JUKE BOX: A MUSIC APP

Thang Pham Duc

October 2024

Contents

1 Introduction

1.1 Background

The aim of the project, Music Application, is to provide a graphical user interface. The interface manages the music tracks and plays them. These days, with ever-growing dependency on digital media, the need has essentially arisen for applications that will let users manage their personal libraries of music efficiently. The target of the project is an application that enables the addition and removal of tracks and searching of tracks as well as handling playlists.
This application development will be useful for enhancing the user experience in managing music collections so that at least the basic functionality regarding track playing and playlist management is as straightforward as possible. Thus, this project provides an opportunity to put to practice some of the concepts learned within the course, most especially those on Object-Oriented Programming.

2 Purpose and Objectives

2.1 Purpose

This project will be to design a music application that will enable users to manage their personal music collections efficiently. It is supposed to offer an interface for adding, searching, and playing music tracks while allowing for the creation and management of playlists. To this end, I try applying the concepts of programming and skills learned during my coursework in implementing this application.

2.2 Objectives

- **Track Management**: Enable users to add, remove, and search for tracks in their music library.
- **Playlist Creation**: Allow users to create and manage playlists that can include multiple tracks.
- **Data Handling**: Implement functionality to store and retrieve track information from CSV files, ensuring data persistence.
- **Playback Features**: Integrate basic playback functionality for individual tracks and entire playlists.

- **Validation**: Implement input validation to ensure that users can only enter valid data when adding or searching for tracks.

3 Project Overview

3.1 Description of the Music App

The user experience (UX) of a music app focuses on creating an intuitive, seamless, and enjoyable journey for users as they explore and listen to music. It emphasizes ease of navigation, ensuring users can effortlessly search, play, and discover new tracks or playlists. A well-designed UX provides features like add, remove, update tracks, and simple yet effective controls that respond quickly to user input. The goal is to enhance the overall experience, making the app feel effortless and engaging, without the user needing to think about the interface itself.

3.2 Key Features and Functionalities

The design focuses on creating an intuitive layout that allows users to manage their personal music collections effectively. Key features include:

- **Track Management:** Users can easily add, remove, and search for tracks within their music library.
- **Playlist Creation:** The app enables users to create and manage playlists, facilitating personalized music experiences.
- **Playback Functionality:** Users can play individual tracks, enhancing their listening experience.

4 Target Audience

By offering a familiar environment and straightforward functionality, this music application aims to cater to users looking for an efficient way to enjoy and organize their music collections.

4.1 Design and Development

4.1.1 Track Player Home

The app's main interface is the track player screen, labeled "Juke Box Music App" This label has 5 buttons: "View Tracks," "Create Track List," "Update

Tracks," and "Exit."

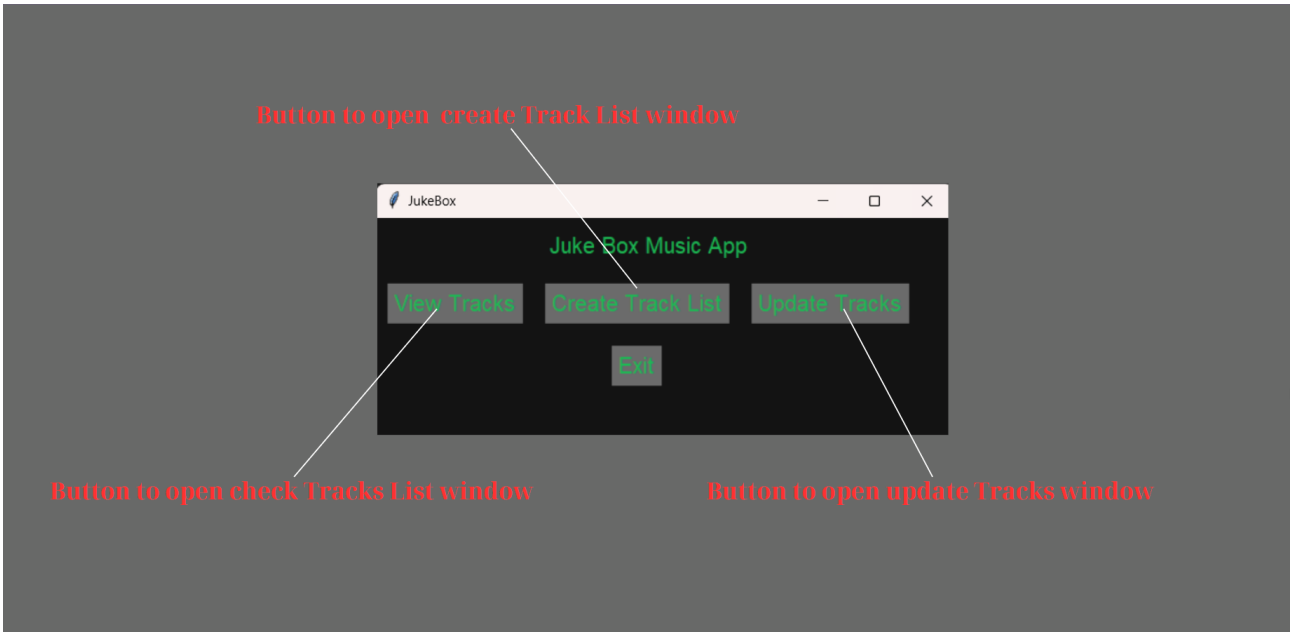

Figure 1: Track Player Home

4.2 View Track

When the 'View Tracks' button on the player interface is clicked, a 'Check Tracks' window opens. The top toolbar's 'List All Tracks' button displays a scrolling list of all available tracks, making it easy to browse through multiple options. Beside this button, an 'Enter Track Number' field allows users to enter a track ID and press 'View Track.' The track's title, artist, rating, and play count are shown in a text field on the right.

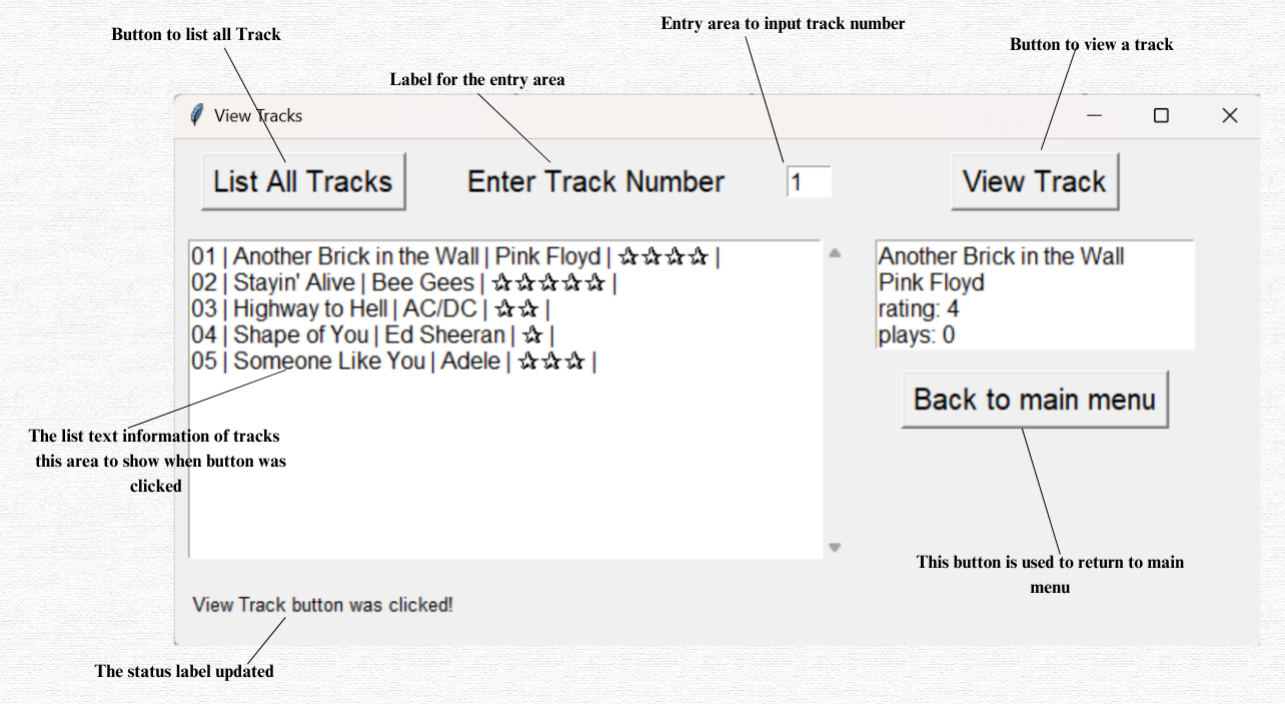

Figure 2: View Track

4.2.1 Create Track List

When the 'Create Track List' button on the main window is clicked, a "Create Track List" window opens. At the top, the 'List All Tracks' button functions like the one in the 'View Tracks' window. Next to it, the entry field allows input of a track ID, with a text area on the left showing all tracks in the list. Beside 'Entry Track Number' is the 'Add to list' button, which adds the selected track to a text area before saving it as a new list below. The 'Clear list' button removes all selected tracks if the user wants to make adjustments. Below 'Clear list' is a text area displaying the play count of selected tracks, along with a 'Play Track' button that increments the play count by 1 with each click and updates the status label at the bottom to reflect the action. Finally, the "Create Track List" section includes a label, 'Enter name for new track list,' paired with an entry field. Below are 'Create Track List' and 'Show Track Lists' buttons, which validate user input to prevent errors. 'Show Track Lists' displays all lists with their names and selected tracks in the adjacent text area. A 'Back to main menu' button mirrors the one in 'View Tracks.'

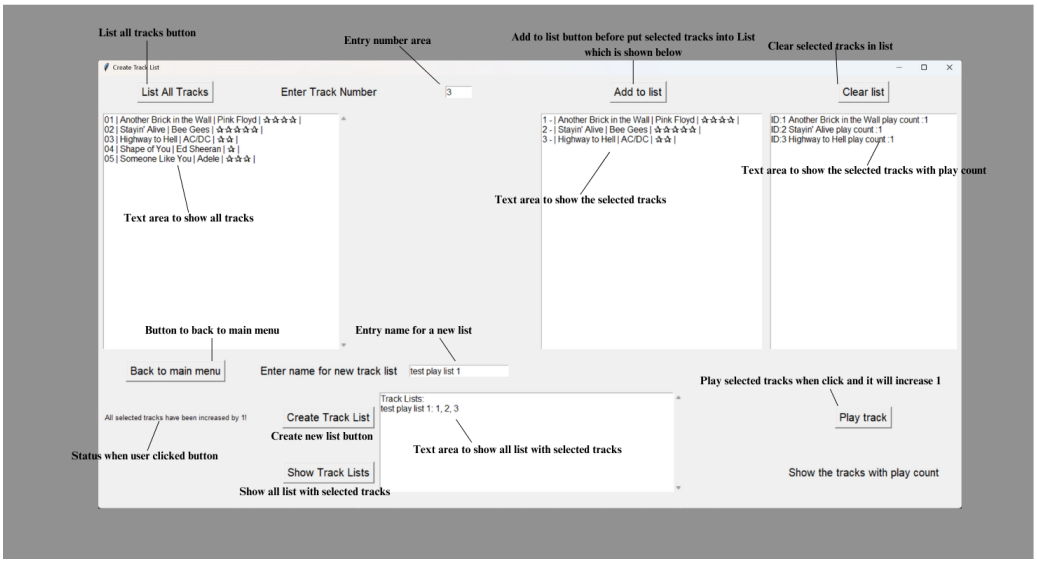

Figure 3: Create Track List

4.3 Update Tracks

When the 'Update Tracks' button on the main window is clicked, an "Update Track" window appears. From top to bottom, the layout is user-friendly, starting with an input field for entering a track ID to retrieve its information, down to the 'Back to main menu' button for easy navigation. Each field and function is labeled clearly, and a figure guides users through each step to ensure ease of use and understanding. This intuitive design makes the feature accessible to all users.

4.3.1 before change and after change

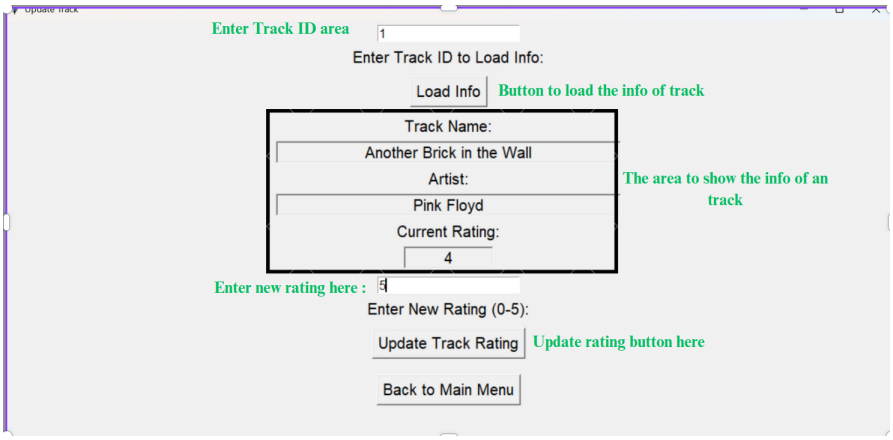

Figure 4: Before updating Tracks

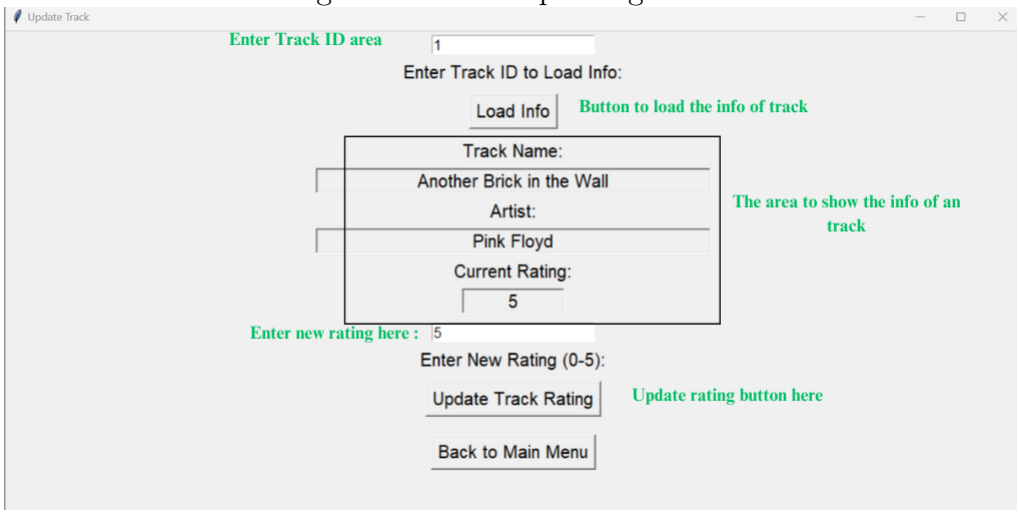

Figure 5: After updating Tracks

5 Testing and Faults

The testing is focused on core app functionality, such as adding, removing, updating, and retrieving components within the app. This includes testing all buttons, the search window, and the update window to ensure basic functions are applied effectively. Additionally, the testing emphasizes data validation, data handling, and image handling.
Most parts of the app are performing as expected, with smooth navigation and accessible features. However, certain issues were identified. For example,

when updating data in the CSV file, the data is not read efficiently, resulting in the data being completely cleared after an update. In image handling, a problem arises when the user enters a keyword in the search bar (e.g., "S"). Although the app displays information for three tracks in text format, only one image appears, corresponding to the last song in the text list.
Data validation is especially crucial for the app. A significant amount of development time was dedicated to implementing suitable validation techniques to reduce unexpected errors. Test cases have been documented in the test table to reflect this focus.
To address these issues, I researched solutions extensively on Stack Overflow and Google. I experimented with various code snippets I found, pasting them into my work and modifying them as needed until they were appropriate. I also compared different approaches to determine which solution would make the app more readable and easier to maintain.

6 Innovations, Reflection, Further Development

6.1 Innovations

- **Enhanced Search Functionality:** A new search button has been added directly to the track player menu. When users enter a keyword, track ID, or artist name and press "Enter," the app instantly displays detailed track information, including the track's image. This streamlined search experience makes it easier for users to find specific tracks and artists without interrupting their playback.

- **Track List Creation:** The "Create Track" window has been enhanced with a new feature allowing users to create custom playlists. This area lets users select tracks from their library and build a new playlist, making it easier to organize and enjoy music tailored to their preferences.

- **CSV File Integration:** Significant progress has been made in enabling the app to handle data changes with CSV files. This integration allows users to import and export track information, such as metadata and playlists, seamlessly. It enhances data management and provides flexibility in how users interact with their music library.

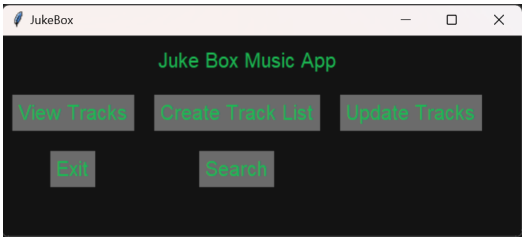

Figure 6: New features in track player window

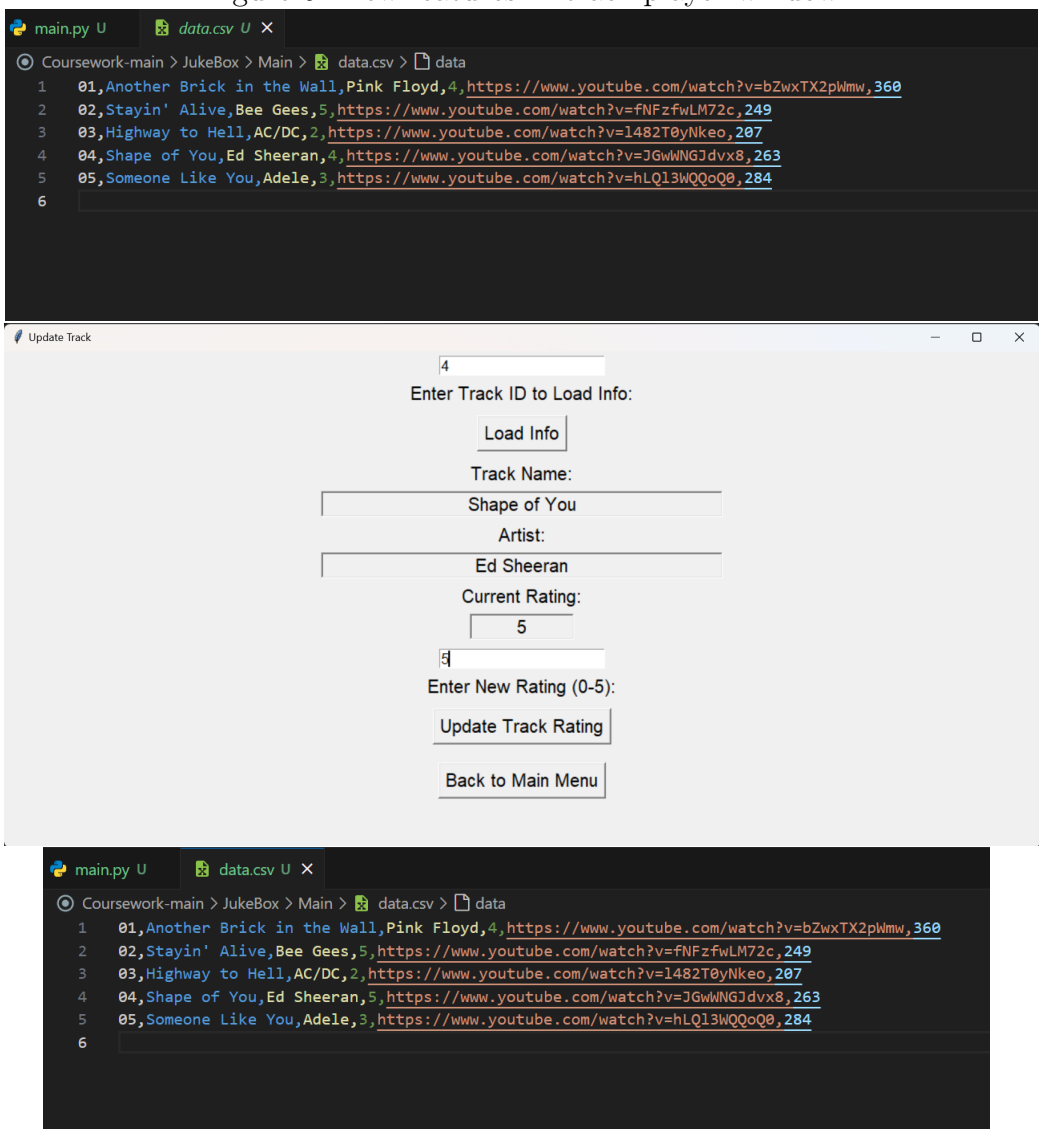

Figure 7: Integrate with Data

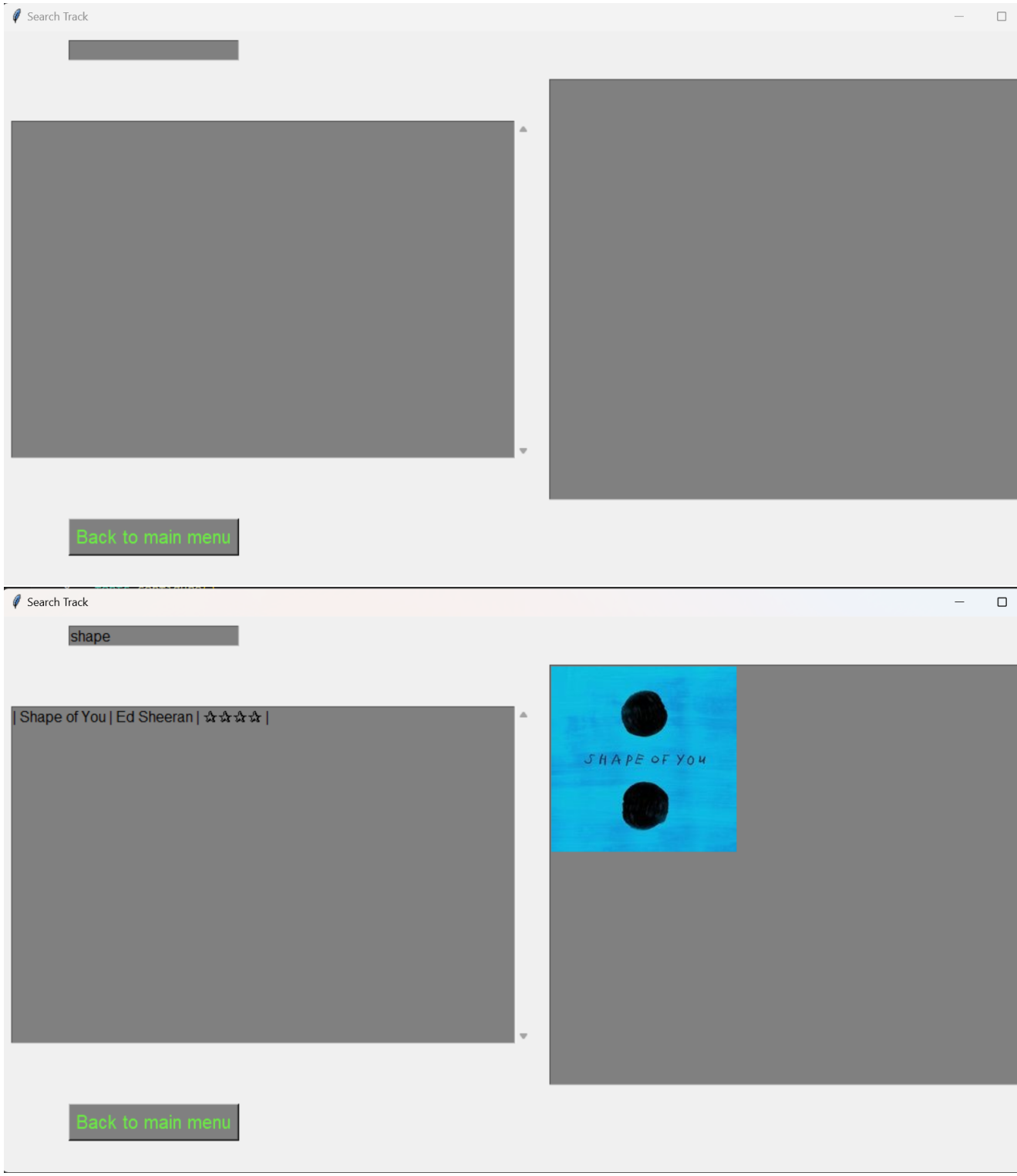

Figure 8: New search feature to show Track's info with image

6.2 Reflection

This coursework has significantly enhanced my programming skills, particularly in areas like data structures, object-oriented programming, and file handling. I've gained a deeper understanding of how to organize and manipulate data using structures like dictionaries and lists, as well as how to work with external files, such as CSV files. These techniques have helped me develop more scalable and efficient solutions.
One of the key takeaways from this coursework is the importance of user experience (UX) in application design. By adding features like a dynamic search function and playlist creation in the music app, I learned how crucial it is to consider usability alongside functionality. This has helped me think from the user's perspective and create more intuitive software.
Additionally, I've improved my debugging and testing skills. While facing issues like handling incorrect inputs or ensuring data consistency, I learned to troubleshoot effectively and iterate on solutions. This experience emphasized the value of clean, maintainable code and proper documentation.
In the long term, the skills I've gained will be useful in both academic and professional settings. The ability to work with data, design user-friendly interfaces, and debug effectively will be valuable for future software development projects. This coursework has provided a solid foundation for more complex work in fields like web development or data science.
Overall, this coursework has improved my technical and problem-solving abilities, given me more confidence in programming, and taught me how to create software that is both functional and user-friendly. I'm eager to apply these skills to future projects and challenges.

6.3 Further Development

Looking ahead, I aim to significantly enhance this music app to make it more like Spotify, with a fully-featured, user-friendly music streaming platform. The next steps will focus on expanding the app's functionality, improving scalability, and refining the user interface.

- **User Account Management:** One of the first features I am working on will be user authentication. Users can make an account and log in, save playlists and preferences personalized. A secure, reliable login system needs to be developed since personalized experiences with Spotify are created for each user.

- **Music Streaming:** This means the app should stream music and not just track information with images. It will have an audio player

integrated into the app, one that will work on streaming such that it provides features like shuffle, repeat, and volume control for playback, among other features of Spotify. One of the main challenges to overcome is likely in the development of smooth, seamless streaming.

- **Playlist and Library Management:** Further development will involve the creation, editing, and sharing of playlists. The development of a library system will also be extended such that users save their favorite songs, albums, and playlists to interactively and dynamically operate the application.

- **Recommendation Engine:** A system of recommendations, taking into account user preferences and listening history, might considerably improve the user experience. The use of machine learning or simpler algorithms, which would employ suggestions about some songs, albums, or artists based on past activity, would make the application more interactive and personalized.

- **Search and Discovery:** While it has implemented a search function, the focus should be on extending and enhancing this feature for song, artist, album, and genre searches; further developing filtering options. There should be the creation of a discovery page that includes trending tracks, new releases, and curated playlists, which was very important in emulating Spotify's content discovery features.

- **Mobile and Multi-Device Support:** In the future, I would like to develop a mobile version of the application where users can take their music and playlists anywhere. Making sure the app works seamlessly across all devices-mobile, tablet, and desktop-will go a long way toward reaching a broader audience.

- **Integration with APIs:** The integration of third-party APIs, such as Spotify's own API or others like Last.fm, will make the application more full-bodied and give access to a big music database so that the app can boast more options with track options, bios of artists, etc., without needing to enter the data manually.

6.4 Appendices

Feature	Input	Action	Expected Output	Pass
Track Player	...	Click "View Tracks" button	View tracks window pops up	Yes
Track Player	...	"Create Track List" button pressed	Create Track List window pops up	Yes
Track Player	...	"Update Tracks" button pressed	Update Tracks window pops up	Yes
Track Player	...	"Search" button pressed	Search Track window pops up	Yes
Track Player	...	"Exit" button pressed	Close the program	Yes
View Track	...	"List All Tracks" button pressed	The status label displays "List Tracks button was clicked!"	Yes
View Track	...	"List All Tracks" button pressed	All tracks appear in the left area of window	Yes
View Track	Input a number "1"	"View Track" button pressed	The status label displays "View Track button was clicked!"	Yes
View Track	Input a number "1"	"View Track" button pressed	The info of a track will appear	Yes
View Track	Input a string "01" or " 0 3 "	"View Track" button pressed	The info of a track will appear	Yes
View Track	Input a invalid "a1" or " abc "	"View Track" button pressed	The message error box will show "Please enter valid input number!"	No
View Track	Input nothing	"View Track" button pressed	The message error box will show "Please enter valid input number!"	No
View Track	Input a invalid number outside the limited area	"View Track" button pressed	The messagebox error will show "Please select a number between 1 and 5!!"	No
View Track	Input a invalid number outside the limited area	"View Track" button pressed	The messagebo error x will show "Please select a number between 1 and 5!!"	No
View Track	...	"Back to main menu" button pressed	Back to the track player window	Yes
Create Track List	...	"List All Tracks" button pressed	The status label displays "List Tracks button was clicked!"	Yes
Create Track List	...	"Back to main menu" button pressed	Back to the track player window	Yes
Create Track List	input a valid number in boundary (1 to 5)	"Add to list" button pressed	There will have a selected track on the right side under Add to list button	Yes
Update Track	Input " 0 3 " or "01"	Click "Add to list" button	The selected track is added	Yes
Create Track List	input an invalid number out of boundary (1 to 5)	"Add to list" button pressed	The messagebox error will show "Please select a number between 1 and 5!!"	No

Table 1: Test Cases for Track Player Features

Feature	Input	Action	Expected Output	Pass
Create Track List	input a invalid " a12 " or " abc "	"Add to list" button pressed	The messagebox error will show "Please enter valid input number!"	No
Create Track List	input nothing	"Add to list" button pressed	The messagebox error will show "Please enter valid input number!"	No
Create Track List	Input selected tracks	"Clear list" button pressed	The status label will show "The playlist was cleared"	Yes
Create Track List	Input nothing	"Clear list" button pressed	The messagebox will show "There is nothing to clear!"	Yes
Create Track List	Input selected tracks into the area under "Add to list" button	Click "Play track" button	There will jump out to the web browser to open the selected tracks	Yes
Create Track List	input nothing in selected tracks area	Click "Play track" button	The messagebox will show error "There is no track to play"	Yes
Update Track	...	Click "Update Tracks" button	The Update Track window pops up	Yes
Update Track	Input valid number "1" in enter track ID	Click "Load Info" button	The info of track pops up	Yes
Update Track	Input nothing in enter track ID	Click "Load Info" button	The messagebox error will show "Please enter valid input number!"	No
Update Track	Input invalid number out of boundary in enter track ID	Click "Load Info" button	The messagebox error will show "Please select a number between 1 and 5!!"	No
Update Track	Input " 0 3 " in enter track ID	Click "Load Info" button	The info of track pops up	Yes
Update Track	Invalid input "abc", "1a"	Click "Load Info" button	The messagebox error will show "Please enter valid input number!"	No
Update Track	Input nothing	Click "Load Info" button	The messagebox error will show "Please enter valid input number!"	No
Update Track	valid input new rating(1..5) in enter new rating	Click "Update Track Rating" button	The new rating will update on the window and also the data	Yes
Update Track	invalid input out of (1..5) in enter new rating	Click "Update Track Rating" button	The messagebox error will show "Please select a number between 1 and 5!"	No
Update Track	invalid input "abc", "a1" in enter new rating	Click "Update Track Rating" button	The messagebox error will show "Please enter valid input number!"	No
Update Track	Input nothing in enter new rating	Click "Update Track Rating" button	The messagebox error will show "Please enter valid input number!" and "Please load a track by entering its ID first"	No

Table 2: Test Cases for Track Player Features

Feature	Input	Action	Expected Output	Pass
Update Track	Input "1" in ID and "6"	Click "Load info" and "Update.." button	The messagebox error will show "Please select a number between 1 and 5!"	No
Update Track	Input "1" in ID and "4"	Click "Load info" and "Update.." button	The messagebox show "Track ID 1 rating updated to 4!"	Yes
Update Track	Input nothing in ID and "4"	Click "Load info" and "Update.." button	The messagebox show "Please load a track by entering its ID first"	No
Update Track	Input "1" in ID and "4" and change ID before updating	Click "Load info" and "Update.." button	The messagebox show "Track ID (after changing) not found in the library"	No
Search	Input "Shape of You"	press Enter on keyboard	The info of track and image will be shown	Yes
Search	Input "S"	press Enter on keyboard	The info of 3 tracks and images will be shown	Yes
Search	Input nothing	press Enter on keyboard	The info of all tracks will be shown	Yes
Search	Input name of artist "Ed Sheeran"	press Enter on keyboard	The info of track by Ed Sheeran will be shown	Yes
Search	Input ID of track "01" or " 0 1 " or "1"	press Enter on keyboard	The track "Another Brick in the Wall" by "Pink Floyd" will be shown	Yes
Search	Input invalid name	press Enter on keyboard	The area show info will show "Not Found" in text area and "No matching tracks found" in image area	No
Search	Input invalid number bigger than 5 and smaller than 1	press Enter on keyboard	The area show info will show "Not Found" in text area and "No matching tracks found" in image area	No

Table 3: Test Cases for Track Player Features

```python
import tkinter as tk
from tkinter import messagebox
import tkinter.scrolledtext as tkst
from tkinter import *
from CreateTrackList import CreateTrackList
import track_library as lib
from track_library import search_Item
import font_manager as fonts


def set_text(text_area, content):           # inserts content into the text_area
    text_area.delete("1.0", tk.END)         # first the existing content is deleted
    text_area.insert(1.0, content)          # then the new content is inserted


class TrackViewer():
    def __init__(self, window):     #constructor will get the self, and window
        self.new_window = tk.Toplevel(window)
        self.new_window.geometry("750x350")    #will appear the window with width : 750 and height : 350
        self.new_window.title("View Tracks")   #this will appear in the title of the window


        list_tracks_btn = tk.Button(self.new_window, text="List All Tracks", command=self.list_tracks_clicked)   #create button List All Tracks
        list_tracks_btn.grid(row=0, column=0, padx=10, pady=10)

        enter_lbl = tk.Label(self.new_window, text="Enter Track Number")    #the name beside the box to enter track number
        enter_lbl.grid(row=0, column=1, padx=10, pady=10)

        self.input_txt = tk.Entry(self.new_window, width=3)              #the box to entry
        self.input_txt.grid(row=0, column=2, padx=10, pady=10)

        check_track_btn = tk.Button(self.new_window, text="View Track", command=self.view_tracks_clicked)        #this will call function view_tracks_clicked
        check_track_btn.grid(row=0, column=3, padx=10, pady=10)                                                  #then when calling successfully it will return the info of Ob

        #back to main menu
        back_btn = tk.Button(self.new_window, text="Back to main menu", command=self.close_window)        #this will call function view_tracks_clicked
        back_btn.grid(row=1, column=3, padx=10, pady=10)


        self.list_txt = tkst.ScrolledText(self.new_window, width=48, height=12, wrap="none") #this one will the area information of tracks
        self.list_txt.grid(row=1, column=0, columnspan=3, sticky="W", padx=10, pady=10)

        self.track_txt = tk.Text(self.new_window, width=24, height=4, wrap="none") #this one will the area information of a track
        self.track_txt.grid(row=1, column=3, sticky="NW", padx=10, pady=10)

        self.status_lbl = tk.Label(self.new_window, text="", font=("Helvetica", 10)) #this one will update the status when user press some buttons
        self.status_lbl.grid(row=2, column=0, columnspan=4, sticky="W", padx=10, pady=10)


        self.list_tracks_clicked()
        self.new_window.protocol("WM_DELETE_WINDOW", self.close_window) #this one will help the program shutdown completely when press X on right of window or Exit b
        self.window = window

    def close_window(self):
        self.new_window.destroy()  # hide this current window
        self.window.deiconify() # this will back to main window of track player

    def view_tracks_clicked(self):
        key = self.input_txt.get().replace(" ", "") #take the keyword in the area and remove spaces

        checkKeyWord = convert_input_number(key) #convert to int
        item = search_Item(checkKeyWord) #get object from library

        if item is not None:
            name = item.name
            artist = item.artist
            stars = item.stars()
            play_count = item._play_count
            track_details = f"{name}\n{artist}\nrating: {stars}\nplays: {play_count}" #f"string to show the info of a track
            set_text(self.track_txt, track_details) #set text on the area to show info
        else:
            return f"Not Found this track!" # if not found will return this f'string
        self.status_lbl.configure(text="View Track button was clicked!") #finally it will appear the status

    def list_tracks_clicked(self):      #this function will appear the status of list tracks
        track_list = lib.list_all()     #call list_all() function in track_library
        set_text(self.list_txt, track_list)
        self.status_lbl.configure(text="List Tracks button was clicked!") #update the status label


def convert_input_number(input): #function to convert
        try:
            track_number = int(input) #convert to integer

        except ValueError:
            messagebox.showerror("Input error", "Please enter a valid input number!.")
        if track_number < 0 or track_number > len(lib.library):
            messagebox.showerror("Selection Error!", f"Please select a number between 1 and {len(lib.library)}")
            return
        return track_number

```


Figure 9: Stage 1