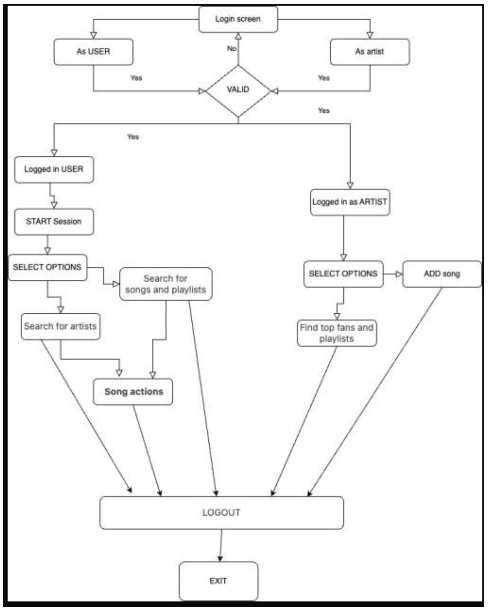
Documentation:

(a) General Overview

We worked on SQLite in Python for this project. The general overview on how the project works is, first the user is prompted by the login/Register page where a new user can register with a username and password, while existing users need to input a valid username and password to enter. After the user logins in, they would be prompted with the first screen which asks the user what task they would like to perform. They can start a session, search for songs and playlists, searches for artists or ends the session. If a user who has not ended the session logs in again, the user would continue the previous session. More information about each option is given below. The user can select a song and after ,either listen to the song, get more information about the song, add to playlist or exit. For an Artist , they can add a song or find top fans and playlist. Below is rough draft for user flow that we used to create our project



User guide:

Login Screen: -> An option for a user/artist login is set up, it detects whether it is a user or an artist and provide the proper menus as specified. There is also the option of users signing up by creating a unique ID and password. After the login, the user is able to log out as well

Start a session: 1 -> if the user chooses this option, they will be able to listen to songs and use the core functionality

User manual:
Search for songs and playlists: 2 -> if the user chooses this option, they will be able to search for the songs and playlists based on the keywords they typed. Now the user will be able prompted to either -listen to the song -see more info about the song -add the song tp the playlist

Search for artists: 3 -> if the user chooses this option, they will be able to search for the name of the artists and from the names of the songs they have performed.

Artist manual:
Add a song: if the artist chooses this option, artists will be able to add a song by providing a title and a duration of it

Find top fans and playlists: Using this option will let the artist see the top 3 users who listen to their songs the longest time and top 3 playlists that include the largest number of their songs.
End the session: 4 -> using this option will end the listening session for the user.

The user types the number corresponding to the function to perform a certain task.

(b) Detailed Design

We broke down our code into multiple user-defined functions, each having its own individual function.

startPage-> This function's main functionality is the start page of the program. The program firstly prompts the user to either register if they are a new user or log in if they are an existing user. If the user registers, they need to input a username followed by a password to enter the program. If the user logs in, then they need to provide a correct username and password to enter the program. The artist have a different login mode where they are prompted to enter their ID and then their password. Once the login is successful they are able to utilise the functions specified.
getOngoingSession-> This function would check if the user ended the session. If the user did not end the session, we would add NULL to the end session in the session table.

startSession-> This function allows the user to start a session and assigns them a unique session id. When the user logins, sessions automatically start, and the current time is updated as the start session table.

searchSong-> For a given inputted keywords, songs with the most inputted keywords in them will be displayed first while displaying all songs with matching keywords.. For each matching song, id, title, and duration will be displayed. We used the like operator to get songs with the most matches with the given keywords.

searchPlaylist->For a given inputted keywords, a playlist with the most inputted keywords in them will be displayed first while displaying all playlists with matching keywords. For each matching song, id, title, and total duration of the songs will be displayed.(We wrote 2 different functions for searchSongs and searchPlaylist to make the code more concise and workable)

HashPlistSongCnt -> It gets the matching keyword for the songs and hashes on bases on count of song and playlist. Once results are displayed , It merges the song and playlist together based on their count. Then use the key to iterate through to put it into one list. If no song with the count of keywords exists yet , then a new list needs to be made for the count and the same for playlist. Returns a dictionary.

searchSongAndPlaylist() -> The function prompts the user to input keywords for them to search songs and playlist. This function calls searchSongs and searchPlaylist using the help of the HashPlistSongCnt. This function has the scroller also implemented as only 5 items would show at a time and we can use ‘[‘ to scroll up and ‘]’ to scroll down.

searchArtistHelper-> This function queries through all the artists in the db usingSQL and creates a artistHash to compile all results form the multiple SQL queries. In the end the function returns a list of the aid that have been mention in the keyword search in the right order of display.

getArtistTuples->It queries the db for each artist to get the related info for each artist queried from the searchArtistHelper function. And gets the tuples of the conscience information to be displayed in the form of a list of tuples

searchArtist-> It prompts the user for keywords input, and then calls the helper function searchArtistHelper and getArtistTuples. The outputs from theses helper functions is used to create the paginated format of the results.This function has the scroller also implemented as only 5 items would show at a time and we can use '[' to scroll up and ']' to scroll down. The user is also able to make selection based on the result shown and then is able to get details of the artist and the songs they have performed. And if they chose a song from the displayed then it takes the user to selectedSong

addSongToPlist -> This function insert a certain song into a playlist. If the song is already in the playlist , it would not include it again and show an error message saying that song is already in the playlist

createPlist -> This function first gets the last playlists id and then adds 1 to that to get a unique playlist id. Using that it creates a new playlist.

sPlaylistsin -> This function returns the playlist the song is int

getUPlaylists -> Returns the user's playlist.

endSession -> This function ends the current section and updates the session by updating the end time with the current time when the session ended in the session table.

selectedSong -> This function allows the user to either listen to the song, more information about the song, add to playlist or exit. If a user wants to listen to a song, a session must be started. If the session is not started, the user will be prompted with an option to start a session and a new session will be created. If the user selects more information about the song, details about the song will be displayed. If the user selects to add the song to playlist , that certain song will be added to a new playlist or an existing one depending on the playlist id the user enters.

selectedPlaylist-> This function shows the information of the playlist like shows the songs in the playlist , playlist id etc. The user can then select the song or exit if they like.

addSongArtisit-> This function helps the artist to add new songs . When the artist is adding a song, if the same artist has a song with the same title and duration, the user is warned and either they can reject it or ask the user to confirm that they indeed want to add it as a new song.

getTop3FansandPlaylists -> This function gets the top 3 fans (users) of the artist and displays them. This function return the playlist with the most songs in them , ordered by displaying the playlists with most songs in the start.

main() -> The main function just calls all the above-mentioned user defined function in order. Whenever a user selects a option , a respective function is called which performs the task. First the main function prompts the login page using the startPage function. If the person logged in is a normal user then the function will ask the user what action they would like to perform in Start session, "End session", "Search for songs or playlist", "Search for artists", "Exit" . For each action a respective user defined function is called which performs the task. Like , startSession , endSession , selectedSong and selectedPlaylist , searchArtist functions will be called for

respective options. If the user is an artist , they can add Songs that calls the addSongArtist and they can display top3 fans and top playlists using the get3TopFansAndPlaylists().

(c)Testing Strategy:

Used Test data to verify the SQL queries, code functionality, and User approachability. We used DBMS software to verify the functionality of SQL queries and also to check if the output were matching our specifications. Python was the software used to build the framework and user interfaces to beautify the program. We tested the edge cases like testing on empty sets, same names for songs etc. We brainstormed for possible cases for each situation and tried to cover all situations possible to make sure the code behaves as we require.

For keyword searching, we build different possible cases to check our search. We added multiple songs with some specific keywords and searched with keywords within to check if the songs with more keywords specified showed up first. The bugs we found in this were , when we searched for songs using keywords, the songs did not show up in the order of keywords that appeared in the title. We used ‘like’ operator to overcome that problem. There was another bug when we searched for empty set. We solved this problem by using left outer join.

During the search Artist section, we encountered a bug in our SQL statements that prevented us from extracting the artist mentioned in the keywords and their songs. Trying to query both would result in duplicates and uneven ordering of priority. We were able to fix this by the use of WITH clause to create named subquery and used it query the song title and names of artist separately which resulted in a cleaner separation of tasks and made it easier to debug and obtain desired results
We tested the login criteria to make sure no 2 users can have the same username . We did not incorporate this before and gave us 2 passwords when we tried to login with the duplicate username . So we made sure the username is unique.

(d)Group Work: Distribution Strategy

We used to schedule a meeting time and spend 2-3 hours per meeting working on the project. We used to call over discord and work on the project via VScode live share, allowing us to work on the code together. If one used to get stuck on a problem, we all brainstormed ideas to help overcome that challenge. We made sure we completed our tasks which were assigned before our next meeting. We also had a discord group where we kept in touch every day stating 1) What work we have done , 2) what work we have pending and 3) What work we are stuck at if any. And if any member was available , we used to work together to solve the problems.
A breakdown of work followed

Dev - login, playlists and song search, playlists selection, song selection artist menu, song section (2) , Artist section
Hours Worked Dev - 18 Hours
Pratham - Worked on end/start session ,keyword search for playlist and song , displaying results , Song section , documentation
Hours Worked Pratham - 12 Hours
Vedd - Worked on the Artists section, name search for artists and select functionality , documentation
Hours Worked Vedd - 12 Hours
We also debugged each other's code in functionality so that we can work faster and be more efficient.