A screenshot of a computer

Description automatically generated

Project Title: Playlist Manager Platform

Author: Emma Foley

Northeastern University EECE 2140.

Computing Fundamentals for Engineers

Fall 2024

Date: November 22, 2024

**Abstract**

Music is an important part of many people’s lives, whether they listen personally or with other people. Such, it is important that there are platforms available that allow for the management of their playlists and the music that they often listen to. The playlist manager project revolved around developing a playlist manager platform with elements and functionality that mirrored other music apps like Apple Music or Spotify. Using object-oriented programming and Tkinter for UI development, a program was created that allows users to add songs to a song library, create and delete playlists from a playlist library, and edit the playlists themselves. Object-oriented programming was used to develop a class-based system with playlists and the playlist manager represented as classes. This style of coding made the code more readable and organized. The use of Tkinter allowed for a simple UI to be constructed that allows users to interact with the backend code. However, the use of Tkinter did lead to some limitations in the project. Tkinter graphics and widgets are somewhat outdated in appearance, and the platform did not have ample support for refreshing windows being displayed by the program. This resulted in the implementation of a system where the user is responsible for manually refreshing the program being implemented.

Chapter 1

**Introduction**

* 1. Background

Music is pervasive in people’s lives. Beyond personal listening, music is played in stores, restaurants, gyms, and more. Due to the prevalence of music listening, playlist and music manager platforms are created to streamline the process of accessing music and creating playlists.

* 1. Problem Statement

Playlist manager platforms are necessary to assist people in accessing and organizing their music and playlists. The playlist manager project addresses this need by providing users with a place to store songs and playlists, as well as by providing users with options for editing the playlists they create.

* 1. Objectives

The primary objective of the project was to develop a program and associated UI that emulated many of the features commonly found on music management apps. This primary objective extended to developing a python script that supports adding songs to a song library, creating playlists in a playlist library, deleting playlists from the playlist library, and editing playlists that have been created. Another aspect of the primary objective was to create a UI that allows users to interact with the backend code to manage and view their playlists and music.

* 1. Scope

For the playlist manager itself, the scope involved supporting four major features: a song library, a playlist library, playlists themselves, and saving program data. The program was designed to allow users to add songs to the song library and to allow users to create and delete playlists in the playlist library. When users access playlists, they can add and delete songs from the playlists. Lastly, users are also able to save their data on the platform, and it is loaded the next time they access the platform.

There were some limitations to the scope of the project including limited familiarity with the Python language and UI development, simplistic UI development methods, and time. Due to lack of experience developing with the Python language and developing UIs in general, it took longer periods of time to develop the final code produced during the project, and more time to plan the development of the UI. Both limitations also fell in line with the limited time frame there was for project development. Lastly, Tkinter was used to develop the UI for the playlist manager, which lead to the development of a UI that is functional, but somewhat rudimentary and outdated in appearance.

Chapter 2

**Technical Approaches and Code UML**

* 1. Development Environment

The playlist manager program was developed in Python with the UI being developed with the python library Tkinter. The IDE used was Spyder version 5.51 to support development with Python version 3.12. Tkinter is a built-in python library that provides an interface for creating GUIs. Tkinter was used to develop the UI by importing it into the playlist manager code.

* 1. Data Collection and Preparation

The playlist manager platform is centered around managing the data associated with user’s song and playlist libraries. The data involved in the playlist manager platform is controlled directly by user interaction. Users control the songs and playlists added to and deleted from the platform. Data collection takes place through prompts and user entries on platform. Users can also save their data on the platform by selecting the save option, which causes the program to write the data currently contained in the program onto text files that are then loaded when the user runs the program again.

* 1. Implementation Details

The code for the playlist manager was implemented in two sections: the backend and the frontend. Backend code development involved creating support for playlist management itself, and frontend code development involved creating a UI for users to interact with the backend code through. To develop support for managing playlists, object-oriented programming was used to develop multiple classes to represent parts of the program and carry out actions related to playlist management. Two major classes were developed to represent playlists and the playlist manager application itself. The playlist class was used to represent playlists with a title and a list of songs contained in the playlist. Methods were then created to allow for the addition and deletion of songs from the list of songs in the playlist. The playlist manager class was created to represent an instance of the playlist manager class with a list of songs to represent the song library, a list of playlists to represent the playlist library, and a playlist that represents the current playlist if one is selected. Methods were then created for the playlist manager class to support adding songs to the song library, creating playlists, modifying playlists by adding and deleting songs, and deleting playlists from the playlist library.

To make it support saving and loading data to the playlist manager, methods were created within the playlist manager class to save playlist manager data and load data that exists in a directory on the user’s computer. The method to save the data creates two text files, one associated with the song library, and another associated with the playlist library. Then, a directory is created on the user’s desktop where the files are stored. To load the data, a method was created to check for the existence of the desired song library and playlist library text files in the directory created by the save method and then parse the text files. The data retrieved from parsing the text files was then used to populate the song library and playlist library fields of the current instance of the playlist manager class. To implement these methods in the program, the user is given the option to save their data in the main menu of the UI, and data is automatically loaded from the playlist manager directory when the application is run.

To develop support for the UI in the frontend of the code, the built-in Python library Tkinter was used. The library supports creating windows with widgets that allow the user to interact with the program. When users interact with widgets, events are triggered that cause certain functions to be run in the code. The UI created for the playlist manager features button and list-box widgets that allow users to navigate the platform and carry out actions related to their playlists. When the application is opened, users view the main menu, which they can navigate by interacting with buttons to pull up new windows to view the song library, the playlist library, or playlists themselves.

Chapter 3

**Project Demonstration**

* 1. Screenshots and Code Snippets

When the application is opened, the main menu window is pulled up, and the user is given options to navigate to the song library, navigate to the playlist library, or save their data.

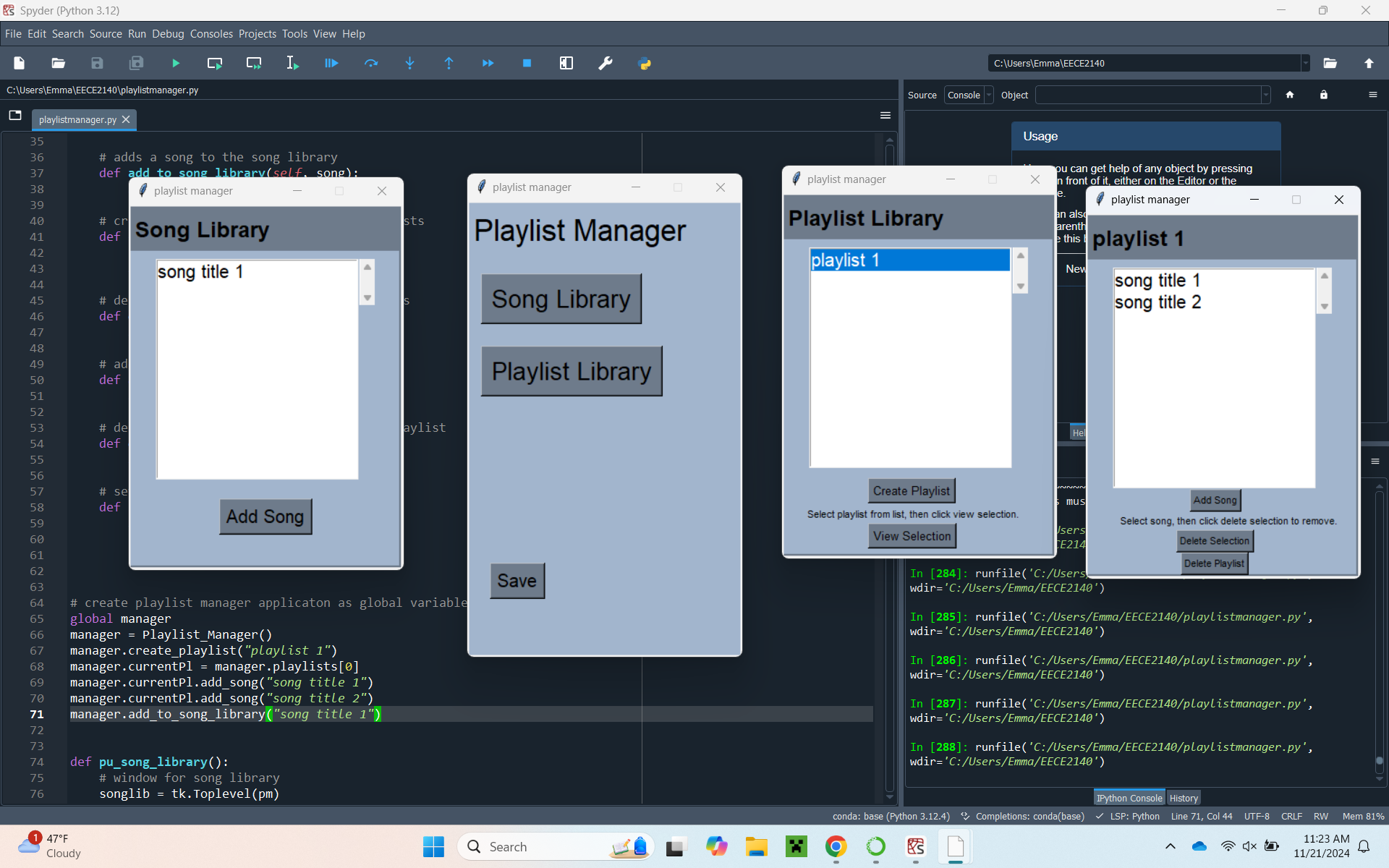


Figure 1 – Playlist Manager main menu window.

If the user navigates to the song library, a new window is pulled up that displays the contents of the song library. The songs are displayed in a list box that the user can scroll through should the amount of songs exceed the space necessary to view them in the box.

A screenshot of a computer

Description automatically generated

Figure 2 – Playlist Manager song library window.

Users are also given the option to add songs to the song library from the song library window. When pressed, the “Add Song” button pulls up a small window where users can enter the title of the song that they would like to add to the library.

A computer screen shot of a computer screen

Description automatically generated

Figure 3 – Song title entry window.

If the user navigates to the playlist library, a new window is pulled up that displays the contents of the playlist library. In a similar format to the song library, the playlists are displayed in a list box with a scroll bar.

A screenshot of a computer

Description automatically generated

Figure 4 – Playlist Manager playlist library window.

In the playlist library window, users have the option to create a playlist by selecting the “Create Playlist” button, which brings up a smaller window to enter the title of the new playlist similarly to how adding a song to the song library works. Users can view the individual playlists by selecting them in the list box, which causes them to be highlighted in blue, and then choosing the “Select” button. This action causes a new window to be pulled up that displays the songs in the playlist.

A screenshot of a computer

Description automatically generated

Figure 5 – Playlist Manager individual playlist window.

While accessing an individual playlist, the user has the option to add songs, delete songs, or delete the playlist. The “Add Song” feature functions the same way as the “Add Song” feature in the song library where a window is pulled up to allow users to enter the title of the song that they would like to add to the playlist. The “Delete Selection” feature allows users to delete songs from the playlist by deleting the song that they select in the list box. The “Delete Playlist” feature allows users to delete the playlist currently in view from the playlist library.

The save button on the main menu causes the current song library and playlist library of the playlist manager to be saved as text files. The files are saved to a designated playlist manager directory on the user’s desktop. If the directory does not already exist on the user’s computer, then the program creates a directory.

A screenshot of a computer

Description automatically generated

Figure 6 – Directory with text files for the saved song library and playlist library of the playlist manager.

Chapter 4

**Discussion and Future Work**

The result of the project was an application that could effectively display and store data related to songs and playlists that allows the user to manage their music. The application provides simple options that allow the user to interact with the program. The code for the backend part of the project was written using object-oriented programming by representing aspects of the project as classes with methods that carry out actions related to the object. This style of coding lead to a well-organized section of code for the backend because all the objects being represented by the application were clearly distinguished, and the code related to the objects was grouped within the classes. Choosing to represent the playlist manager itself as a class also allowed for all the data related to the instance of the playlist manager being run to be stored in one location, which increases the readability and reusability of the code.

Tkinter was used to build the UI because it is easy to access as a built-in Python library, it works well cross platform, and there is a lot of documentation on it. Tkinter works well for building small scale, simple UIs for the aforementioned reasons; however, using Tkinter to build the UI for the playlist manager did provide some limitations. The graphics supported by Tkinter appear outdated and rudimentary, which decreases the visual quality of UIs built with Tkinter. A major Tkinter related limitation was that the windows of the playlist manager must be manually refreshed to view updates to the song library and playlist library due to the lack of support for refreshing windows in Tkinter.

In the future, there are some alterations and extensions to features of the playlist manager that could be made. For alterations, the UI could be modified so that the user uses “back” buttons to navigate back to the main menu from other windows to avoid opening new windows and possibly fix some of the problems with refreshing pages in the UI. For extensions of features, support for changing the order of songs on the playlists could be added, and the ability to upload mp3 files to the platform could be added, so that audio could be played through the application. The files could be added to a directory that is accessed by the application and played through the playlist manager when songs are selected.

Chapter 5

**Conclusion**

In the end, a playlist manager application was developed that allows users to manage their song library and playlists. The application has support for adding songs to a song library, adding and deleting playlists from the playlist library, as well as saving data from the application and loading data to the application. The use of object-oriented design in the development of the application resulted in code that was well organized and readable. The use of Tkinter allowed for the creation of a simplistic UI that allowed users to interact with the play list manager code. Object-oriented design and a Tkinter-based UI allowed for the successful emulation of some of the key features of popular music apps like Apple Music and Spotify by allowing for the support of a song library and a playlist library.

Overall, the process of development and techniques used during the development of the playlist manager are significant because of their importance to developing projects in general. Object-oriented design is an extremely common style of code design because it results in the production of code that is readable, well organized, and reusable. Working on a project using object-oriented design was significant in that it provided experience in developing a program with an object-oriented code style. Similarly, developing a UI with Tkinter, despite some limitations, provided more experience in designing a UI and working to make it cohesive with the backend code.

**Bibliography**

[1] “Array - Efficient Arrays of Numeric Values.” *Python Documentation*, Python Software Foundation, docs.python.org/3/library/array.html. Accessed 12 Nov. 2024.

[2] “Reading and Writing to Text Files in Python.” *GeeksforGeeks*, 7 Oct. 2024, www.geeksforgeeks.org/reading-writing-text-files-python/. Accessed 13 Nov. 2024.

[3] “Graphical User Interfaces with TK.” *Python Documentation*, Python Software Foundation, docs.python.org/3/library/tk.html. Accessed 14 Nov. 2024.