            self.worker.finished.connect(self.loop_finished)  # do something in the gui
when the worker loop ends
            self.worker.finished.connect(self.thread.quit)  # tell the thread it's time
to stop running
            self.worker.finished.connect(self.worker.deleteLater)  # have worker mark
itself for deletion
            self.thread.finished.connect(self.thread.deleteLater)  # have thread mark
itself for deletion
            try:
                    idx = open("audioSettingsFile.txt","r").readlines()[1].split(",")[1]
                    if idx == "":
                            raiseError("Please set Audio Device in settings")
                    else:
```



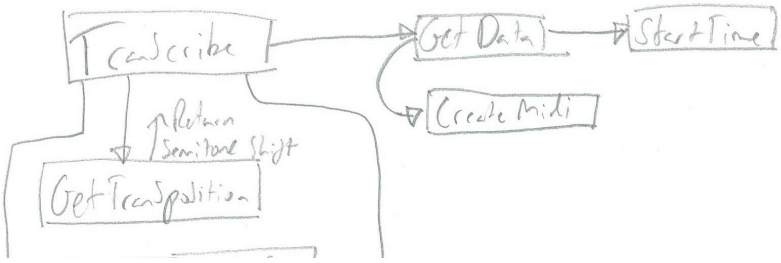

```
            ms = 0
        elif s == 59 and m < 59:
            m+=1
            s = 0
            ms = 0
        else:
            if m < 59:
                s = 0
                m += 1
            elif m == 59 and h < 24:
                h += 1
                m = 0
                s = 0
                ms = 0
            else:
                self.timer.stop()
                raiseError("Recording Too Long")

    time = "{0:02d}:{1:02d}:{2:02d}:{3:03d}".format(h,m,s,ms)
```

Computer Science NEA - NoteSwitch

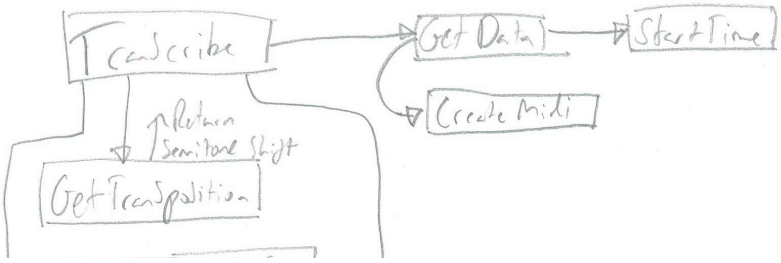



```
        Dialog.setFixedSize(557, 359)
        self.sensitivityDataTxt = QtWidgets.QLabel(Dialog)
        self.sensitivityDataTxt.setGeometry(QtCore.QRect(190, 130, 71, 21))
        self.sensitivityDataTxt.setFrameShape(QtWidgets.QFrame.WinPanel)
        self.sensitivityDataTxt.setWordWrap(False)
        self.sensitivityDataTxt.setObjectName("sensitivityDataTxt")
        self.startKeyLabel = QtWidgets.QLabel(Dialog)
        self.startKeyLabel.setGeometry(QtCore.QRect(50, 30, 101, 16))
        font = QtGui.QFont()
        font.setPointSize(10)
        font.setBold(True)
        font.setWeight(75)
        self.startKeyLabel.setFont(font)
        self.startKeyLabel.setCursor(QtGui.QCursor(QtCore.Qt.ArrowCursor))
        self.startKeyLabel.setObjectName("startKeyLabel")
        self.endKeyLabel = QtWidgets.QLabel(Dialog)
        self.endKeyLabel.setGeometry(QtCore.QRect(50, 60, 101, 21))
        font = QtGui.QFont()
        font.setPointSize(10)
        font.setBold(True)
```

Computer Science NEA - NoteSwitch

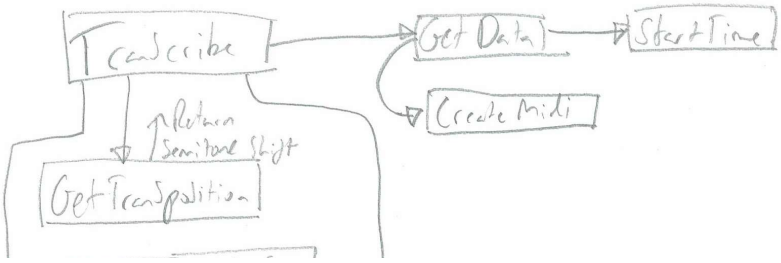



```
        self.verticalLayout.setObjectName("verticalLayout")
        self.currentNoteImage = QtWidgets.QLabel(self.verticalLayoutWidget)
        self.currentNoteImage.setText("")
        self.currentNoteImage.setObjectName("currentNoteImage")
        self.verticalLayout.addWidget(self.currentNoteImage)
        self.splitter = QtWidgets.QSplitter(Dialog)
        self.splitter.setGeometry(QtCore.QRect(290, 10, 200, 100))
        self.splitter.setOrientation(QtCore.Qt.Horizontal)
        self.splitter.setObjectName("splitter")
        self.startKeyDial = QtWidgets.QDial(self.splitter)
        self.startKeyDial.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
        self.startKeyDial.setMinimum(0)
        self.startKeyDial.setMaximum(13)
        self.startKeyDial.setSingleStep(1)
        self.startKeyDial.setPageStep(2)
        self.startKeyDial.setProperty("value", 0)
        self.startKeyDial.setSliderPosition(0)
        self.startKeyDial.setOrientation(QtCore.Qt.Horizontal)
        self.startKeyDial.setWrapping(True)
        self.startKeyDial.setNotchTarget(13.0)
```

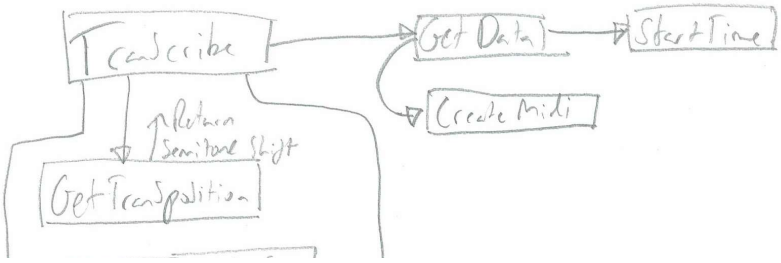



```
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.endKeyCombo.addItem("")
        self.splitter_3 = QtWidgets.QSplitter(Dialog)
        self.splitter_3.setGeometry(QtCore.QRect(280, 130, 241, 32))
        self.splitter_3.setOrientation(QtCore.Qt.Horizontal)
        self.splitter_3.setObjectName("splitter_3")
        self.label_2 = QtWidgets.QLabel(self.splitter_3)
        self.label_2.setAutoFillBackground(False)
        self.label_2.setObjectName("label_2")
        self.sensitivitySlider = QtWidgets.QSlider(self.splitter_3)
```

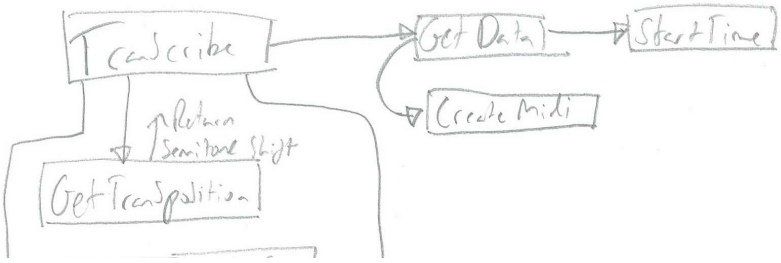
Computer Science NEA - NoteSwitch




```
        self.currentInitialNoteLabel.setText(_translate("Dialog", "Initial Note"))
        self.currentInitialNoteUpdateLabel.setText(_translate("Dialog", "-"))
        self.startKeyCombo.setItemText(0, _translate("Dialog", "-"))
        self.startKeyCombo.setItemText(1, _translate("Dialog", "C"))
        self.startKeyCombo.setItemText(2, _translate("Dialog", "C#/Db"))
        self.startKeyCombo.setItemText(3, _translate("Dialog", "D"))
        self.startKeyCombo.setItemText(4, _translate("Dialog", "D#/Eb"))
        self.startKeyCombo.setItemText(5, _translate("Dialog", "E"))
        self.startKeyCombo.setItemText(6, _translate("Dialog", "F"))
        self.startKeyCombo.setItemText(7, _translate("Dialog", "F#/Gb"))
        self.startKeyCombo.setItemText(8, _translate("Dialog", "G"))
        self.startKeyCombo.setItemText(9, _translate("Dialog", "G#/Ab"))
        self.startKeyCombo.setItemText(10, _translate("Dialog", "A"))
        self.startKeyCombo.setItemText(11, _translate("Dialog", "A#/Bb"))
        self.startKeyCombo.setItemText(12, _translate("Dialog", "B"))
        self.endKeyCombo.setItemText(0, _translate("Dialog", "-"))
        self.endKeyCombo.setItemText(1, _translate("Dialog", "C"))
        self.endKeyCombo.setItemText(2, _translate("Dialog", "C#/Db"))
        self.endKeyCombo.setItemText(3, _translate("Dialog", "D"))
        self.endKeyCombo.setItemText(4, _translate("Dialog", "D#/Eb"))
```

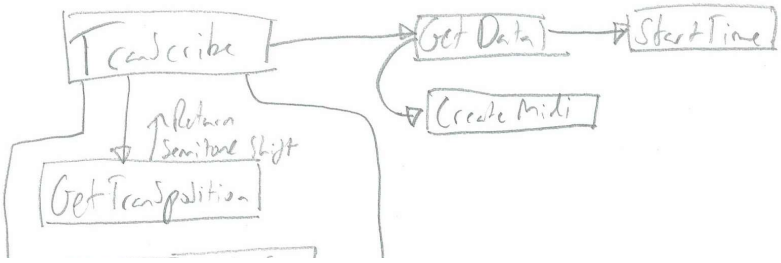



```
        self.refreshAudioDevicesBUtton.setFont(font)
        self.refreshAudioDevicesBUtton.setObjectName("refreshAudioDevicesBUtton")
        self.audioDeviceComboBox = QtWidgets.QComboBox(settingsWindow)
        self.audioDeviceComboBox.setGeometry(QtCore.QRect(190, 40, 161, 22))
        self.audioDeviceComboBox.setObjectName("audioDeviceComboBox")

        self.retranslateUi(settingsWindow)
        QtCore.QMetaObject.connectSlotsByName(settingsWindow)

    def retranslateUi(self, settingsWindow):
        _translate = QtCore.QCoreApplication.translate
        settingsWindow.setWindowTitle(_translate("settingsWindow", "Settings"))
        self.audioInputDeviceLabel.setText(_translate("settingsWindow", "Audio Input
Device:"))
        self.applyAudioSettingsButton.setText(_translate("settingsWindow", "Apply"))
        self.refreshAudioDevicesBUtton.setText(_translate("settingsWindow", "Refresh
Devices"))


class Ui_downloadWindow(object):
```

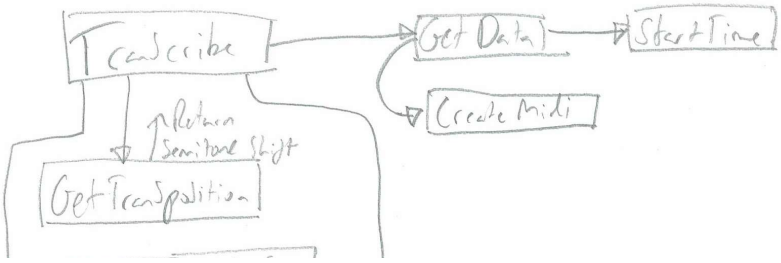



```
                print ("Attempting to stop threads")
                liveAudioIdentification.join()
                print ("Stopped audio identification thread")
                time.sleep(0.5)
                print ("Attempting to stop transAudio Thread")
                transAudioTR.terminate()
                print ("Stopped transAudio Thread")
        except Exception as e:
                raiseError("StopButton Error")
                print (e)
def convertNoteToOnlySharps(note):
        if note == "B#":
                return "C"
        elif note == "Eb":
                return "D#"
        elif note =="Ab":
                return "G#"
        elif note =="Bb":
                return "A#"
        elif note =="Cb":
```

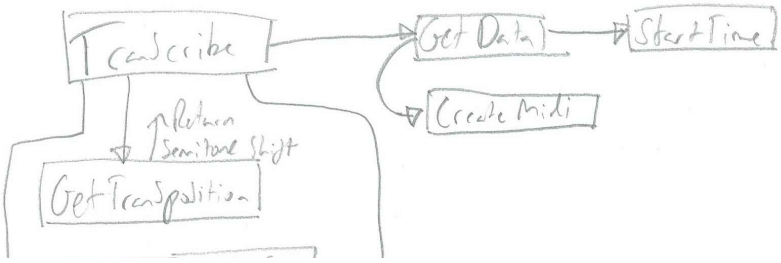
Computer Science NEA - NoteSwitch



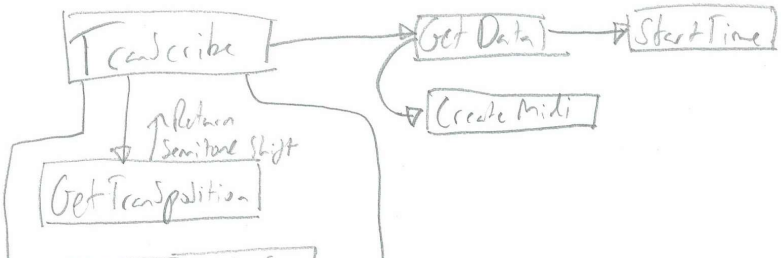

```
        try:
                return newNote
        except UnboundLocalError:
                return newNote

@pyqtSlot()
def updateNoteLabel(note):
        transUI.currentInitialNoteUpdateLabel.setText(note[0].upper()+note[1:])
        ##transpose into next key
        note = transposeNote(note)
        path = locateResources.findFile(note+".png")
        transUI.currentNoteUpdateLabel.setText(note.upper())
        notePicture = QPixmap(path)
        notePicture = notePicture.scaled(250, 250)
        transUI.currentNoteImage.setPixmap(notePicture)
```



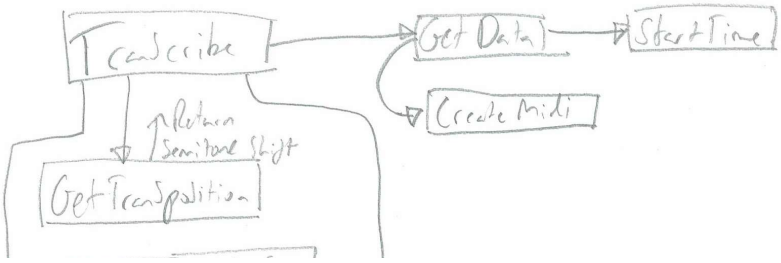

```
                        scribeUI.startKeyCombo.setEnabled(False)
                        scribeUI.endKeyCombo.setEnabled(False)
                        scribeUI.fileNameTextBox.setEnabled(False)
                        scribeUI.userNameTextBox.setEnabled(False)
                else:
                        raiseError("File Name Required")
        else:
                if scribeUI.fileNameTextBox.text() =="":
                        raiseError("File Name Required")
                else:
                        raiseError("Username Required")


else:
        try:
                idx = open("audioSettingsFile.txt","r").readlines()[1].split(",")[1]
                if idx == "":
                        raiseError("Please set Audio Device in settings")
                else:
                        print ("LOOP SHOULD START")
```



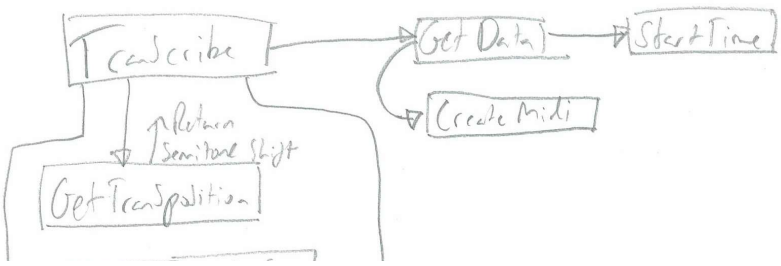

```
def checkUniqueName(name):
        unique = False
        cnxn = sql.connect_DB()
        try:
                name = "/score/"+name+"_scoreFile.pdf"
                cnxn.files_get_temporary_link(name)
                unique = False
        except:
                unique = True
        return unique
def debugTrace():
    from PyQt5.QtCore import pyqtRemoveInputHook, pyqtRestoreInputHook
    import pdb
    import sys
    pyqtRemoveInputHook()
    try:
        debugger = pdb.Pdb()
        debugger.reset()
        debugger.do_next(None)
        user_frame = sys._getframe().f_back
```



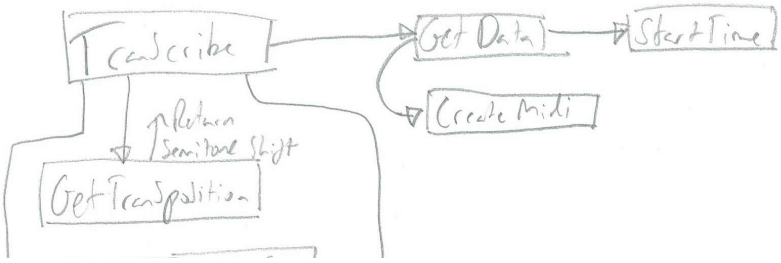

```
    scribeUI.startButton.setEnabled(True)
    scribeUI.stopButton.setEnabled(False)
    scribeUI.resetButton.setEnabled(True)
    scribeUI.startKeyDial.setEnabled(True)
    scribeUI.endKeyDial.setEnabled(True)
    scribeUI.startKeyCombo.setEnabled(True)
    scribeUI.endKeyCombo.setEnabled(True)
    scribeUI.fileNameTextBox.setEnabled(True)
    scribeUI.userNameTextBox.setEnabled(True)
    scribeUI.timer.stop()
    ###save()


@pyqtSlot()
def transcribeResetButton_clicked():
    scribeUI.startButton.setEnabled(True)
    scribeUI.stopButton.setEnabled(False)
    scribeUI.resetButton.setEnabled(True)
    scribeUI.startKeyDial.setEnabled(True)
    scribeUI.endKeyDial.setEnabled(True)
```



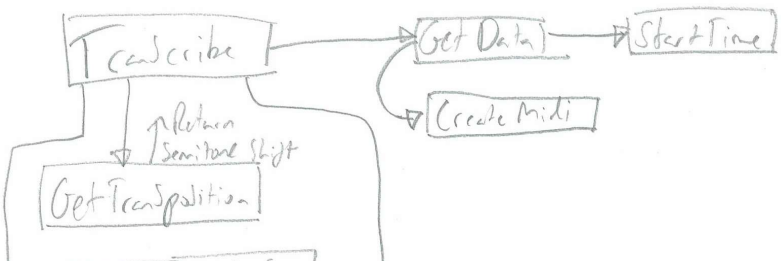

```
    transUI.stopButton.clicked.connect(stopButton_clicked)
    transUI.sensitivitySlider.valueChanged.connect(getSensitivityData)
    transUI.startButton.clicked.connect(startButton_clicked)

def setupSlotsMain():
    mainUI.transposeButton.clicked.connect(transposeButton_clicked)
    mainUI.transcribeButton.clicked.connect(transcribeButton_clicked)
    mainUI.sqlSearchButton.clicked.connect(sqlButton_clicked)
    mainUI.settingsButton.clicked.connect(settingsButton_clicked)
@pyqtSlot()
def applyAudioSettings():
        raiseError("Settings Applied!",'#77dd77',"Success!")
        audioDeviceSettingIs = settingsUI.audioDeviceComboBox.currentText()
        audioDeviceSettingIndex = settingsUI.audioDeviceComboBox.currentIndex()
        print ("audio device selected = "+str(audioDeviceSettingIs))
        with open("audioSettingsFile.txt","w") as f:
                f.write("FORMAT:DEVICENAME,INDEX\n")
                f.write("{},{}".format(audioDeviceSettingIs,
                                       audioDeviceSettingIndex))
                f.close()
```



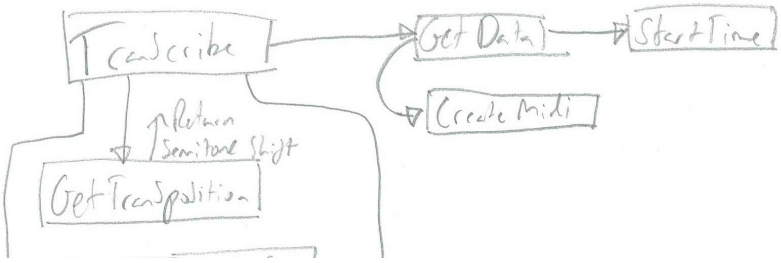

```
background-color: {}
}};""".format("#FF6961"))
    else:
            errorWindow.setStyleSheet("""QMainWindow{{
background-color: {}
}};""".format(colouring))
    if text=="":
            errorUI.errorLabel.setText("Error!")
            errorWindow.setWindowTitle("Error Message")
    else:
            errorUI.errorLabel.setText(text)
            errorWindow.setWindowTitle("{}".format(text))
    errorUI.label.setText(str(message))
    errorWindow.show()

class liveNoteThread(threading.Thread):
        def __init__(self,name='liveNoteThread',sensitivity=0.6):
                self._stopevent = threading.Event()
                self.identifiedNoteQueue = queue.Queue()
                self.sensitivity = sensitivity
```




```
            print ("stream initialised")
        except Exception as e:
            print ("[E] {}".format(e))
        while True:
            data = self.stream.read(1024)
            samples = np.fromstring(data,
                dtype=aubio.float_type)
            pitch = pDetection(samples)[0]
            sens = 5.0*(int(self.sensitivity)/6)
            if len(sumPitch)<sens:
                sumPitch.append(round(pitch))
            else:
                accu=int()
                for item in sumPitch:
                    accu +=item
                frequenc = round(np.mean(accu)/20)
                self.closest(frequenc)
        ##            print frequenc
                sumPitch=list()
```

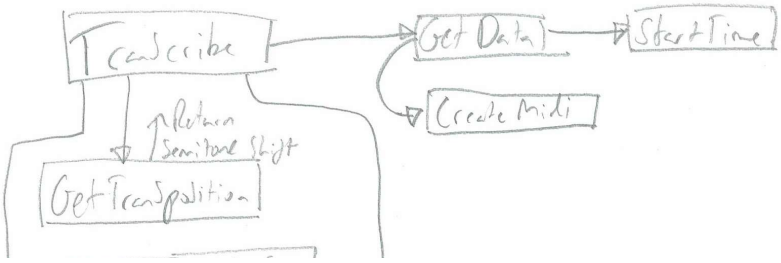



```
ss:
                    MainWindow.setStyleSheet(ss.read())
                global mainUI
                mainUI = Ui_MainWindow()
                mainUI.setupUi(MainWindow)


                setupSlotsMain()

                global transposeWindow
                transposeWindow = QtWidgets.QDialog()
                transposeWindow.setWindowIcon(QIcon(LOGO_PATH))
                transposeWindow.setWindowTitle("Transpose") ##################
                global transUI
                transUI = Ui_Dialog()
                transUI.setupUi(transposeWindow)
                transUI.currentNoteImage.setPixmap(QtGui.QPixmap(LOGO_PATH))
                setupSlotsTrans()
                ##StyleSheet
                with open(locateResources.findFile("transWindow.stylesheet"),"r") as ss:
```

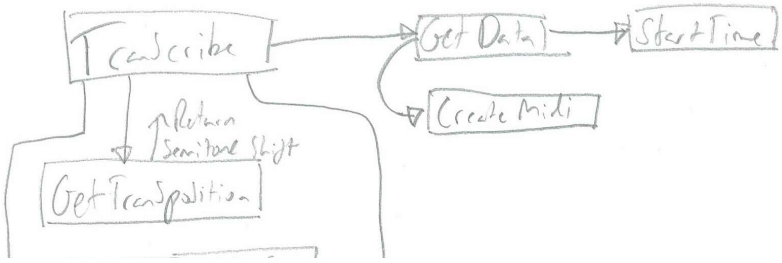



```
##    sqlWindowUI.refreshTable(df) ##populate with data from sql

global downloadWindow
global downloadWindowUI
downloadWindow = QtWidgets.QDialog()
downloadWindowUI = Ui_downloadWindow()
downloadWindowUI.setupUi(downloadWindow)


MainWindow.show()
transposeWindow.hide()
transcribeWindow.hide()
errorWindow.hide()
sqlWindow.hide()
downloadWindow.hide()
settingsWindow.hide()

global transAudioTR
transAudioTR = transAudioThread()
mainGui = guiThread()
```

Computer Science NEA - NoteSwitch

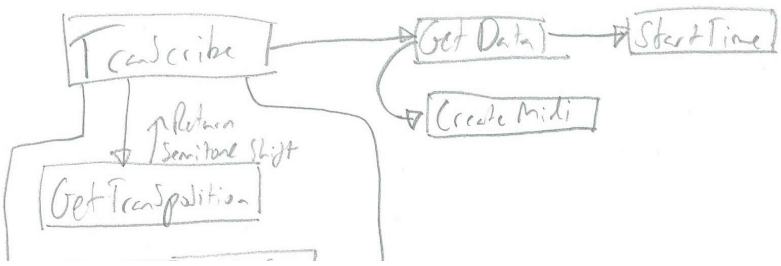



```
            pass
        self.MidiData.append(note)
        self.duration.append(duration)
        self.volume.append(100)
        self.NoteOnSet.append(onset)
    def setTempo(self,bpm):
        self.tempo = bpm
    def export(self):
        self.MidiData = self.MidiData[1:]
        self.duration = self.duration[1:]
        self.volume = self.volume[1:]
        self.NoteOnSet = self.NoteOnSet[1:]

def toValueOctave(txt):
    print(txt)
    txt = txt.upper()
    octave = int(txt[-1])+1
    note = txt[:-1]
    if note == "Eb":
        note = "D#"
```

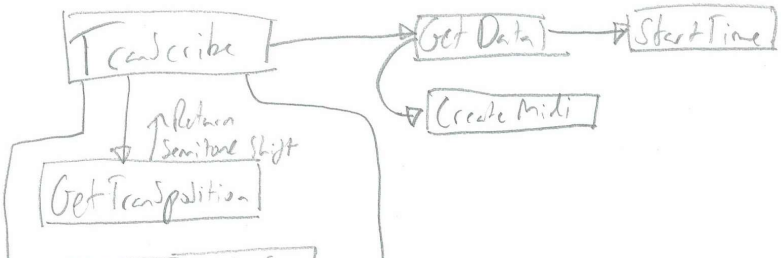



```
### Calls batch script to convert, param1 is the input and param2 is the output file in
the curdir
    print("[D] Converted")
    outputfiles = (os.getcwd(),(str("{}_midiFile.mid".format(name)),str(name)))
    print(outputfiles)
    print ("[D] Waiting until score file exists")
    while not (os.path.exists("{}_scoreFile.pdf".format(name))):
        print ("[D] PDF FILE NOT FOUND")
        time.sleep(1)
    print ("[D] PDF FILE FOUND")


def debugTrace():
    from PyQt5.QtCore import pyqtRemoveInputHook, pyqtRestoreInputHook
    import pdb
    import sys
    pyqtRemoveInputHook()
    try:
        debugger = pdb.Pdb()
        debugger.reset()
```

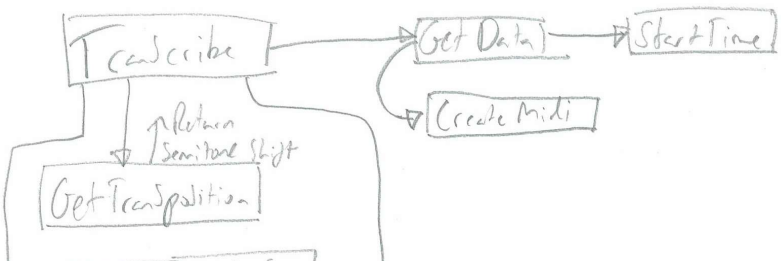
Computer Science NEA - NoteSwitch




```
        pitches += [pitch]
        confidences += [confidence]
        total_frames += read
        if read < hop_s: break


    tempdata = list()

    timeOffsets = 1.0/64
    noteOffsets = 0
    startTime = 0
    endTime = 0
    condensedData = list() #[note,startTime,endTime,Duration]
    for entry in data:
        ##print entry,
        idx = data.index(entry)
        try:
                if data[idx+1][1]-noteOffsets <= data[idx][1]
<=data[idx+1][1]+noteOffsets:
                #print("SAME NOTE")
```



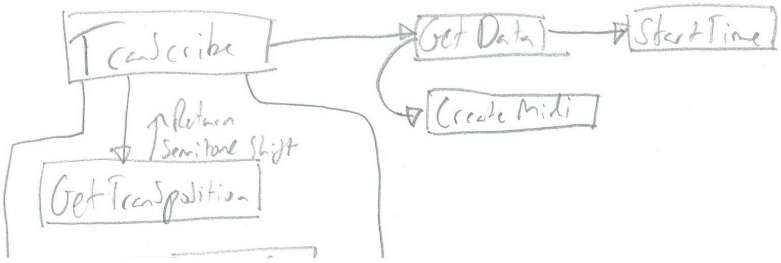

```
win_s = 4096 // downsample # fft size
hop_s = 512  // downsample # hop size


s = source(path, samplerate, hop_s)
samplerate = s.samplerate
o = tempo("specdiff", win_s, hop_s, samplerate)
# List of beats, in samples
beats = []
# Total number of frames read
total_frames = 0

while True:
    samples, read = s()
    is_beat = o(samples)
    if is_beat:
        this_beat = o.get_last_s()
        beats.append(this_beat)
        #if o.get_confidence() > .2 and len(beats) > 2.:
        #    break
```



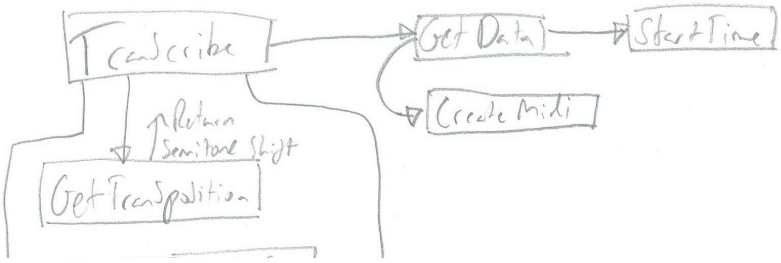

```
                return newNote

def postAnalyseAudio(wav_file,name,transpositionValue):
    AudioData=dataStore()
    AudioData = analyse(wav_file,AudioData)
    AudioData.setTempo(round(getBPM(wav_file)))
    save(AudioData,name,transpositionValue)
```

locateResources.py

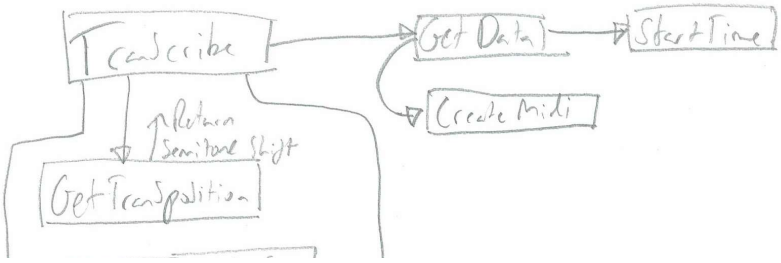
```
import os
import difflib
cwd = os.getcwd()
possibleFiles,possiblePaths = list(),list()

def findMostSimilarFile(fileToFind,possible):
    similarFileFound = difflib.get_close_matches(fileToFind,possible,1,cutoff=0.3)[0]
```




```
from dropbox.files import WriteMode
from dropbox.exceptions import AuthError
import pandas as pd
import os
import pyodbc
if __name__ == "__main__":
        midiFileName = 'out.mid'
        scoreFileName = 'uploadtest1.pdf'

        access_token =
'Iw2ShQdqqwAAAAAAAAAAFHPJuDGkEFBvA4vqaFqBes6W-YBZ_nzRwiVHfk7nDumj'
        fileName = ''
        filePathDropBox = "/"+'' #Name on dropbox

        connection = dropbox.Dropbox(access_token)
        try:
                connection.users_get_current_account()
        except AuthError as err:
           print ("[E] ERROR: Invalid access token; try re-generating an access token
from the app console on the web.")
```



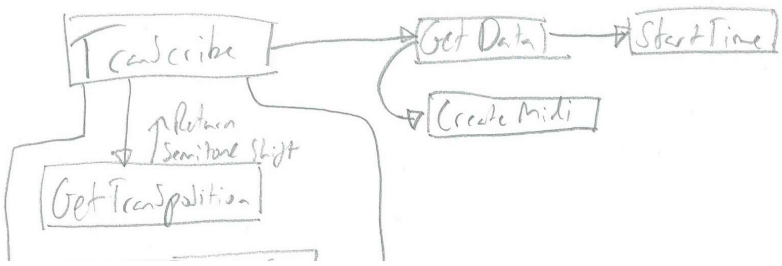

```
                print (e)
        return cnxn

def query(cmd,cnxn):
        try:
            df = pd.read_sql_query(cmd,cnxn) ##use single quotes for ntext
            cnxn.commit()
            return df

        except Exception as e:
                print ("[E] Error: {}".format(e))

def checkHash(HASH,column,cnxn):
        unique = False
        df = query("SELECT {0} FROM info WHERE [{0}]=({1})".format(column,HASH),cnxn)
        if len(df) == 0:
                unique = True
        else:
                unique = False
        return unique
```



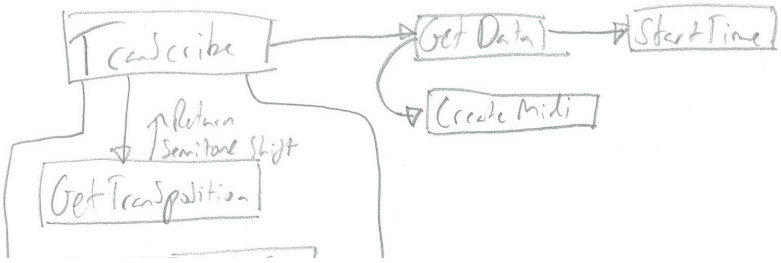

```
        border-color: black;
        font: bold;
        }
```

transcribeWindow.stylesheet

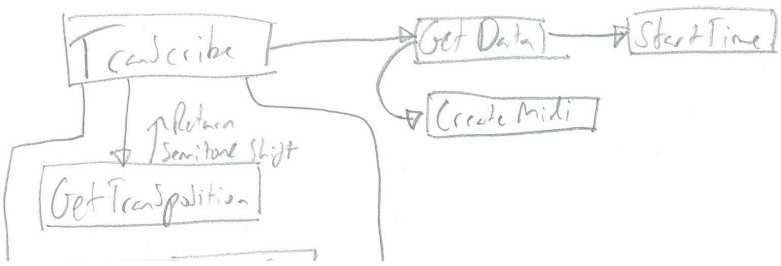
```
QLineEdit{
        border:2px;
        border-color:black;
        background-color:#E6E6E6;
        }

QDial {
        background-color: #E6E6E6;
        }
```




```
    font: bold 20px;
}
QPushButton#stopButton{
    background-color: red;
    border-color: black;
    font: bold 20px;
}

QDialog{
        background-color: #B3B3B3;
        }

QComboBox {
    border: 0px;
        border: black;
    border-radius: 0px;
        background-color: #E6E6E6;
}
```




```
        }
QTableView{
        background-color:#E6E6E6;
        selection-background-color:#779ECB;
        border:5px;
        border-color:black;
        border-radius:5px;
        }

QHeaderView::section{
        background-color:#B3B3B3;
        font: bold;}

QSplitter::handle{
        image: None;}

QPushButton{
        border: 4px;
        border-color:black;
```

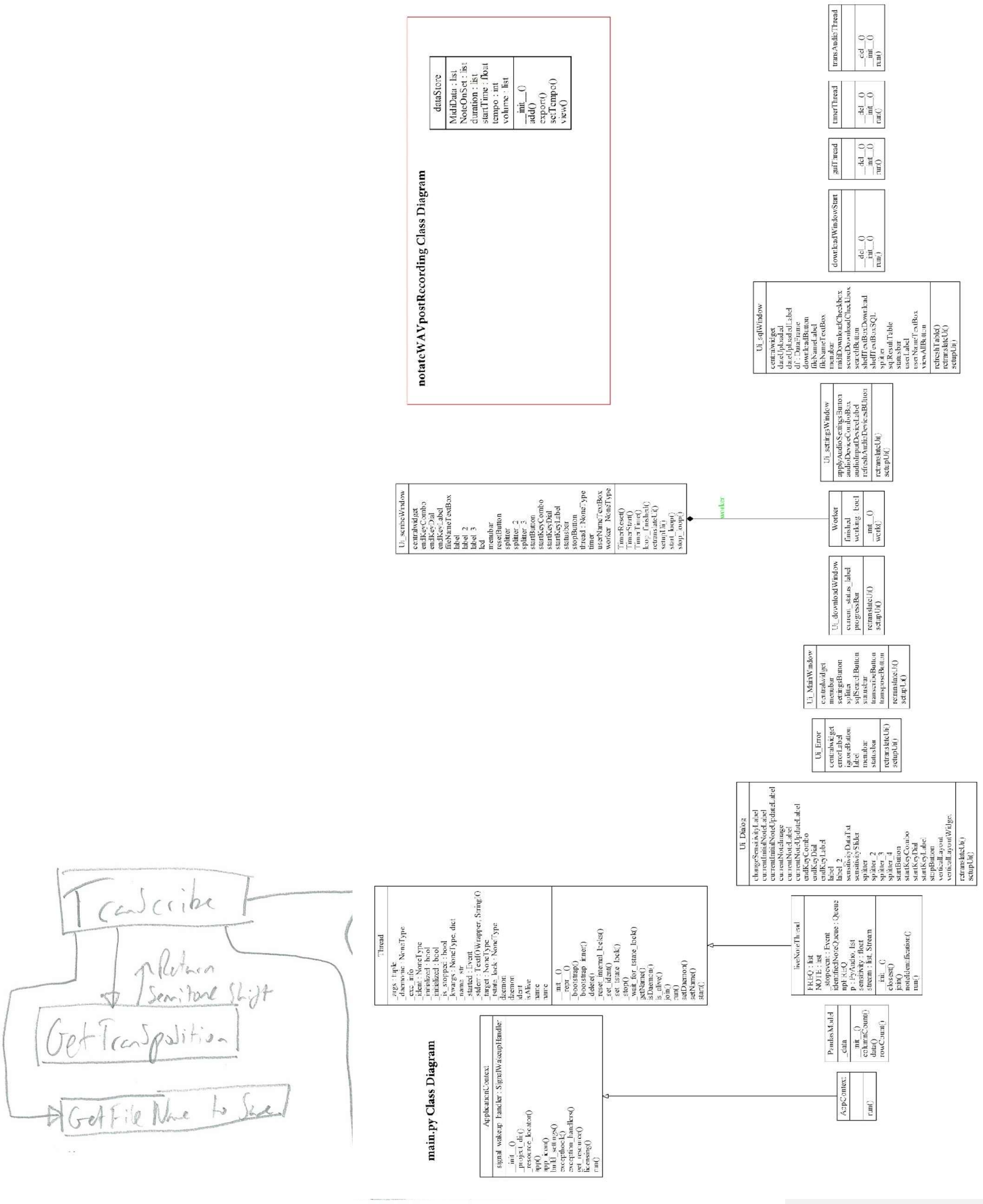



```
QSplitter::handle{
        image: None;}
```

Convert.bat

```
@echo off
start "CONVERT" "Musescore 3\bin\MuseScore3.exe" %~1 -o %~2%.pdf
EXIT /B 0
```

Appendix vi



notateWAVpostRecording Class Diagram
main.py Class Diagram

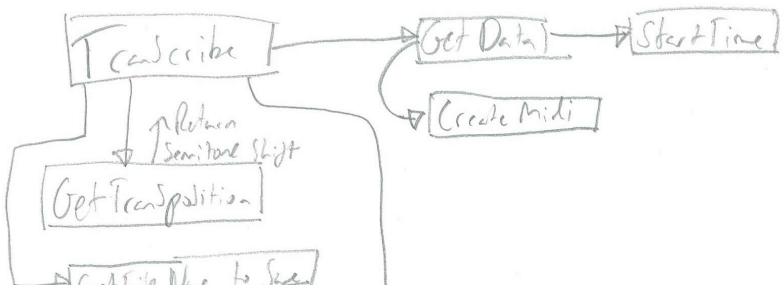
Appendix vii

NoteSwitch - Final End User Review - Oscar Lindenbaum

Currently, it is complicated to either transpose or identify the notes being played on any musical instrument without years of learning by ear and perfect pitch. As a musician myself I have found on multiple occasions where either I create a melody or tune and play it but when I want to switch to my other saxophone I would have to transpose all of the notes and remember where they all are on both instruments or pass my sheet music to another player and because their instrument is in a different key so therefore the notes are different. What the application does is make it easier to switch between instruments with either written music or single notes without a lot of thinking and working out the notes one by one. The transpose option does single live note transposition, the transcribe option does whole recordings and the SQL Search allows you to download and share your work.

4097 Oscar Lindenbaum 62113 Wheatley Park School

Computer Science NEA - NoteSwitch



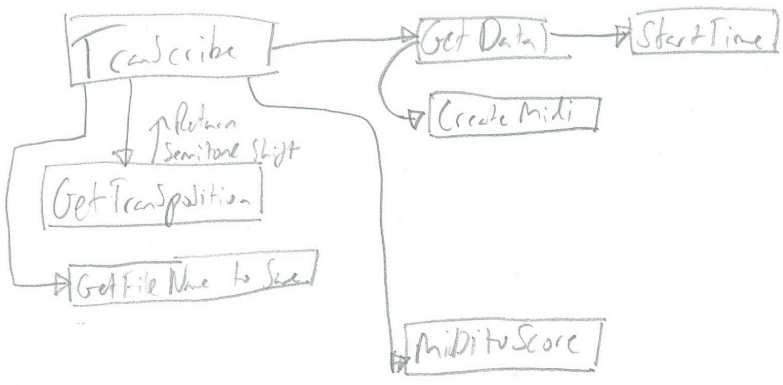
Questions
Name: **Age:** **Date:**

COMMENTS IN BOLD

- *What do you think of the applications layout?*

- *How did you find navigating the application?*

- *Where could you see a use for this in your current music life?*


Transcribe
Get Data
Start Time
Create Midi
Return Semitone Shift
Get Transposition
Get File Name to Scan
MidiToScore

Appendix viii

Questions

Name: LOUIS HODGSON **Age:** 17

COMMENTS IN BOLD

Date: 13.03.19.

- *What do you think of the applications layout?*

very easy to ~~navigate~~ use and very clear.

- *How did you find navigating the application?*

very easy to move between the different parts of the application.

- *Where could you see a use for this in your current music life?*

When coming up with musical ideas, I can use it to create a score with ease.

- *Would the program encourage you to create your own musical compositions and why?*

yes because it makes the transfer between live music and ideas to a score a much quicker and easier process. Also, it means that I can transfer saxophone into score which I can't do on other programmes because I can't plug in my sax.

- *What additional features would you like to see implemented?*

Being able to layer lines of music on top of eachother.

- *Do you think that the program was too complicated or that you might have trouble using it?*

NO, it was very easy to use.

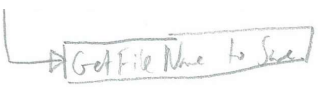
- *Where there any problems when using the application?*

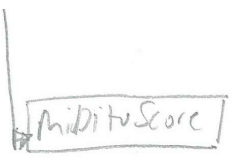
NO, it ~~was~~ ran very smoothly and was very effective.

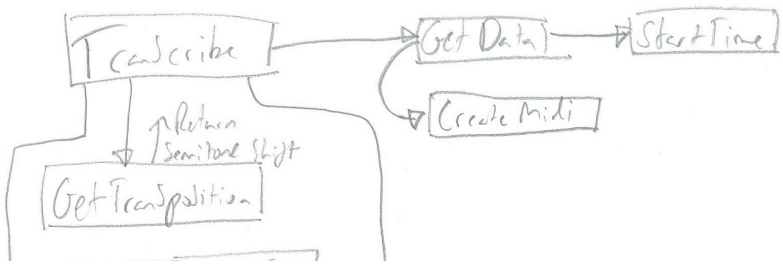
- *Any additional comments?*

I think it's a very useful application for non-electrical instruments like the saxophone.

Signed LHodgson.







Questions
Name: Libby Jervis **Age:** 17

COMMENTS IN BOLD
Date: 13.03.19

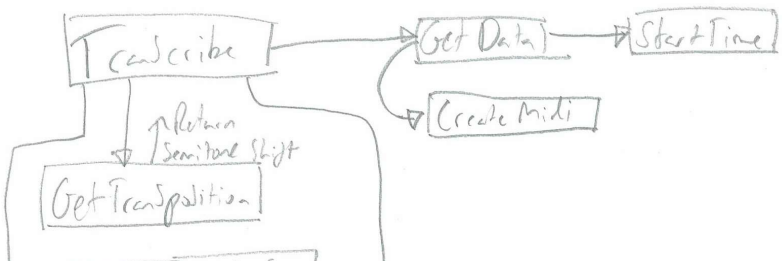
- *What do you think of the applications layout?*

Looked nice, easy to read

- *How did you find navigating the application?*

Pretty clear

- *Where could you see a use for this in your current music life?*



Questions **COMMENTS IN BOLD**

Name: Stan Prichard **Age:** 15 **Date:** 13 / 3 / 19

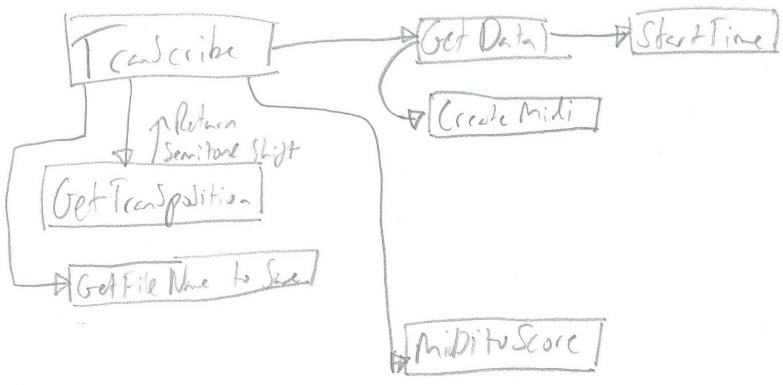
- *What do you think of the applications layout?*

It was simple but effective and relatively easy to navigate.

- *How did you find navigating the application?*

relatively easy

- *Where could you see a use for this in your current music life?*


Transcribe
Get Data
Start Time
Create Midi
Return Semitone Shift
Get Transposition
Get File Name to Scan
MidiToScore