Music Mixer

CSE 111 Final Project Report

By Tyler Lennen and Prabhdeep Samra

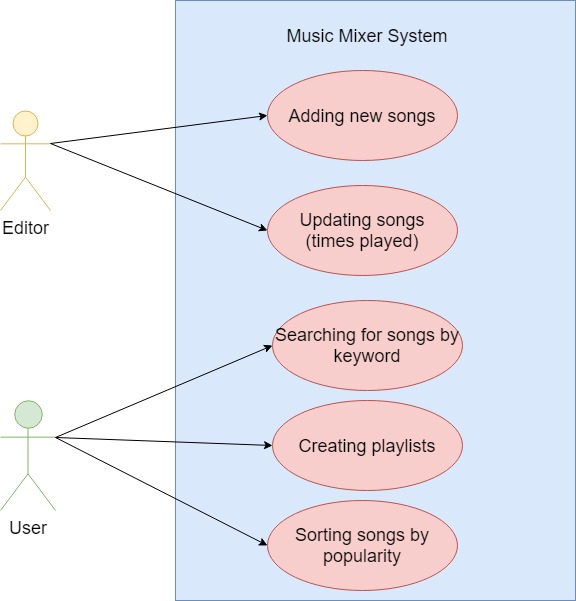
**Introduction:**

Our project was a music database that serves as a collection of songs and a way for users to create playlists for songs. It allows administrators to maintain a data table of songs that can then be used by others to reference. The project is a GUI application built in Python 3 using sqlite3 and tkinter. More information on using the interface can be found in implementation.

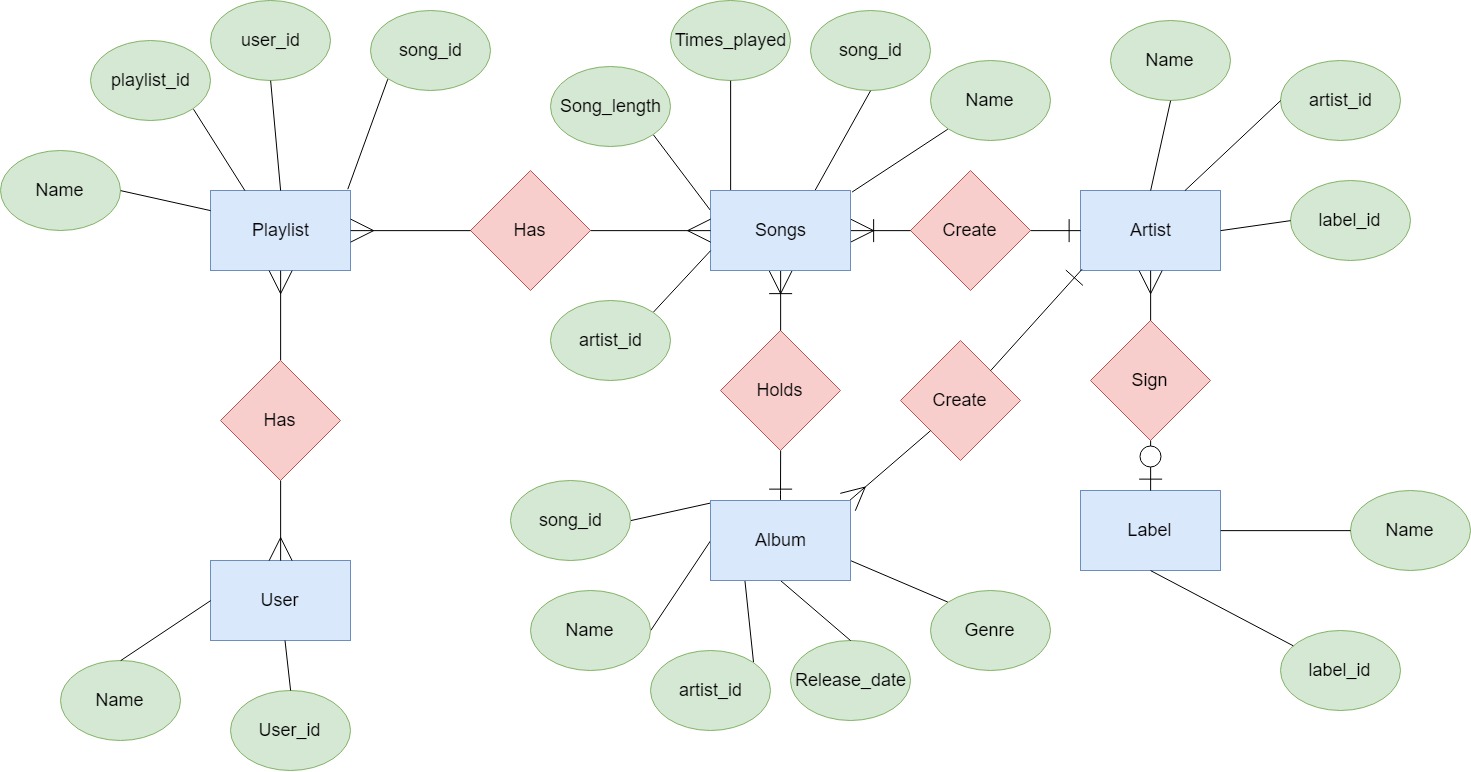
**Use Cases:**

There are two main subsets use cases, the admin and the user. The admin’s job is to keep the database updated so in the application they can add songs, artists, labels, albums, and they can update songs for number of times played. The user can look up these songs and search through the catalog by any of these tables. They can create and manage playlists of songs and check other users’ playlists out. Users can also check out the most popular songs, by genre, to find new songs to listen to.

This is the UML diagram with both use cases:



**ER Diagram(can be seen on the presentation):**



**Relation Schema**

* Label(l\_name, l\_label\_id)
  + (0-1 -> Many) Artist
* Artist(a\_name, a\_artist\_id, a\_label\_id)
  + (1->Many) Songs and Album
* Songs(s\_n\_name, s\_song\_id, s\_artist\_id, s\_song\_length, s\_times\_played,s\_album\_id)
  + (Many->1)Album/Artist
  + (Many->Many)Playlist
* Album(al\_name, al\_song\_id, al\_artist\_id, al\_release\_date, al\_genre)
  + (1->Many)Songs
  + (Many->1)Artist
* User\_playlists(up\_playlist\_id,\_up\_user\_id)
  + (Many->Many)User
  + (Many->Many)Playlist
* Playlist\_songs(ps\_playlist\_id,p\_song\_id)
  + (Many->Many)Songs
  + (Many->Many)Playlist
* Playlist(p\_playlist\_id, p\_name)
  + (Many->Many)User\_playlists
  + (Many->Many)Playlist\_songs
* User(u\_name,u\_user\_id)
  + (Many->Many)User\_playlists

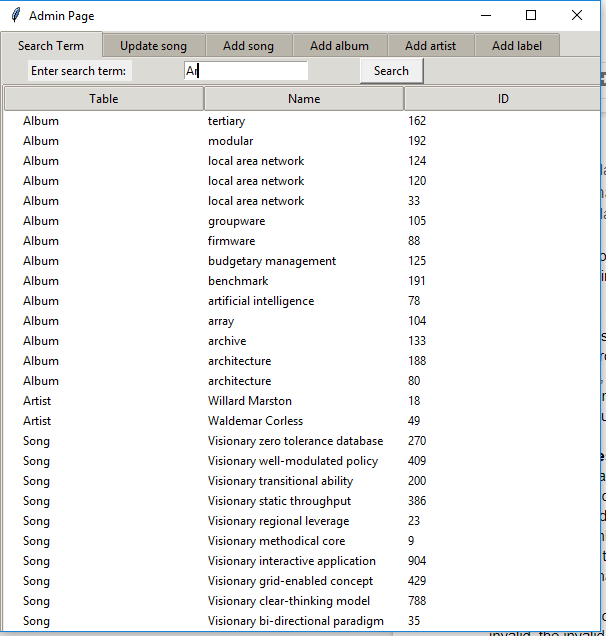
The tables user\_playlists and playlist\_songs are between the tables in their names and are used to handle the ids in the database.

**Implementation:**

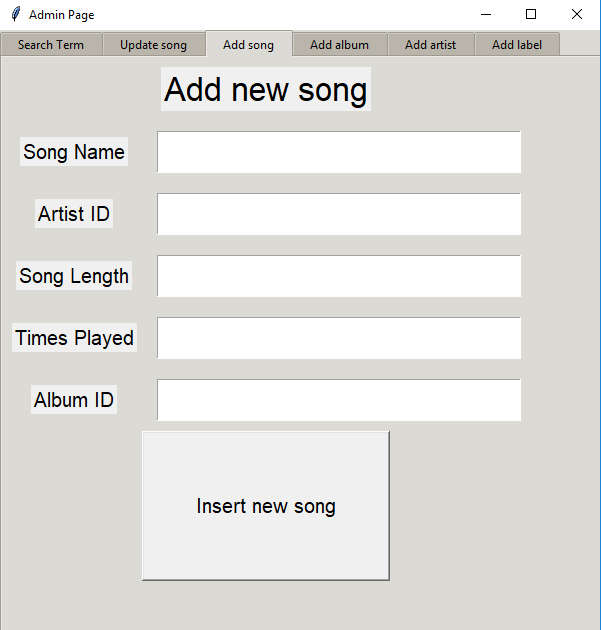
The program uses sqlite3 and tkinter to create an application that can be used to manage and maintain music profiles. The admin and user uses cases are separated to show off the capability of the application, which is chosen by a window with two buttons. Once a user type has been chosen, a new window will pop up with the given use case. The queries that were used for this application are found at the top of the music\_gui.py file.

**Admin Use Cases:**

The admin use cases are about maintaining the database. They can add a song, artist, album, or label and they can update the times a song was played. These functions are separated into tabs, which the admin switches through in order to add new data. There are checks in place against adding unknown values to prevent bad data. Every input that depends on another table requires the id of the given table and in order to find that, they can locate it using a built in search function that searches the song, label, artist, and album tables. It is shown below:



The search functions are entry text boxes that allow for data to be inputted. If the data field is invalid, the invalid fields will be replaced with the value ‘Invalid’. An example of the song add function is below.

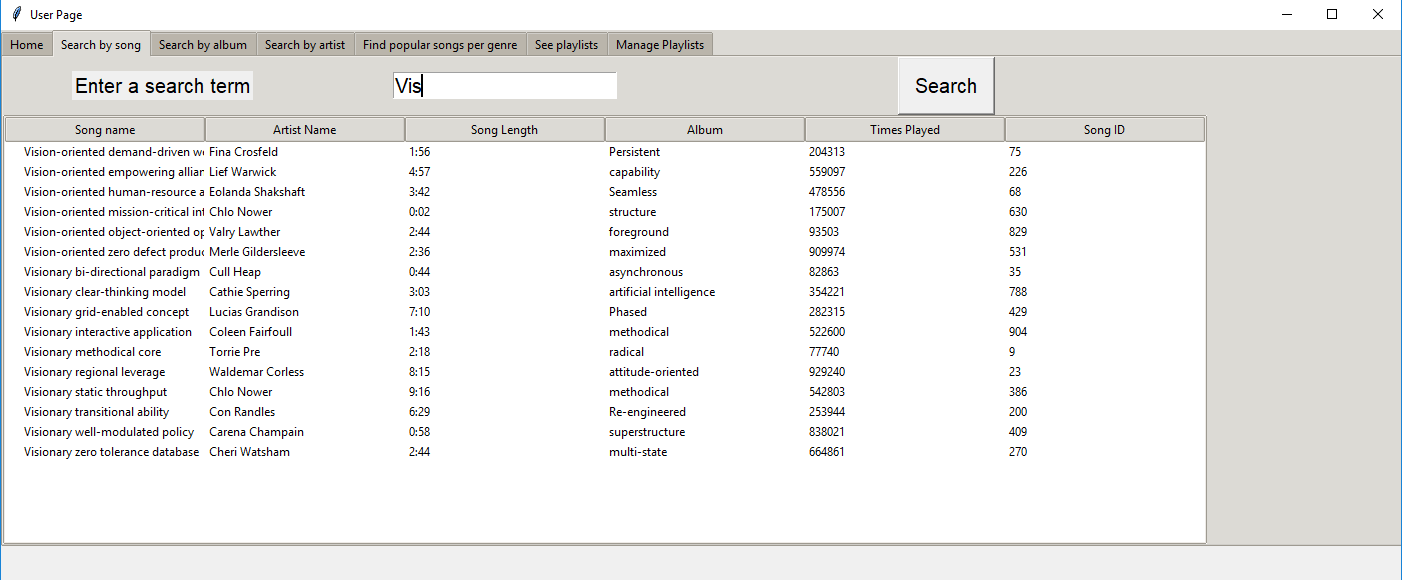


The admin can also update the times played with the update song tab. These are all the admin use cases in our application.

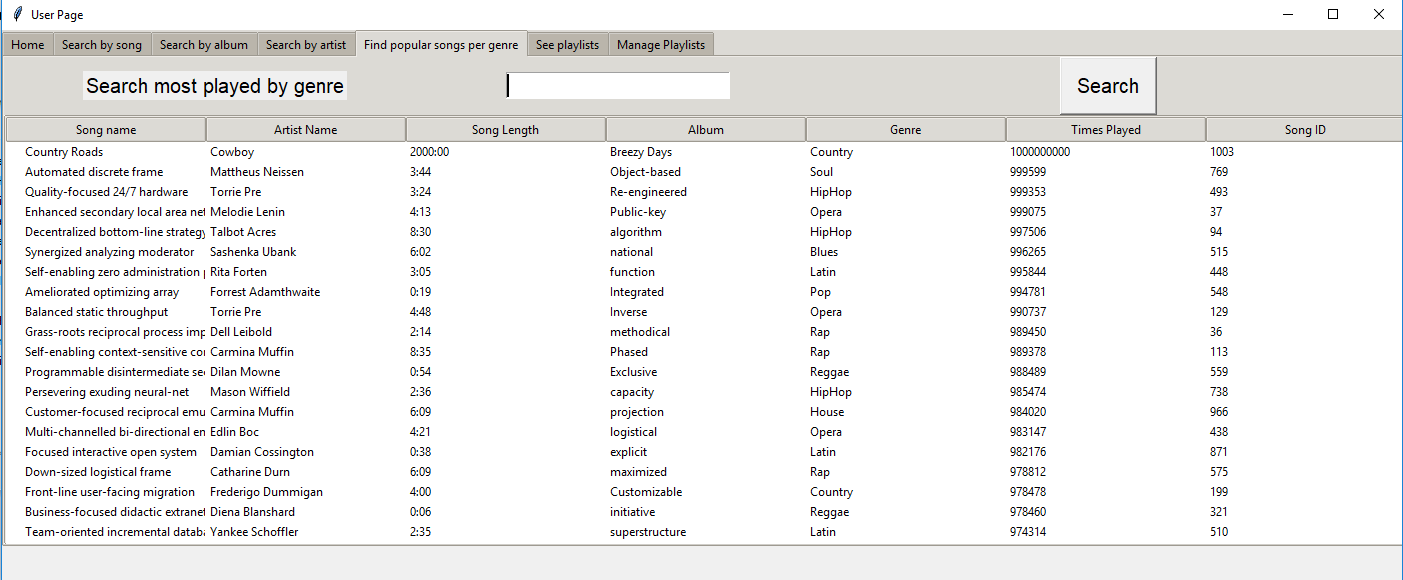
**User Use Cases:**

The user has several things that they can do: they can find new songs, search up songs, and create and manage playlists from the application. This part of the application has checks and balances to make sure that they data they are either creating or looking for is correct and does not break the database’s conventions. Users are saved in the user table in the database and are differentiated by id. This combination can be used to login, using our login screen. If a user is new, they can create a new profile and are assigned a login. This is shown below:

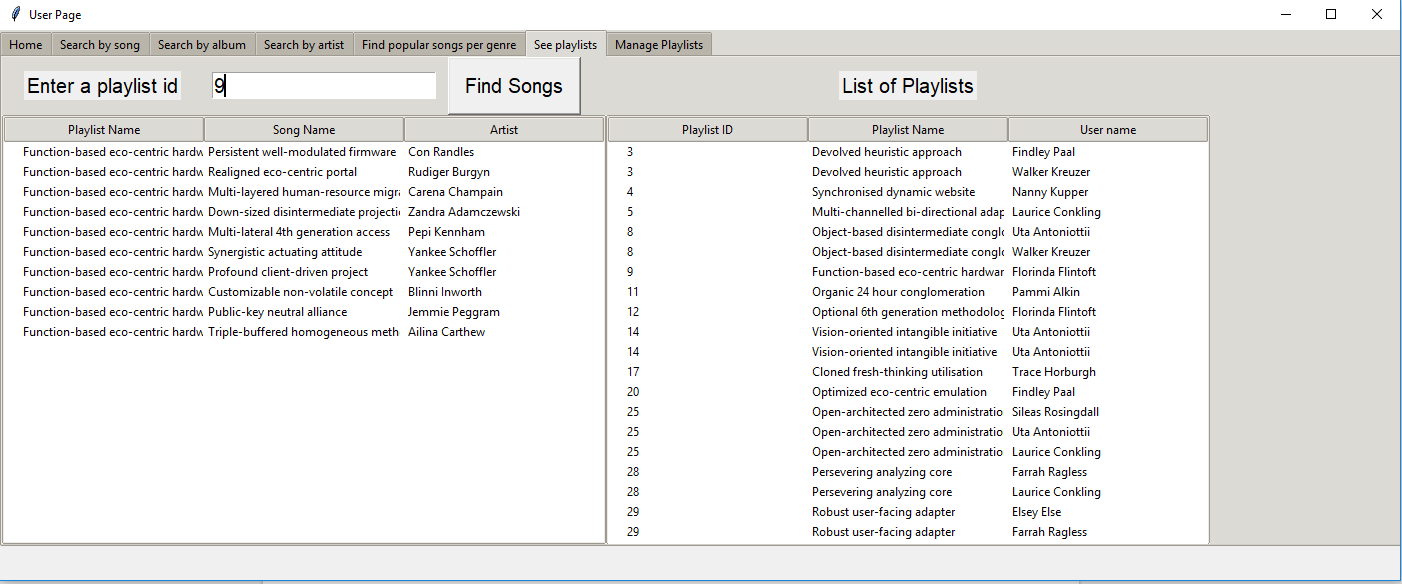
The search function is also implemented to find albums, artists, and songs. These are in separate tabs for each, but all contain the same data in a different priority order. The uses for this are when a user knows a song, album, or artist and wants to find more information about them. This is shown below:



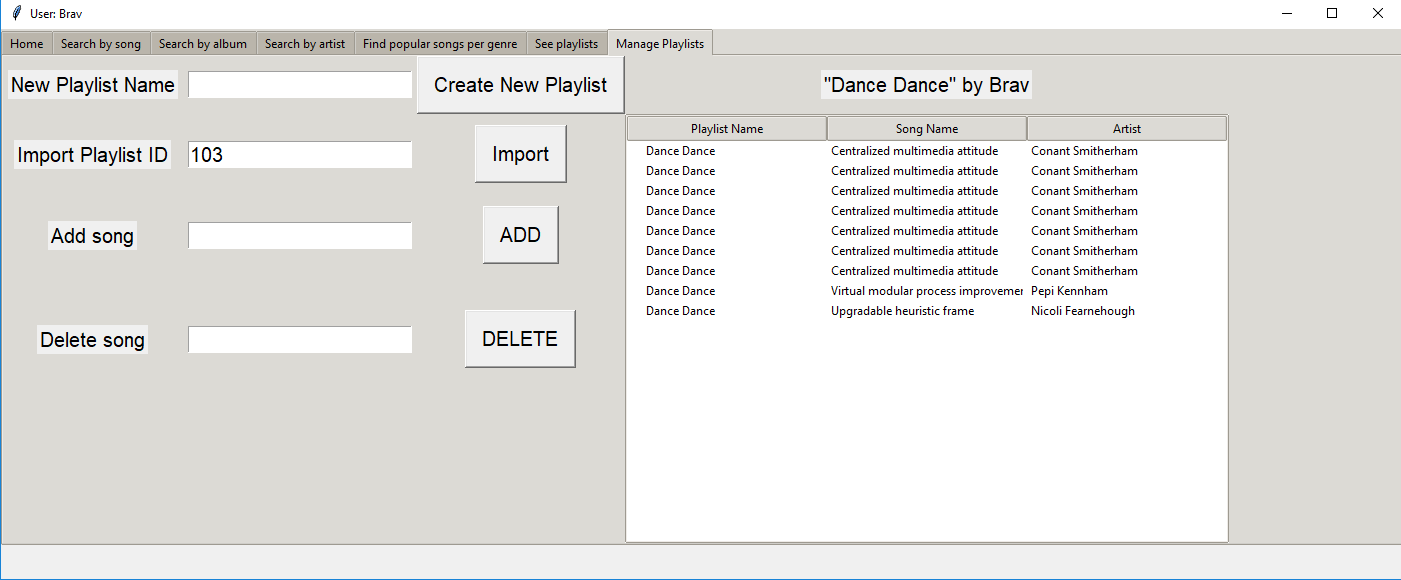
The way for users to find new songs is to look by genre to see the most played songs. If no genre is selected, it shows the top played of all songs. This allows users to find new songs based in their interests by looking at the popularity. This is shown below:



The user can also look at other people’s playlists and search through the songs. The playlists are found through there ids and show the songs inside the playlists. This is shown below:



The final thing that user’s can do is create playlists. They can either import playlists that already exist or they can create new ones. The playlist can remove songs or it can add songs and can be managed. This is shown below:



**Running the application:**

The application needs a Python 3 environment and depends on the sqlite3 and tkinter libraries. The application can be double-clicked and run if the above are satisfied.