TECHNICAL MANUAL

by
Aaron Ennis
C00190504

Supervisor: Paul Barry

Music Application Project

Date: 18/04/2018

Institiúid Teicneolaíochta Cheatharlach

INSTITUTE *of* TECHNOLOGY
CARLOW

At the heart of South Leinster

Table of Contents

Introduction

In this document, you will find the code I wrote for the application along with how to get the application running on your machine. I list the dependencies of the application to run and talk about the hardware required to use it.

How to run

Dependencies

1. Python 3 (Python 3 compiler to run the code)
2. PyAudio (Used to record and create the .wav files)
3. Numpy (Maths library for algorithms)
4. PyQt5 (Framework for the user interface)

How to acquire these dependencies?

You can get any version of Python 3 on the official python.org website. For this application, I wrote it using Python 3.6.

If you install the python package installer “pip”, each of the dependencies can be easily installed in the command prompt with the command “py -3 -m pip install” followed by the name of the module you would like to install. This is the quickest and easiest way for installing modules in python. For example: py -3 -m pip install numpy. You can download pip at pypi.org.

If you do not want to use the pip command, each one of the modules can be downloaded from their official sites. You can download the zipped contents of each package and manually put them in to the libraries of your Python 3 folder. You need to be careful doing it this way as it can be done wrong and won't work. This is why I strongly recommend using pip to install the modules.

Running the application

After all of the dependencies are installed correctly, all you have to do to run the application is open the main.py file. You can do this by double clicking on it, or by opening a command prompt in the working directory and entering the command “py main.py”. The only thing you will need other than these dependencies, is a working microphone on the machine and a guitar.

Code

Music_Utils

Music_Utils.py is where I wrote all of the algorithms for the project. This file contains the back bone of the application. Here is where I created the functions to record, create the .wav files and generate the tab.

The info() function at the beginning of the code is not used in the application but was left in this file in case the user wants to find all of the information of a .wav file. It can be called by running a command prompt in the working directory of the application. Then type the following commands.

1. py
2. import music_utils
3. music_utils.info("file.wav")

It should look like this:

```
C:\Users\aaron>cd C:\Users\aaron\Desktop\Music-Application

C:\Users\aaron\Desktop\Music-Application>py
Python 3.6.3 |Anaconda custom (64-bit)| (default, Nov  8 2017, 15:10:56) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import music_utils
>>> music_utils.info("test.wav")
channels
1

get samp width
2

get frame rate(sample rate)
22050

get n frames(number of samples)
223232

lenght of audio in seconds
10.12390022675737

get file params
_wave_params(nchannels=1, sampwidth=2, framerate=22050, nframes=223232, comptype='NONE', compname='not compressed')

b'\x00\x00'
(0, 0)
>>>
```

Here is the code contained in music_utils.py.

```
## Author: Aaron Ennis
## Title: Music Application
##
## Description:
## A music application that allows the user to record/play back WAV files.
## The application transcribes the audio data from the WAV files and
## transcribes the notes to tabliture form and displays it.
#_____________________________________________________________________

import wave, struct, os, time, json
import pyaudio
import numpy as np
from array import array
from struct import pack


def info(_file):
    #Current working dir
    cwd = os.getcwd()
    existing_files = os.listdir(cwd + '\Recordings')
    if _file in existing_files:
        file_path = os.path.join(cwd + '\Recordings', _file)
        #open a wav format music
        f = wave.open(file_path,"rb")

        print("channels")
        print(f.getnchannels()) ##Returns number of audio channels (1 for mono, 2 for stereo).
        print(" ")
        print("get samp width")
        print(f.getsampwidth()) ##Returns sample width in bytes.
        print(" ")
        print("get frame rate(sample rate)")
        print(f.getframerate()) ##Returns sampling frequency
        print(" ")
        print("get n frames(number of samples)")
        print(f.getnframes()) ##Returns number of audio frames.
        print(" ")
        print("lenght of audio in seconds ")
        print(f.getnframes() / f.getframerate())
```

```
        print("lenght of audio in seconds ")
        print(f.getnframes() / f.getframerate())
        print(" ")
        print("get file params")
        print(f.getparams()) ##Returns a namedtuple() (nchannels, sampwidth, framerate
        print(" ")
        wave_data = f.readframes(1) ##Reads and returns at most n frames of audio
        print(wave_data)
        print(struct.unpack("hh", b"\x00\x00\x00\x00"))

        f.close()
    else:
        print('This file does not exist')

#plays audio
def play(_file):
    #Current working dir
    cwd = os.getcwd()
    existing_files = os.listdir(cwd + '\Recordings')
    if _file in existing_files:
        file_path = os.path.join(cwd + '\Recordings', _file)

        #open a wav format music
        f = wave.open(file_path,"rb")

        #define stream chunk
        chunk = 1024

        #instantiate PyAudio
        p = pyaudio.PyAudio()
        #open stream
        stream = p.open(format = p.get_format_from_width(f.getsampwidth()),
                        channels = f.getnchannels(),
                        rate = f.getframerate(),
                        output = True)
        #read data
        data = f.readframes(chunk)
```

```
75
76          #play stream
77          while data:
78              stream.write(data)
79              data = f.readframes(chunk)
80
81          #stop stream
82          stream.stop_stream()
83          stream.close()
84
85          #close PyAudio
86          p.terminate()
87      else:
88          print('This file does not exist')
89
90  def delete(_file):
91      #Current working dir
92      cwd = os.getcwd()
93      existing_files = os.listdir(cwd + '\Recordings')
94      tab = _file[:-4]
95      tab = tab + '.json'
96      if _file in existing_files:
97          file_path = os.path.join(cwd + '\Recordings', _file)
98          tab_file_path = os.path.join(cwd + '\Tabs', tab)
99          if os.path.isfile(file_path) and os.path.isfile(tab_file_path):
100             os.unlink(file_path)
101             os.unlink(tab_file_path)
102         else:
103             pass
104     else:
105         print('This file does not exist')
106
107 def check_tab():
108     #Current working dir
109     cwd = os.getcwd()
110     existing_files = os.listdir(cwd + '\Recordings')
111     if len(os.listdir(cwd + '\Tabs')) > 0:
112         existing_tabs = os.listdir(cwd + '\Tabs')
```

```
    existing_files = os.listdir(cwd + '\Recordings')
    if len(os.listdir(cwd + '\Tabs')) > 0:
        existing_tabs = os.listdir(cwd + '\Tabs')
        tabs = []
        files = []
        for t in existing_tabs:
            t=t[:-5]
            tabs.append(t)
        for f in existing_files:
            f=f[:-4]
            files.append(f)

        for t in tabs:
            if t not in files:
                tab_file_path = os.path.join(cwd + '\Tabs', t + '.json')
                if os.path.isfile(tab_file_path):
                    os.unlink(tab_file_path)
            else:
                pass
    else:
        pass

def get_tab(_file):
    #strings on the guitar
    e = ''
    B = ''
    G = ''
    D = ''
    A = ''
    E = ''
    #Current working dir
    cwd = os.getcwd()
    existing_tabs = os.listdir(cwd + '\Tabs')
    json_file = _file[:-4] + '.json'
    if len(os.listdir(cwd + '\Tabs')) > 0:
        if json_file in existing_tabs:
            with open(os.path.join(cwd + '\Tabs', '%s' % (json_file)), 'r') as f:
```

```
146             with open(os.path.join(cwd + '\Tabs', '%s' % (json_file)), 'r') as f:
147                 data = json.load(f)
148 
149             if json_file in os.listdir(cwd + '\Tabs'):
150                 notes = []
151                 count = 0
152                 tab = []
153                 #gets every fourth note
154                 for v in data.values():
155                     notes.append(v)
156                 for i in notes:
157                     if count == 3:
158                         tab.append(i)
159                         count = 0
160                     else:
161                         count += 1
162                 #Creating the tab
163                 for n in tab:
164                     if n[0] == 'A':                 #Note
165                         if n[-1:] == '3':           #Octave
166                             e = e + '- '
167                             B = B + '- '
168                             G = G + '- '
169                             D = D + '- '
170                             A = A + '0 '
171                             E = E + '- '
172                         elif n[-1:] == '4':
173                             e = e + '- '
174                             B = B + '- '
175                             G = G + '2 '
176                             D = D + '- '
177                             A = A + '- '
178                             E = E + '- '
179                         elif n[-1:] == '5':
180                             e = e + '5 '
181                             B = B + '- '
182                             G = G + '- '
183                             D = D + '- '
```

```
178                             E = E + '- '
179                         elif n[-1:] == '5':
180                             e = e + '5 '
181                             B = B + '- '
182                             G = G + '- '
183                             D = D + '- '
184                             A = A + '- '
185                             E = E + '- '
186
187                     elif n[0] == 'A#':
188                         if n[-1:] == '3':
189                             e = e + '- '
190                             B = B + '- '
191                             G = G + '- '
192                             D = D + '- '
193                             A = A + '1 '
194                             E = E + '- '
195                         elif n[-1:] == '4':
196                             e = e + '- '
197                             B = B + '- '
198                             G = G + '3 '
199                             D = D + '- '
200                             A = A + '- '
201                             E = E + '- '
202                         elif n[-1:] == '5':
203                             e = e + '6 '
204                             B = B + '- '
205                             G = G + '- '
206                             D = D + '- '
207                             A = A + '- '
208                             E = E + '- '
209
210                     elif n[0] == 'B':
211                         if n[-1:] == '3':
212                             e = e + '- '
213                             B = B + '- '
214                             G = G + '- '
215                             D = D + '- '
```

```
                    G = G + '- '
                    D = D + '- '
                    A = A + '2 '
                    E = E + '- '
                elif n[-1:] == '4':
                    e = e + '- '
                    B = B + '- '
                    G = G + '4 '
                    D = D + '- '
                    A = A + '- '
                    E = E + '- '
                elif n[-1:] == '5':
                    e = e + '7 '
                    B = B + '- '
                    G = G + '- '
                    D = D + '- '
                    A = A + '- '
                    E = E + '- '

            elif n[0] == 'C':
                if n[-1:] == '3':
                    e = e + '- '
                    B = B + '- '
                    G = G + '- '
                    D = D + '- '
                    A = A + '3 '
                    E = E + '- '
                elif n[-1:] == '4':
                    e = e + '- '
                    B = B + '- '
                    G = G + '5 '
                    D = D + '- '
                    A = A + '- '
                    E = E + '- '
                elif n[-1:] == '5':
                    e = e + '8 '
                    B = B + '- '
                    G = G + '- '
```

```
                    D = D + '- '
                    A = A + '- '
                    E = E + '- '

                elif n[0] == 'C#':
                    if n[-1:] == '3':
                        e = e + '- '
                        B = B + '- '
                        G = G + '- '
                        D = D + '- '
                        A = A + '4 '
                        E = E + '- '
                    elif n[-1:] == '4':
                        e = e + '- '
                        B = B + '- '
                        G = G + '6 '
                        D = D + '- '
                        A = A + '- '
                        E = E + '- '
                    elif n[-1:] == '5':
                        e = e + '9 '
                        B = B + '- '
                        G = G + '- '
                        D = D + '- '
                        A = A + '- '
                        E = E + '- '

                elif n[0] == 'D':
                    if n[-1:] == '3':
                        e = e + '- '
                        B = B + '- '
                        G = G + '- '
                        D = D + '0 '
                        A = A + '- '
                        E = E + '- '
                    elif n[-1:] == '4':
                        e = e + '- '
                        B = B + '3 '
```

```
289                         e = e + '- '
290                         B = B + '3 '
291                         G = G + '- '
292                         D = D + '- '
293                         A = A + '- '
294                         E = E + '- '
295                     elif n[-1:] == '5':
296                         e = e + '10 '
297                         B = B + '- '
298                         G = G + '- '
299                         D = D + '- '
300                         A = A + '- '
301                         E = E + '- '
302
303                 elif n[0] == 'D#':
304                     if n[-1:] == '3':
305                         e = e + '- '
306                         B = B + '- '
307                         G = G + '- '
308                         D = D + '1 '
309                         A = A + '- '
310                         E = E + '- '
311                     elif n[-1:] == '4':
312                         e = e + '- '
313                         B = B + '4 '
314                         G = G + '- '
315                         D = D + '- '
316                         A = A + '- '
317                         E = E + '- '
318                     elif n[-1:] == '5':
319                         e = e + '11 '
320                         B = B + '- '
321                         G = G + '- '
322                         D = D + '- '
323                         A = A + '- '
324                         E = E + '- '
325
326                 elif n[0] == '5':
```

```
                elif n[0] == 'E':
                    if n[-1:] == '3':
                        e = e + '- '
                        B = B + '- '
                        G = G + '- '
                        D = D + '- '
                        A = A + '- '
                        E = E + '0 '
                    elif n[-1:] == '4':
                        e = e + '- '
                        B = B + '- '
                        G = G + '- '
                        D = D + '2 '
                        A = A + '- '
                        E = E + '- '
                    elif n[-1:] == '5':
                        e = e + '0 '
                        B = B + '- '
                        G = G + '- '
                        D = D + '- '
                        A = A + '- '
                        E = E + '- '

                elif n[0] == 'F':
                    if n[-1:] == '3':
                        e = e + '- '
                        B = B + '- '
                        G = G + '- '
                        D = D + '- '
                        A = A + '- '
                        E = E + '1 '
                    elif n[-1:] == '4':
                        e = e + '- '
                        B = B + '- '
                        G = G + '- '
                        D = D + '3 '
                        A = A + '- '
                        E = E + '- '
```

```
365                         e = e + '1 '
366                         B = B + '- '
367                         G = G + '- '
368                         D = D + '- '
369                         A = A + '- '
370                         E = E + '- '
371
372                     elif n[0] == 'F#':
373                         if n[-1:] == '3':
374                             e = e + '- '
375                             B = B + '- '
376                             G = G + '- '
377                             D = D + '- '
378                             A = A + '- '
379                             E = E + '2 '
380                         elif n[-1:] == '4':
381                             e = e + '- '
382                             B = B + '- '
383                             G = G + '- '
384                             D = D + '4 '
385                             A = A + '- '
386                             E = E + '- '
387                         elif n[-1:] == '5':
388                             e = e + '2 '
389                             B = B + '- '
390                             G = G + '- '
391                             D = D + '- '
392                             A = A + '- '
393                             E = E + '- '
394
395                     elif n[0] == 'G':
396                         if n[-1:] == '3':
397                             e = e + '- '
398                             B = B + '- '
399                             G = G + '- '
400                             D = D + '- '
401                             A = A + '- '
402                             E = E + '3 '
```

```
403                     elif n[-1:] == '4':
404                         e = e + '- '
405                         B = B + '- '
406                         G = G + '0 '
407                         D = D + '- '
408                         A = A + '- '
409                         E = E + '- '
410                     elif n[-1:] == '5':
411                         e = e + '3 '
412                         B = B + '- '
413                         G = G + '- '
414                         D = D + '- '
415                         A = A + '- '
416                         E = E + '- '
417 
418                 elif n[0] == 'G#':
419                     if n[-1:] == '3':
420                         e = e + '- '
421                         B = B + '- '
422                         G = G + '- '
423                         D = D + '- '
424                         A = A + '- '
425                         E = E + '- '
426                     elif n[-1:] == '4':
427                         e = e + '- '
428                         B = B + '- '
429                         G = G + '1 '
430                         D = D + '- '
431                         A = A + '- '
432                         E = E + '- '
433                     elif n[-1:] == '5':
434                         e = e + '4 '
435                         B = B + '- '
436                         G = G + '- '
437                         D = D + '- '
438                         A = A + '- '
439                         E = E + '- '
440                 elif n[0] == '-':
```

```
439                             E = E + '- '
440                     elif n[0] == '-':
441                             e = e + '- '
442                             B = B + '- '
443                             G = G + '- '
444                             D = D + '- '
445                             A = A + '- '
446                             E = E + '- '
447
448                 return e,B,G,D,A,E
449
450         else:
451             e = e + '-'
452             B = B + '-'
453             G = G + 'No Tab Found'
454             D = D + '-'
455             A = A + '-'
456             E = E + '-'
457             return e,B,G,D,A,E
458     else:
459         e = e + '-'
460         B = B + '-'
461         G = G + 'No Tab Found'
462         D = D + '-'
463         A = A + '-'
464         E = E + '-'
465         return e,B,G,D,A,E
466
467
468 def record(_recording_lenght):
469     NOTE_MIN = 40       # E2
470     NOTE_MAX = 76       # E4
471     FSAMP = 22050       # Sampling frequency in Hz
472     FRAME_SIZE = 2048   # samples per frame
473     FRAMES_PER_FFT = 16  # FFT takes average across how many frames?
474
475     ##############################################################################
476     # Derived quantities from constants above. Note that as
```

```
476     # Derived quantities from constants above. Note that as
477     # SAMPLES_PER_FFT goes up, the frequency step size decreases (sof
478     # resolution increases); however, it will incur more delay to process
479     # new sounds.
480
481     SAMPLES_PER_FFT = FRAME_SIZE * FRAMES_PER_FFT
482     FREQ_STEP = float(FSAMP) / SAMPLES_PER_FFT
483
484     ######################################################################
485     # For printing out notes
486
487     NOTE_NAMES = 'E F F# G G# A A# B C C# D D#'.split()
488
489     #frequency to MIDI number
490     def freq_to_number(f): return 76 + 12 * np.log2(f / 659.255)
491
492     #MIDI number to frequency
493     def number_to_freq(n): return 659.255 * 2.0**((n - 76) / 12.0)
494
495     #Gets the names of the notes + octaves
496     def note_name(n):
497         return NOTE_NAMES[n % NOTE_MIN % len(NOTE_NAMES)] + str(int(n / 12 - 1))
498
499     #Gets discrete FFT bin
500     def note_to_fftbin(n): return number_to_freq(n) / FREQ_STEP
501
502     #Gets real Min and Max freq
503     imin = max(0, int(np.floor(note_to_fftbin(NOTE_MIN - 1))))
504     imax = min(SAMPLES_PER_FFT, int(np.ceil(note_to_fftbin(NOTE_MAX + 1))))
505
506     # Allocate space to run an FFT.
507     buf = np.zeros(SAMPLES_PER_FFT, dtype=np.float32)
508     num_frames = 0
509
510     p = pyaudio.PyAudio()
511     # Initialize audio
512     stream = p.open(format=pyaudio.paInt16,
513                                     channels=1
```

```
518
519     # Create Hanning window function
520     window = 0.5 * (1 - np.cos(np.linspace(0, 2*np.pi, SAMPLES_PER_FFT, False)))
521
522     # Print initial text
523     print ('sampling at', FSAMP, 'Hz with max resolution of', FREQ_STEP, 'Hz')
524
525     stream.start_stream()
526     # As long as we are getting data:
527     start = time.time()
528     time.clock()
529     elapsed = 0
530     r = array('h')
531     notes = {}
532     num_frames = 0
533     print('recording')
534     #normalizes the audio to reduce white noise
535     audio_normalize = 'normalize.wav'
536     root = 'C:/Users/aaron/Desktop/Music-Application/Static'
537     normalize_rate, normalize_data = scipy.io.wavfile.read(os.path.join(root, audio_normalize))
538     snd_data = array('h', normalize_data)
539     r.extend(snd_data)
540     #records for x amount of seconds
541     while elapsed < _recording_lenght:
542         elapsed = time.time() - start
543         read = np.fromstring(stream.read(FRAME_SIZE), np.int16)
544         # Shift the buffer down and new data in
545         buf[:-FRAME_SIZE] = buf[FRAME_SIZE:]
546         buf[-FRAME_SIZE:] = read
547
548         snd_data = array('h', read)
549
550         r.extend(snd_data)
551         idx = np.abs(read).argmax()
552
553         if np.abs(read[idx]) > 300:
554             # Run the FFT on the windowed buffer
555             fft = np.fft.rfft(buf * window)
```

```
554             # Run the FFT on the windowed buffer
555             fft = np.fft.rfft(buf * window)
556 
557             # Get frequency of maximum response in range
558             freq = (np.abs(fft[imin:imax]).argmax() + imin) * FREQ_STEP
559 
560             # Get note number and nearest note
561             n = freq_to_number(freq)
562             n0 = int(round(n))
563 
564             # Console output once we have a full buffer
565             num_frames += 1
566 
567             print ('freq: ', freq, '  ', note_name(n0))
568             notes[elapsed] = note_name(n0)
569         else:
570             print ('---')
571             notes[elapsed] = '---'
572 
573     return notes, len(normalize_data), r, p, stream
574 
575 def create_file(_file, _notes, _normalize_data_lenght, r, p, stream):
576     #Current working dir
577     cwd = os.getcwd()
578     FSAMP = 22050       # Sampling frequency in Hz
579 
580     def normalize(snd_data):
581         #Average the volume out"
582         MAXIMUM = 16384
583         times = float(MAXIMUM)/max(abs(i) for i in snd_data)
584 
585         r = array('h')
586         for i in snd_data:
587             r.append(int(i*times))
588         return r
589 
590     def record_to_file(path,data,sW):
591         #Records from the microphone and outputs the resulting data to 'path'
```

```
582         MAXIMUM = 16384
583         times = float(MAXIMUM)/max(abs(i) for i in snd_data)
584 
585         r = array('h')
586         for i in snd_data:
587             r.append(int(i*times))
588         return r
589 
590     def record_to_file(path,data,sW):
591         #Records from the microphone and outputs the resulting data to 'path'
592         data = pack('<' + ('h'*len(data)), *data)
593 
594         wf = wave.open(path, 'wb')
595         wf.setnchannels(1)
596         wf.setsampwidth(sW)
597         wf.setframerate(FSAMP)
598         wf.writeframes(data)
599         wf.close()
600 
601     sample_width = p.get_sample_size(pyaudio.paInt16)
602     stream.stop_stream()
603     stream.close()
604     p.terminate()
605     r = normalize(r)
606     #removes the normalizing sound
607     r = r[_normalize_data_lenght:]
608 
609     path = os.path.join(cwd + '\Recordings', _file)
610     record_to_file(path,r,sample_width)
611 
612     json_data = {}
613     json_data[_file] = _notes
614     #create JSON
615     with open(os.path.join(cwd+ '\Tabs', '%s.json' % (_file[:-4])), 'w') as f:
616         json.dump(_notes, f)
617 
618 
```

Main

In the main.py file, all of the code to do with the user interface can be found. This is the main file of the application and it is the file that needs to be run to open the application.

```
## Author: Aaron Ennis
## Title: Music Application
##
## Description:
## A music application that allows the user to record/play back WAV files.
## The application transcribes the audio data from the WAV files and
## transcribes the notes to tablature form and displays it.
#_____________________________________________________________________

import sys
import music_utils
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *

#Thread class
class Worker(QRunnable):
    def __init__(self, fn, *args, **kwargs):
        super(Worker, self).__init__()
        # Store constructor arguments (re-used for processing)
        #fn = function for threads, arg/kwargs = function parameters
        self.fn = fn
        self.args = args
        self.kwargs = kwargs

    @pyqtSlot()
    def run(self):
        self.fn(*self.args, **self.kwargs)

class UIPlay(QWidget):
    def __init__(self, parent=None):
        super(UIPlay, self).__init__(parent)

        self.grid = QGridLayout()
        self.setLayout(self.grid)

        self.HOMESCREEN = QPushButton('Go to home', self)
        self.HOMESCREEN.resize(self.HOMESCREEN.sizeHint())
```

```
40          self.SMALLE = QLabel('', self)
41          self.B = QLabel('', self)
42          self.G = QLabel('', self)
43          self.D = QLabel('', self)
44          self.A = QLabel('', self)
45          self.E = QLabel('', self)
46
47          self.SMALLE2 = QLabel('', self)
48          self.B2 = QLabel('', self)
49          self.G2 = QLabel('', self)
50          self.D2 = QLabel('', self)
51          self.A2 = QLabel('', self)
52          self.E2 = QLabel('', self)
53
54          self.space = QLabel('', self)
55
56          newfont = QFont("Times", 14, QFont.Bold)
57          self.SMALLE.setFont(newfont)
58          self.B.setFont(newfont)
59          self.G.setFont(newfont)
60          self.D.setFont(newfont)
61          self.A.setFont(newfont)
62          self.E.setFont(newfont)
63
64          self.SMALLE2.setFont(newfont)
65          self.B2.setFont(newfont)
66          self.G2.setFont(newfont)
67          self.D2.setFont(newfont)
68          self.A2.setFont(newfont)
69          self.E2.setFont(newfont)
70
71          self.space.setFont(newfont)
72
73          self.PLAY_LBL = QLabel('Play', self)
74          self.DELETE_LBL = QLabel('Delete', self)
75
76          cwd = music_utils.os.getcwd()
77          existing_files = music_utils.os.listdir(cwd + '\Recordings')
            self.RECORDINGS = QComboBox(self)
```

```
77          existing_files = music_utils.os.listdir(cwd + '\Recordings')
78          self.RECORDINGS = QComboBox(self)
79          self.RECORDINGS.resize(self.RECORDINGS.sizeHint())
80          self.DELETE = QComboBox(self)
81          self.DELETE.resize(self.DELETE.sizeHint())
82
83          for i in existing_files:
84              #only pulls the Wav files
85              if i[-4:] == '.wav':
86                  self.RECORDINGS.addItem(str(i))
87                  self.DELETE.addItem(str(i))
88
89          self.grid.addWidget(self.SMALLE,1,1)
90          self.grid.addWidget(self.B,2,1)
91          self.grid.addWidget(self.G,3,1)
92          self.grid.addWidget(self.D,4,1)
93          self.grid.addWidget(self.A,5,1)
94          self.grid.addWidget(self.E,6,1)
95          self.grid.addWidget(self.space,7,1)
96          self.grid.addWidget(self.SMALLE2,8,1)
97          self.grid.addWidget(self.B2,9,1)
98          self.grid.addWidget(self.G2,10,1)
99          self.grid.addWidget(self.D2,11,1)
100         self.grid.addWidget(self.A2,12,1)
101         self.grid.addWidget(self.E2,13,1)
102
103         self.grid.addWidget(self.HOMESCREEN,14,0)
104         self.grid.addWidget(self.PLAY_LBL,13,3)
105         self.grid.addWidget(self.DELETE_LBL,13,7)
106         self.grid.addWidget(self.RECORDINGS,14,3)
107         self.grid.addWidget(self.DELETE,14,7)
108
109
110 class UIHome(QWidget):
111     def __init__(self, parent=None):
112         super(UIHome, self).__init__(parent)
113
114         self.grid = QGridLayout()
            self.setLayout(self.grid)
```

```
115         self.setLayout(self.grid)
116 
117         self.PLAYSCREEN = QPushButton("Go to play", self)
118         self.PLAYSCREEN.resize(self.PLAYSCREEN.sizeHint())
119         self.grid.addWidget(self.PLAYSCREEN,0,1)
120 
121         self.RECORDING_LBL = QLabel('Lenght of recording (seconds)', self)
122         self.RECORDING_LBL.resize(self.RECORDING_LBL.sizeHint())
123         self.grid.addWidget(self.RECORDING_LBL,1,0)
124 
125         self.RECORDED_LBL = QLabel('', self)
126         self.RECORDED_LBL.resize(self.RECORDING_LBL.sizeHint())
127         self.grid.addWidget(self.RECORDED_LBL,2,0)
128 
129         self.RECORDING_TIME = QComboBox(self)
130         self.RECORDING_TIME.addItem(str(5))
131         self.RECORDING_TIME.addItem(str(10))
132         self.RECORDING_TIME.addItem(str(15))
133         self.RECORDING_TIME.addItem(str(20))
134         self.RECORDING_TIME.addItem(str(25))
135         self.RECORDING_TIME.addItem(str(30))
136         self.RECORDING_TIME.resize(self.RECORDING_TIME.sizeHint())
137         self.grid.addWidget(self.RECORDING_TIME,1,1)
138 
139         self.RECORD = QPushButton("Record!", self)
140         self.RECORD.resize(self.RECORD.sizeHint())
141         self.grid.addWidget(self.RECORD,1,2)
142 
143         self.QUIT = QPushButton("Quit!", self)
144         self.QUIT.resize(self.QUIT.sizeHint())
145         self.grid.addWidget(self.QUIT,2,1)
146 
147 
148 class UIEmptyHome(QWidget):
149     def __init__(self, parent=None):
150         super(UIEmptyHome, self).__init__(parent)
151         self.grid = QGridLayout()
152         self.setLayout(self.grid)
```

```
153
154         self.PLAYSCREEN = QPushButton("Go to play", self)
155         self.PLAYSCREEN.resize(self.PLAYSCREEN.sizeHint())
156         self.PLAYSCREEN.setEnabled(False)
157         self.PLAYSCREEN.setVisible(False)
158         self.grid.addWidget(self.PLAYSCREEN,0,1)
159
160         self.RECORDING_LBL = QLabel('Lenght of recording (seconds)', self)
161         self.RECORDING_LBL.resize(self.RECORDING_LBL.sizeHint())
162         self.grid.addWidget(self.RECORDING_LBL,1,0)
163
164         self.RECORDED_LBL = QLabel('', self)
165         self.RECORDED_LBL.resize(self.RECORDING_LBL.sizeHint())
166         self.grid.addWidget(self.RECORDED_LBL,2,0)
167
168         self.RECORDING_TIME = QComboBox(self)
169         self.RECORDING_TIME.addItem(str(5))
170         self.RECORDING_TIME.addItem(str(10))
171         self.RECORDING_TIME.addItem(str(15))
172         self.RECORDING_TIME.addItem(str(20))
173         self.RECORDING_TIME.addItem(str(25))
174         self.RECORDING_TIME.addItem(str(30))
175         self.RECORDING_TIME.resize(self.RECORDING_TIME.sizeHint())
176         self.grid.addWidget(self.RECORDING_TIME,1,1)
177
178         self.RECORD = QPushButton("Record!", self)
179         self.RECORD.resize(self.RECORD.sizeHint())
180         self.grid.addWidget(self.RECORD,1,2)
181
182         self.QUIT = QPushButton("Quit!", self)
183         self.QUIT.resize(self.QUIT.sizeHint())
184         self.grid.addWidget(self.QUIT,2,1)
185
186 #Main window of application
187 class MainWindow(QMainWindow):
188     def __init__(self, parent=None):
189         super(MainWindow, self).__init__(parent)
190         self.setGeometry(50, 50, 650, 500)
            self.setWindowTitle('Music Application')
```

```
193         #setting threads
194         self.threadpool = QThreadPool()
195 
196 
197         self.Home_Screen = UIHome( self )
198         self.Home_Screen.PLAYSCREEN.clicked.connect( self.play_screen )
199         self.Home_Screen.RECORD.clicked.connect( self.record )
200         self.Home_Screen.QUIT.clicked.connect( self.quit_app )
201 
202         self.Empty_Home_Screen = UIEmptyHome( self )
203         self.Empty_Home_Screen.PLAYSCREEN.clicked.connect( self.play_screen )
204         self.Empty_Home_Screen.RECORD.clicked.connect( self.record )
205         self.Empty_Home_Screen.QUIT.clicked.connect( self.quit_app )
206 
207         self.Play_Screen = UIPlay( self )
208         self.Play_Screen.HOMESCREEN.clicked.connect( self.home_screen )
209         self.Play_Screen.RECORDINGS.activated[str].connect(self.play_audio)
210         self.Play_Screen.DELETE.activated[str].connect(self.delete_recording)
211 
212         self.stack = QStackedWidget(self)
213         self.stack.addWidget(self.Home_Screen)
214 
215         self.stack.addWidget(self.Empty_Home_Screen)
216 
217         self.stack.addWidget( self.Play_Screen )
218         self.setCentralWidget( self.stack )
219         #checks if there are existing recordings
220         cwd = music_utils.os.getcwd()
221         existing_files = music_utils.os.listdir(cwd + '\Recordings')
222 
223         if len(existing_files) == 0:
224             self.setWindowTitle("Music Application / Home")
225             self.stack.setCurrentIndex( 1 )
226         else:
227             self.setWindowTitle("Music Application / Home")
228             self.stack.setCurrentIndex(0)
229 
230 
    def home_screen(self):
```

```
231     def home_screen(self):
232         #checks if there are existing recordings
233         cwd = music_utils.os.getcwd()
234         existing_files = music_utils.os.listdir(cwd + '\Recordings')
235         if len(existing_files) == 0:
236             self.setWindowTitle("Music Application / Home")
237             self.stack.setCurrentIndex( 1 )
238         else:
239             self.setWindowTitle("Music Application / Home")
240             self.stack.setCurrentIndex(0)
241             self.Empty_Home_Screen.PLAYSCREEN.setEnabled(False)
242             self.Empty_Home_Screen.PLAYSCREEN.setVisible(False)
243 
244     def play_screen(self):
245         self.setWindowTitle("Music Application / Play")
246         self.stack.setCurrentIndex( 2 )
247 
248 
249     def _play(self, _file):  #play audio
250         self.Play_Screen.RECORDINGS.setEnabled(False)
251         self.Play_Screen.DELETE.setEnabled(False)
252         self.Home_Screen.RECORD.setEnabled(False)
253         self.Empty_Home_Screen.RECORD.setEnabled(False)
254 
255         e,B,G,D,A,E = music_utils.get_tab(_file)
256 
257         if len(e) < 81:
258             self.Play_Screen.SMALLE.setGeometry(50,50,700,50)
259             self.Play_Screen.SMALLE.setText(e)
260             self.Play_Screen.B.setGeometry(50,65,700,50)
261             self.Play_Screen.B.setText(B)
262             self.Play_Screen.G.setGeometry(50,80,700,50)
263             self.Play_Screen.G.setText(G)
264             self.Play_Screen.D.setGeometry(50,95,700,50)
265             self.Play_Screen.D.setText(D)
266             self.Play_Screen.A.setGeometry(50,110,700,50)
267             self.Play_Screen.A.setText(A)
268             self.Play_Screen.E.setGeometry(50,125,700,50)
                self.Play_Screen.E.setText(E)
```

```
271             self.Play_Screen.SMALLE2.setText('')
272             self.Play_Screen.B2.setText('')
273             self.Play_Screen.G2.setText('')
274             self.Play_Screen.D2.setText('')
275             self.Play_Screen.A2.setText('')
276             self.Play_Screen.E2.setText('')
277 
278             music_utils.play(_file)
279 
280             self.Play_Screen.RECORDINGS.setEnabled(True)
281             self.Play_Screen.DELETE.setEnabled(True)
282             self.Home_Screen.RECORD.setEnabled(True)
283             self.Empty_Home_Screen.RECORD.setEnabled(True)
284         elif len(e) > 80:
285             half_way_point = len(e) / 2
286             half_way_point = int(half_way_point)
287 
288             self.Play_Screen.SMALLE.setGeometry(50,50,700,50)
289             self.Play_Screen.SMALLE.setText(e[:half_way_point])
290             self.Play_Screen.B.setGeometry(50,65,700,50)
291             self.Play_Screen.B.setText(B[:half_way_point])
292             self.Play_Screen.G.setGeometry(50,80,700,50)
293             self.Play_Screen.G.setText(G[:half_way_point])
294             self.Play_Screen.D.setGeometry(50,95,700,50)
295             self.Play_Screen.D.setText(D[:half_way_point])
296             self.Play_Screen.A.setGeometry(50,110,700,50)
297             self.Play_Screen.A.setText(A[:half_way_point])
298             self.Play_Screen.E.setGeometry(50,125,700,50)
299             self.Play_Screen.E.setText(E[:half_way_point])
300 
301             self.Play_Screen.SMALLE2.setGeometry(50,150,700,50)
302             self.Play_Screen.SMALLE2.setText(e[half_way_point:])
303             self.Play_Screen.B2.setGeometry(50,165,700,50)
304             self.Play_Screen.B2.setText(B[half_way_point:])
305             self.Play_Screen.G2.setGeometry(50,180,700,50)
306             self.Play_Screen.G2.setText(G[half_way_point:])
307             self.Play_Screen.D2.setGeometry(50,195,700,50)
308             self.Play_Screen.D2.setText(D[half_way_point:])
                self.Play_Screen.A2.setGeometry(50,210,700,50)
```

```
309             self.Play_Screen.A2.setGeometry(50,210,700,50)
310             self.Play_Screen.A2.setText(A[half_way_point:])
311             self.Play_Screen.E2.setGeometry(50,225,700,50)
312             self.Play_Screen.E2.setText(E[half_way_point:])
313 
314             music_utils.play(_file)
315 
316             self.Play_Screen.RECORDINGS.setEnabled(True)
317             self.Play_Screen.DELETE.setEnabled(True)
318             self.Home_Screen.RECORD.setEnabled(True)
319             self.Empty_Home_Screen.RECORD.setEnabled(True)
320 
321     def getText(self):
322         text, okPressed = QInputDialog.getText(self, "Recording name","Enter name of recording:", QLineEc
323         if okPressed:
324             return text
325         else:
326             self.Home_Screen.RECORDED_LBL.setText('')
327             self.Empty_Home_Screen.RECORDED_LBL.setText('')
328             self.Play_Screen.RECORDINGS.setEnabled(True)
329             self.Play_Screen.DELETE.setEnabled(True)
330             self.Home_Screen.RECORD.setEnabled(True)
331             self.Empty_Home_Screen.RECORD.setEnabled(True)
332             self.Empty_Home_Screen.PLAYSCREEN.setEnabled(True)
333             self.Empty_Home_Screen.PLAYSCREEN.setVisible(True)
334 
335     def getTextError(self):
336         text, okPressed = QInputDialog.getText(self, "ERROR!","Enter different name:", QLineEdit.Normal,
337         if okPressed:
338             return text
339         else:
340             self.Home_Screen.RECORDED_LBL.setText('')
341             self.Empty_Home_Screen.RECORDED_LBL.setText('')
342             self.Play_Screen.RECORDINGS.setEnabled(True)
343             self.Play_Screen.DELETE.setEnabled(True)
344             self.Home_Screen.RECORD.setEnabled(True)
345             self.Empty_Home_Screen.RECORD.setEnabled(True)
346             self.Empty_Home_Screen.PLAYSCREEN.setEnabled(True)
```

```
346             self.Empty_Home_Screen.PLAYSCREEN.setEnabled(True)
347             self.Empty_Home_Screen.PLAYSCREEN.setVisible(True)
348
349     def _delete(self, _file):    #deletes audio files
350         music_utils.delete(_file)
351         idx = self.Play_Screen.DELETE.currentIndex()
352         self.Play_Screen.RECORDINGS.removeItem(idx)
353         self.Play_Screen.DELETE.removeItem(idx)
354
355     def play_audio(self, _file):  #creates thread for playing audio files
356         worker = Worker(self._play, _file)
357         self.threadpool.start(worker)
358
359     def record(self): #creates thread for recording
360
361         if self.stack.currentIndex() == 0:
362             sec = self.Home_Screen.RECORDING_TIME.currentText()
363         else:
364             sec = self.Empty_Home_Screen.RECORDING_TIME.currentText()
365
366         self.Home_Screen.RECORDED_LBL.setText('RECORDED!')
367         self.Empty_Home_Screen.RECORDED_LBL.setText('RECORDED!')
368         self.Play_Screen.RECORDINGS.setEnabled(False)
369         self.Play_Screen.DELETE.setEnabled(False)
370         self.Home_Screen.RECORD.setEnabled(False)
371         self.Empty_Home_Screen.RECORD.setEnabled(False)
372
373         notes, norm_len, r, p, stream = music_utils.record(int(sec))
374
375         self.text = self.getText()
376         self.text = self.text  + '.wav'
377
378         cwd = music_utils.os.getcwd()
379         existing_files = music_utils.os.listdir(cwd + "\Recordings")
380         if self.text in existing_files or self.text == '.wav' or len(self.text) > 20:
381             while self.text in existing_files or self.text == '.wav' or len(self.text) > 20:
382                 self.text = self.getTextError()
383                 self.text = self.text  + '.wav'
```

```
382                 self.text = self.getTextError()
383                 self.text = self.text  + '.wav'
384             music_utils.create_file(self.text, notes, norm_len, r, p, stream)
385 
386             self.Home_Screen.RECORDED_LBL.setText('')
387             self.Empty_Home_Screen.RECORDED_LBL.setText('')
388             self.Play_Screen.RECORDINGS.setEnabled(True)
389             self.Play_Screen.DELETE.setEnabled(True)
390             self.Home_Screen.RECORD.setEnabled(True)
391             self.Empty_Home_Screen.RECORD.setEnabled(True)
392             self.Empty_Home_Screen.PLAYSCREEN.setEnabled(True)
393             self.Empty_Home_Screen.PLAYSCREEN.setVisible(True)
394 
395             self.Play_Screen.RECORDINGS.addItem(self.text)
396             self.Play_Screen.DELETE.addItem(self.text)
397         else:
398             music_utils.create_file(self.text, notes, norm_len, r, p, stream)
399 
400             self.Home_Screen.RECORDED_LBL.setText('')
401             self.Empty_Home_Screen.RECORDED_LBL.setText('')
402             self.Play_Screen.RECORDINGS.setEnabled(True)
403             self.Play_Screen.DELETE.setEnabled(True)
404             self.Home_Screen.RECORD.setEnabled(True)
405             self.Empty_Home_Screen.RECORD.setEnabled(True)
406             self.Empty_Home_Screen.PLAYSCREEN.setEnabled(True)
407             self.Empty_Home_Screen.PLAYSCREEN.setVisible(True)
408 
409             self.Play_Screen.RECORDINGS.addItem(self.text)
410             self.Play_Screen.DELETE.addItem(self.text)
411 
412     def delete_recording(self, _file): #creates thread for deleting files
413 
414         choice = QMessageBox.question(self, 'Delete?',
415                                       "Are you sure to delete?",
416                                       QMessageBox.Yes | QMessageBox.No)
417 
418         if choice == QMessageBox.Yes:
419             worker = Worker(self, delete, file)
```

```
410             self.Play_Screen.DELETE.addItem(self.text)
411
412     def delete_recording(self, _file): #creates thread for deleting files
413
414         choice = QMessageBox.question(self, 'Delete?',
415                                       "Are you sure to delete?",
416                                       QMessageBox.Yes | QMessageBox.No)
417
418         if choice == QMessageBox.Yes:
419             worker = Worker(self._delete, _file)
420             self.threadpool.start(worker)
421         elif choice == QMessageBox.No:
422             pass
423
424     def quit_app(self): #exits out of the app
425
426         choice = QMessageBox.question(self, 'Quit?',
427                                       "Are you sure to quit?",
428                                       QMessageBox.Yes | QMessageBox.No)
429
430         if choice == QMessageBox.Yes:
431             print("Closing App!")
432             sys.exit()
433         elif choice == QMessageBox.No:
434             pass
435
436
437 if __name__ == '__main__':
438     music_utils.check_tab()
439     if not QApplication.instance():
440         app = QApplication(sys.argv)
441     else:
442         app = QApplication.instance()
443
444     w = MainWindow()
445     w.show()
446     sys.exit(app.exec_())
```