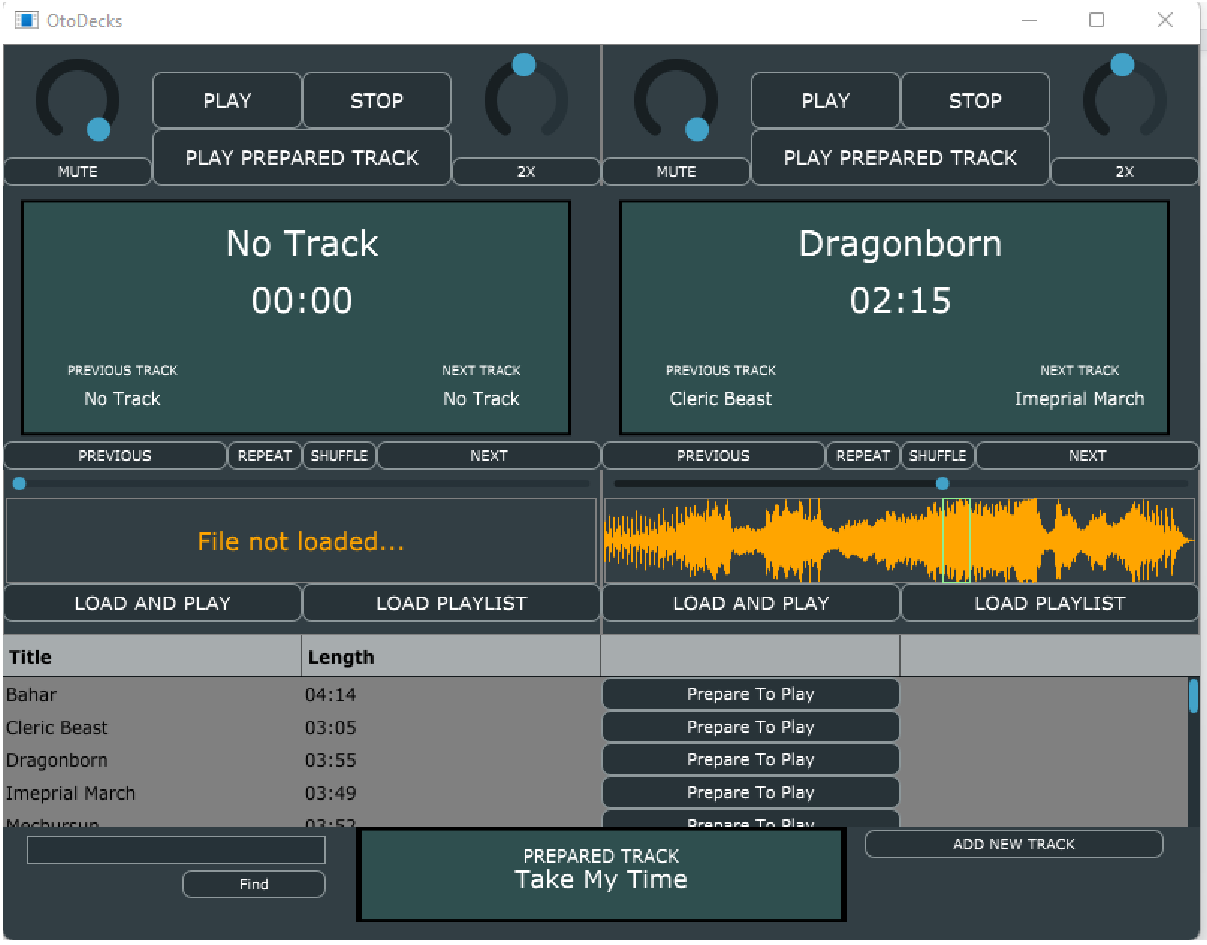
INTRODUCTION

In my version of OtoDecks, I have fully met all of the requirements and even extended the related functions. The final version of the app looks like below, which is very different than the original app in the course.



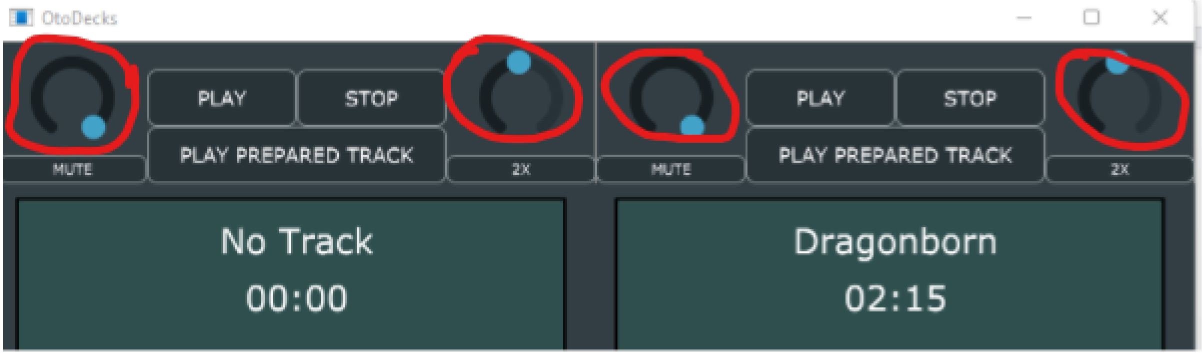
Title	Length		
Bahar	04:14	Prepare To Play	
Cleric Beast	03:05	Prepare To Play	
Dragonborn	03:55	Prepare To Play	
Imeprial March	03:49	Prepare To Play	
Mecburcun	03:52	Prepare To Play	

REQUIREMENTS

R1

The application is fully capable of every basic functionalities shown in the class. It can load files into the audio players and it can play two tracks simultaneously, as it has two different decks and players. It can directly load and play tracks from ‘load and play’, can load tracks from ‘drag and drop’, load tracks from playlist or the user has a chance to prepare a song from the playlist and play it from the ‘PLAY PREPARED TRACK’ button.

Both decks have different speed and volume sliders, so it is also capable of mixing tracks with different volumes and speeds.



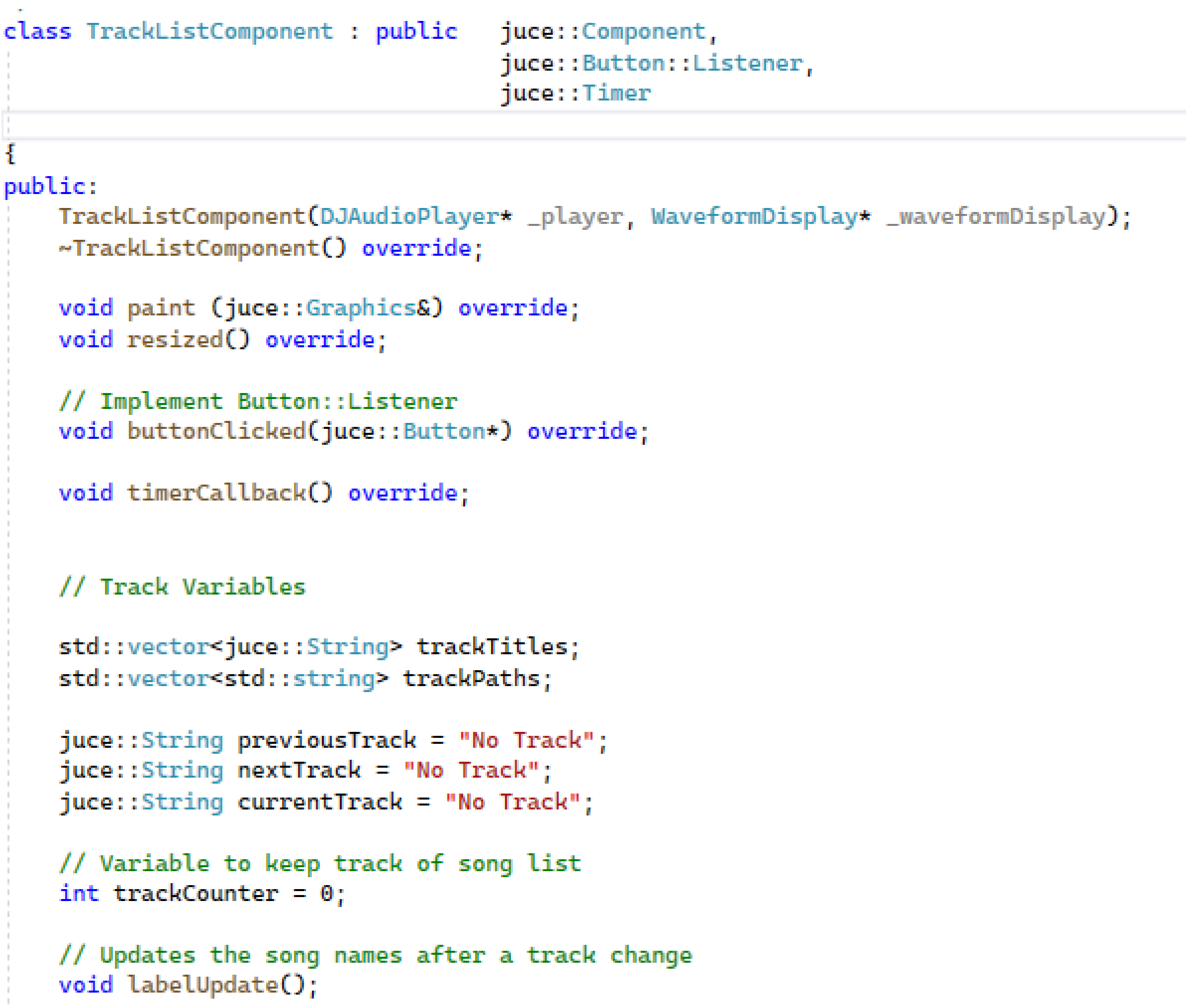
R2

The application has many extra controls and features. It also has another standalone component, ‘TrackListComponent’. It controls several control buttons, which are ‘Previous’, ‘Next’, ‘Shuffle’ and ‘Repeat’. It has a big screen which shows the playing tracks along with next and previous tracks in the deck. It also has a timer for tracks.

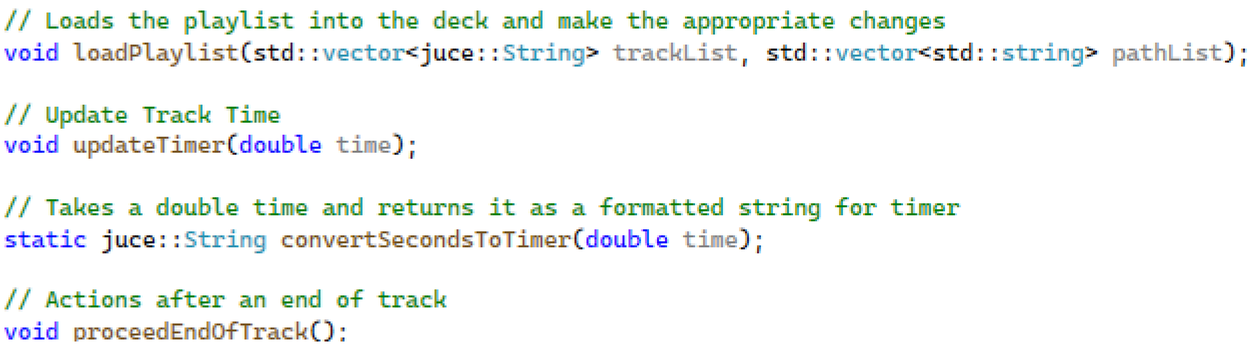
The component has two main responsibilities:

- Managing tracklists loaded in the deck.
- Being a medium for user-interaction for tracklist control.

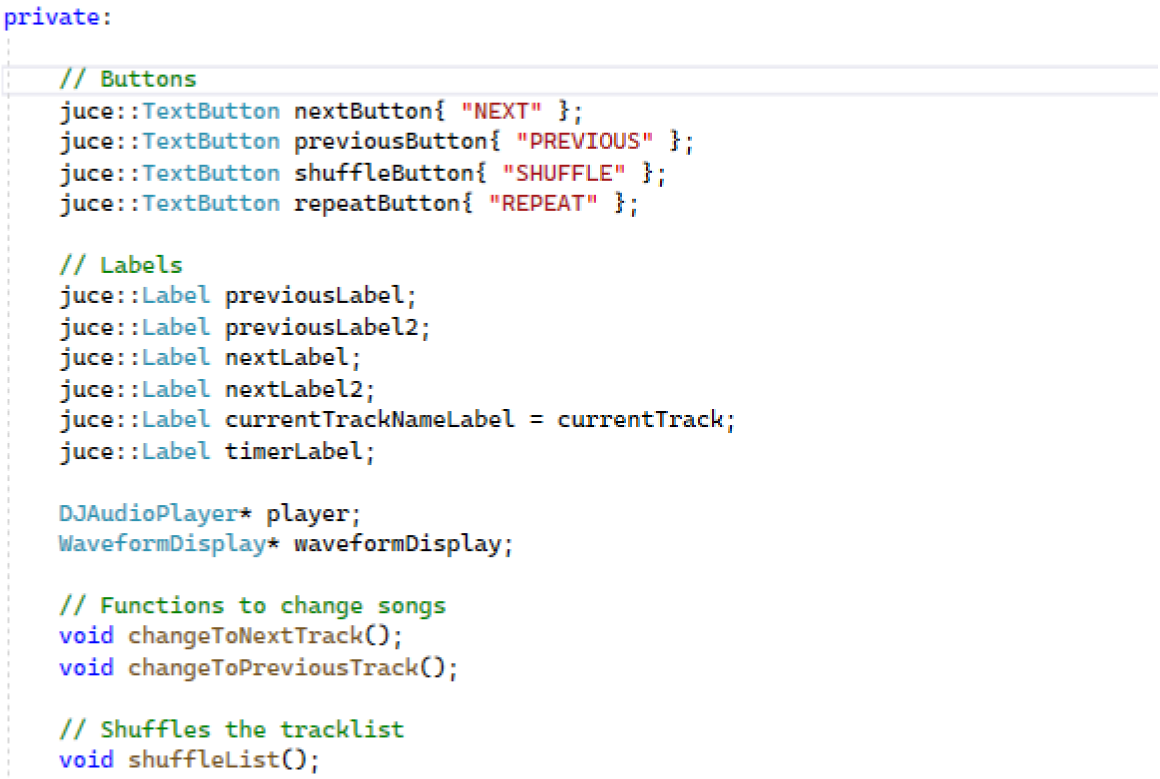
The variables and functions the tracklist controls are as below:



```
class TrackListComponent : public   juce::Component,
                                    juce::Button::Listener,
                                    juce::Timer

{
public:
    TrackListComponent(DJAudioPlayer* _player, WaveformDisplay* _waveformDisplay);
    ~TrackListComponent() override;

    void paint (juce::Graphics&) override;
    void resized() override;

    // Implement Button::Listener
    void buttonClicked(juce::Button*) override;

    void timerCallback() override;


    // Track Variables

    std::vector<juce::String> trackTitles;
    std::vector<std::string> trackPaths;

    juce::String previousTrack = "No Track";
    juce::String nextTrack = "No Track";
    juce::String currentTrack = "No Track";

    // Variable to keep track of song list
    int trackCounter = 0;

    // Updates the song names after a track change
    void labelUpdate();
```



```
// Loads the playlist into the deck and make the appropriate changes
void loadPlaylist(std::vector<juce::String> trackList, std::vector<std::string> pathList);

// Update Track Time
void updateTimer(double time);

// Takes a double time and returns it as a formatted string for timer
static juce::String convertSecondsToTimer(double time);

// Actions after an end of track
void proceedEndOfTrack();
```



```
private:

    // Buttons
    juce::TextButton nextButton{ "NEXT" };
    juce::TextButton previousButton{ "PREVIOUS" };
    juce::TextButton shuffleButton{ "SHUFFLE" };
    juce::TextButton repeatButton{ "REPEAT" };

    // Labels
    juce::Label previousLabel;
    juce::Label previousLabel2;
    juce::Label nextLabel;
    juce::Label nextLabel2;
    juce::Label currentTrackNameLabel = currentTrack;
    juce::Label timerLabel;

    DJAudioPlayer* player;
    WaveformDisplay* waveformDisplay;

    // Functions to change songs
    void changeToNextTrack();
    void changeToPreviousTrack();

    // Shuffles the tracklist
    void shuffleList();
```

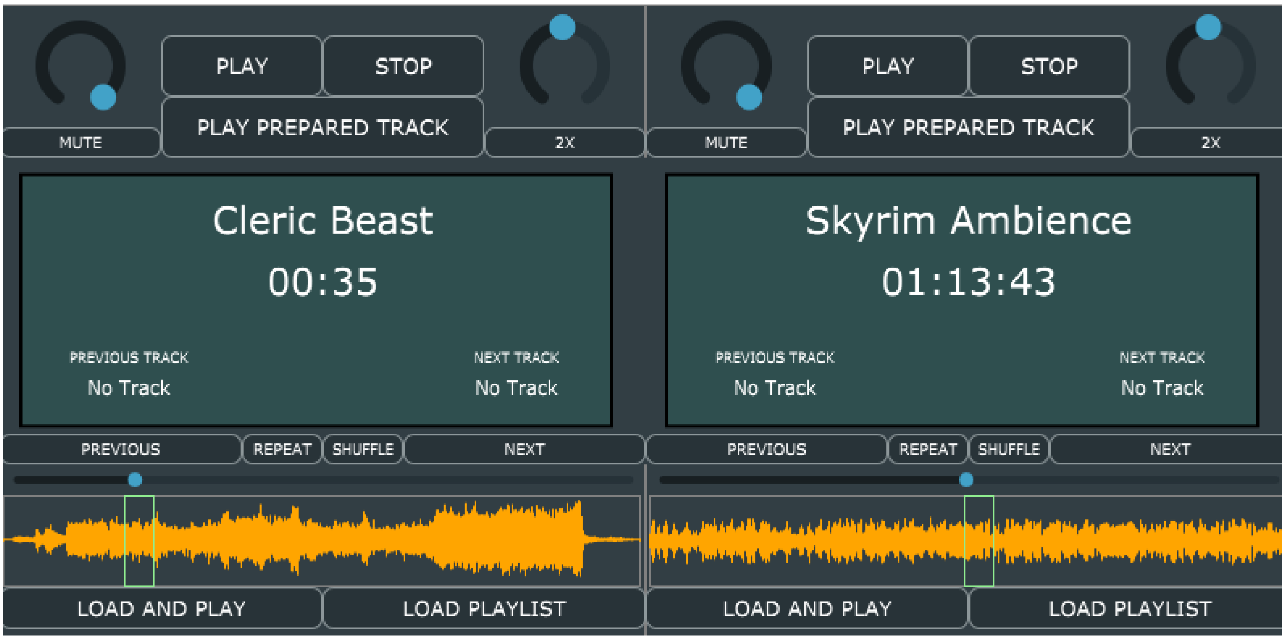
The tracklist keeps two vectors, trackTitles and trackPaths. These two vectors are responsible for storing tracks loaded in decks.

‘Next’ and ‘Previous’ buttons traverse between loaded tracks. ‘Shuffle’ button shuffles the tracklist loaded in the deck.

TrackListComponent class also contains features from juce::Timer class. It continuously checks if the current track is finished or not with the ‘timer callback’ function. If the track is finished, it goes to the next track. One little exception is ‘repeat’ mode, which is controlled by its fourth and last button, ‘Repeat’. While in repeat mode, when the track is finished, the same track starts from the beginning. However, if the user manually clicks ‘Next’ or ‘Previous’, the track will change.

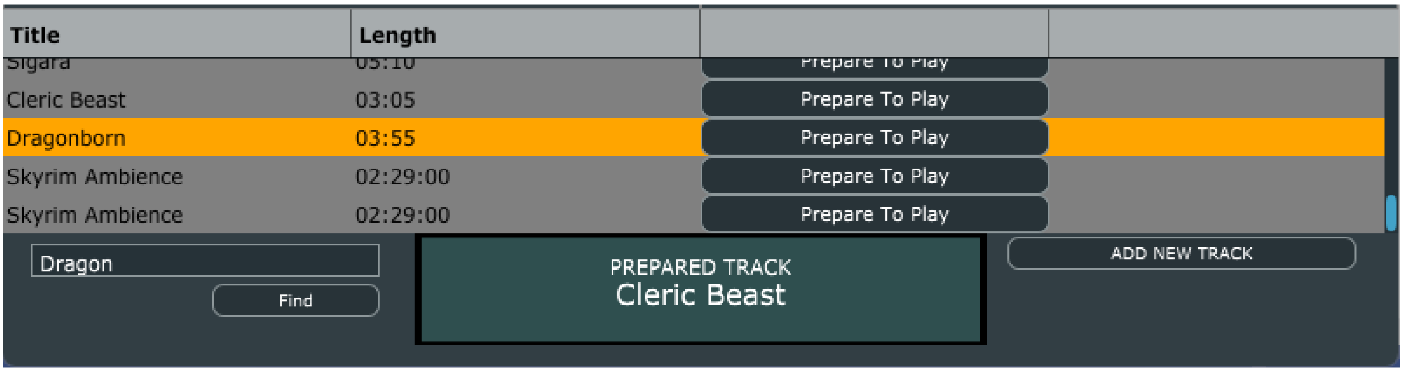
The screen in the component shows four data: Current track, the timer, next track, previous Track.

The track data comes from the trackTitles vector. The component also synchronizes the trackTitles and trackPaths vectors after shuffling. The timer is received as raw seconds. It also has a function to convert raw seconds to a formatted hh:mm:ss string format. If the track is shorter than an hour, it shows it as mm:ss.



R3

The library of the application is controlled by the playlistComponent class.



Title	Length		
Sigara	05:10	Prepare To Play	
Cleric Beast	03:05	Prepare To Play	
Dragonborn	03:55	Prepare To Play	
Skyrim Ambience	02:29:00	Prepare To Play	
Skyrim Ambience	02:29:00	Prepare To Play	

Dragon

PREPARED TRACK
Cleric Beast

The component lets users add new tracks from the ‘ADD NEW TRACK’ button. It shows the metadata of the tracks by their name and their length.

The component also has a search box and the ‘Find’ button. The user is able to quickly find tracks inside the library using this feature. The clicking button is also smart. If the user is searching for the same thing, it first shows the first found track. If the user keeps searching, it will find the next tracks containing the text the user entered.

There are two methods the user can load tracks into the decks from library:

- Loading the whole library
- Preparing a track and playing whenever the user wants.

Loading the whole library happens when the user clicks the 'LOAD PLAYLIST’ button. If the user clicks the 'Prepare To Play’ button, the track will be ready to be loaded into the deck.

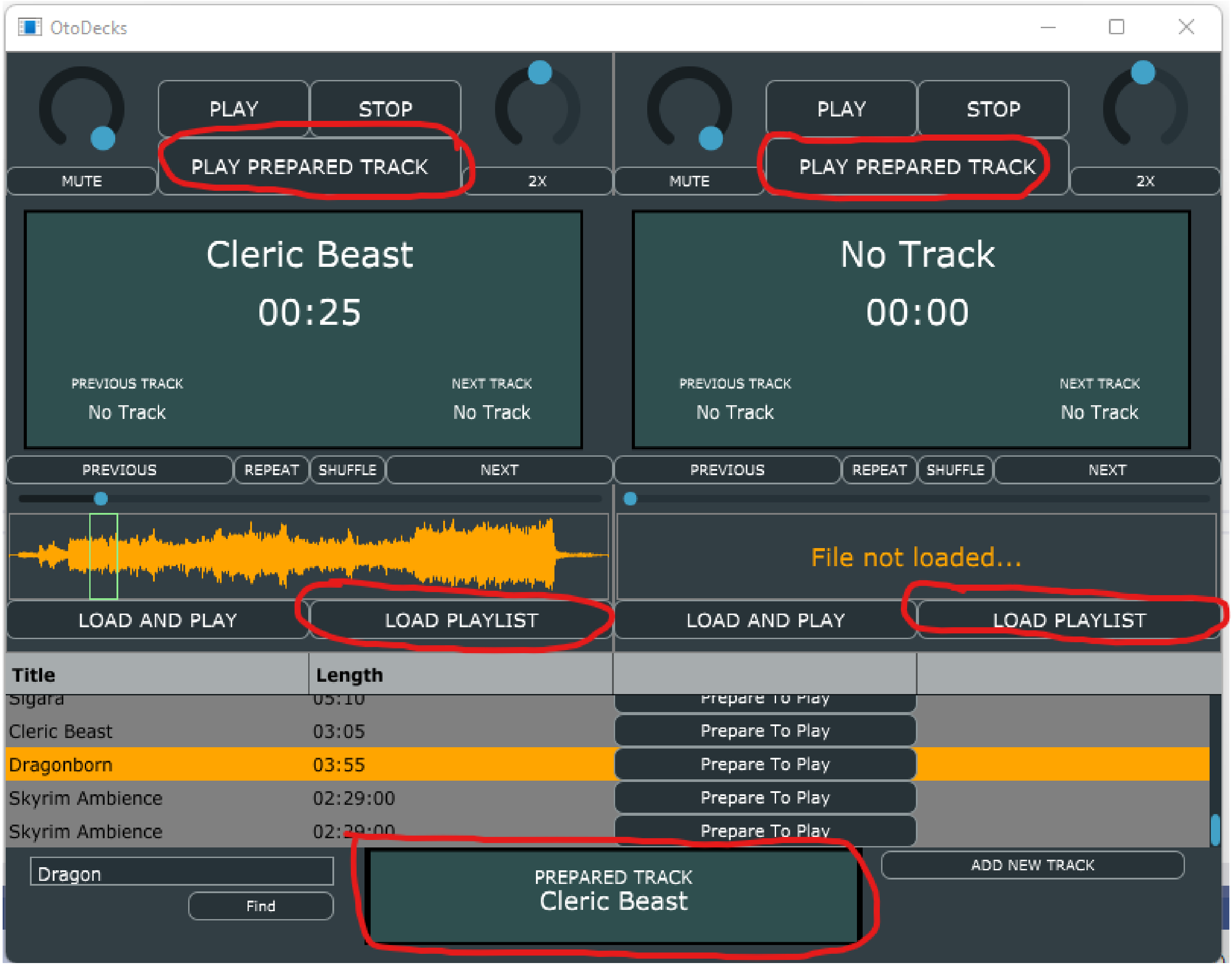
As it is a DJ application, the tracks do not start immediately, but the user decides when to. If a whole library is loaded into the deck, two cases may happen:

1. There is no current track playing
2. There is a track playing

If there is no track playing in the deck, the user can simply start the track by pressing the ‘PLAY’ button. However, if there is a track playing, then either the track will start after the current track is finished, or the user needs to press the ‘Next’ or ‘Previous’ button.

If the user chooses to play a single track, the user prepares a track to be played by the ‘Prepare To Play’ button. Then, a screen under the library table shows which track is ready to play. When the user clicks the 'PLAY PREPARED TRACK’ button, the track will start playing.

The library also persists using a .csv file, called Tracks.csv. The CSVOperator class controls the operations about the csv file. The tracks are kept as file paths. When the application is started, it immediately restores the latest library.

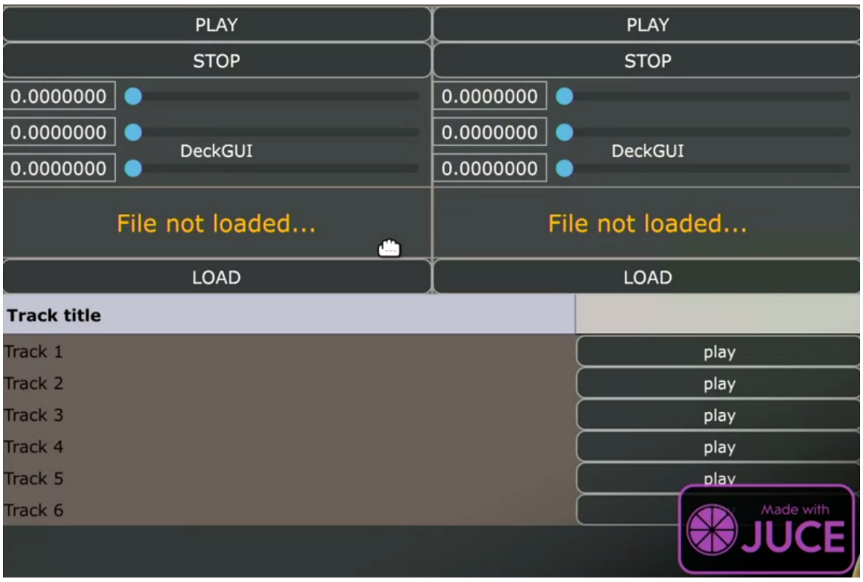


Title	Length		
Sigara	05:10	Prepare To Play	
Cleric Beast	03:05	Prepare To Play	
Dragonborn	03:55	Prepare To Play	
Skyrim Ambience	02:29:00	Prepare To Play	
Skyrim Ambience	02:29:00	Prepare To Play	

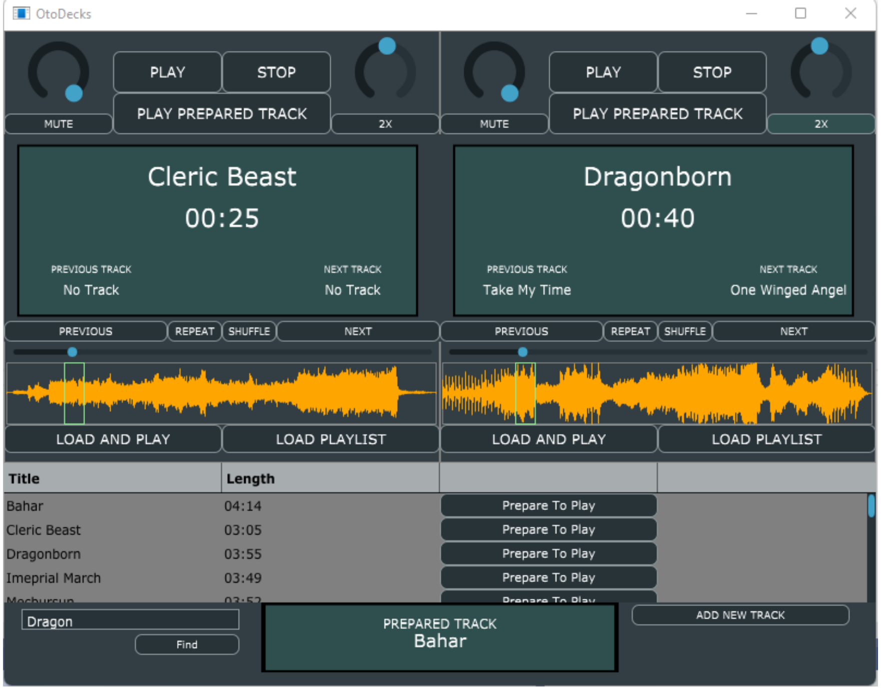
```
C:\Users\oktay\OneDrive\MasaÃ¼stÃ¼\playlist\Bahar.mp3
C:\Users\oktay\OneDrive\MasaÃ¼stÃ¼\playlist\Cleric Beast.mp3
C:\Users\oktay\OneDrive\MasaÃ¼stÃ¼\playlist\Dragonborn.mp3
C:\Users\oktay\OneDrive\MasaÃ¼stÃ¼\playlist\Imeprial March.mp3
C:\Users\oktay\OneDrive\MasaÃ¼stÃ¼\playlist\Mecbursun.mp3
C:\Users\oktay\OneDrive\MasaÃ¼stÃ¼\playlist\One Winged
Angel.mp3
C:\Users\oktay\OneDrive\MasaÃ¼stÃ¼\playlist\One Winged
Angel.mp3
C:\Users\oktay\OneDrive\MasaÃ¼stÃ¼\playlist\Sigara.mp3
C:\Users\oktay\OneDrive\MasaÃ¼stÃ¼\playlist\Take My Time.mp3
C:\Users\oktay\OneDrive\MasaÃ¼stÃ¼\playlist\Cleric Beast.mp3
C:\Users\oktay\OneDrive\MasaÃ¼stÃ¼\playlist\Take My Time.mp3
C:\Users\oktay\OneDrive\MasaÃ¼stÃ¼\playlist\One Winged
Angel.mp3
C:\Users\oktay\OneDrive\MasaÃ¼stÃ¼\playlist\Imeprial March.mp3
C:\Users\oktay\OneDrive\MasaÃ¼stÃ¼\playlist\Take My Time.mp3
C:\Users\oktay\OneDrive\MasaÃ¼stÃ¼\playlist\Mecbursun.mp3
C:\Users\oktay\OneDrive\MasaÃ¼stÃ¼\playlist\Sigara.mp3
C:\Users\oktay\OneDrive\MasaÃ¼stÃ¼\playlist\Cleric Beast.mp3
C:\Users\oktay\OneDrive\MasaÃ¼stÃ¼\playlist\Dragonborn.mp3
C:\Users\oktay\OneDrive\MasaÃ¼stÃ¼\playlist\Skyrim
Ambience.mp3
C:\Users\oktay\OneDrive\MasaÃ¼stÃ¼\playlist\Skyrim
Ambience.mp3
```

R4

The latest GUI is very different from the original one and has many extra controls. The comparison of the applications are shown below:



Track title	
Track 1	play
Track 2	play
Track 3	play
Track 4	play
Track 5	play
Track 6	



Title	Length		
Bahar	04:14	Prepare To Play	
Cleric Beast	03:05	Prepare To Play	
Dragonborn	03:55	Prepare To Play	
Imeprial March	03:49	Prepare To Play	
Mechurcun	03:52	Prepare To Play	

The new GUI has many new controls and buttons. A completely new component, TrackListComponent, is added to the application. Library is extended with many new controls and still persists. Beyond the buttons explained in other sections and components, the new DeckGUI has two new buttons, 'MUTE' button and '2X' button, which even furthers the user's control over speed and volume.

The DeckGUI class also controls 'LOAD AND PLAY' button (which is the same LOAD button from the original app), 'LOAD PLAYLIST' button (explained in R3) and 'PLAY PREPARED TRACK' button (explained in R3 again).