CSCI 3280 Progress Report

Member:
Lee Ho Kong 1155149106
CHEUNG Ka Ho 1155158622
Fung Ngai Man 1155158312
Chan Shi Leung Jonathan 1155142863

Objective

We use C++ to decode the music and Python to develop a music player that allows the user to select and play music files. The music player uses the Tkinter library to create a user interface for the music player.

Overview

We developed the user interface and some basic functions for the music player. User can use the play/stop button and a list control of audio files.

The code starts by importing the necessary libraries: tkinter for the user interface, fnmatch and os to work with files and directories, and pygame to play music. It then sets up the user interface by creating a canvas.

```python
import tkinter as tk
import fnmatch
import os
from pygame import mixer
```

We also defined functions for each button action.

1. The select() function is called when the user clicks the play button, and it plays the selected song using the Pygame mixer.
2. The stop() function is called when the user clicks the stop button, and it stops the currently playing song.
3. The play_next() function is called when the user clicks the next button, and it plays the next song in the playlist.
4. The play_prev() function is called when the user clicks the previous button, and it plays the previous song in the playlist.
5. The pause_song() function is called when the user clicks the pause button, and it pauses or unpauses the currently playing song.

```python
#action for the play button: to select a song to be play
def select():
    label.config(text = listBox.get("anchor"))
    mixer.music.load(rootpath + "\\" + listBox.get("anchor"))
    mixer.music.play()
#action for the stop button: to clear a song when it is activated
def stop():
    mixer.music.stop()
    listBox.select_clear('active')
#action for the next button: to select the next song, by adding 1 to 
the current playing song
def play_next():
    next_song = list.curselection()
    next_song = next_song[0] + 1
    next_song_name = listBox.get(next_song)
    label.config(text = next_song_name)
    mixer.music.load(rootpath + "\\" + next_song_name)
    mixer.music.play()
    listBox.select_clear(0, 'end')
    listBox.activate(next_song)
    listBox.select_set(next_song)
#action for the prev button: to select the previous song, by minus 1 to 
the current playing song
def play_prev():
    next_song = list.curselection()
    next_song = next_song[0] - 1
    next_song_name = listBox.get(next_song)
    label.config(text = next_song_name)
    mixer.music.load(rootpath + "\\" + next_song_name)
    mixer.music.play()
    listBox.select_clear(0, 'end')
    listBox.activate(next_song)
    listBox.select_set(next_song)
#action for the pause button: to pause the song when it is playing and 
unpuase it  when it is paused
def pause_song():
    if pauseButton["text"] == "Pause":
        mixer.music.pause()
        pauseButton["text"] == "Play"
    else:
        mixer.music.unpause()
        pauseButton["text"] == "Pause"
```

User Interface event loop is started using the canvas.mainloop() function, user can run the user interface until application window.

```
canvas.mainloop()
```

Workload distribution

Requirement	Member-InCharge	Programming Language
Basic UI and documents	Chan Shi Leung Joathan	Python
Decoding and Playback	Fung Ngai Man	C++
Music management and Searching	CHEUNG Ka Ho	Python
Information and Lyrics Display	Lee Ho Kong	Python