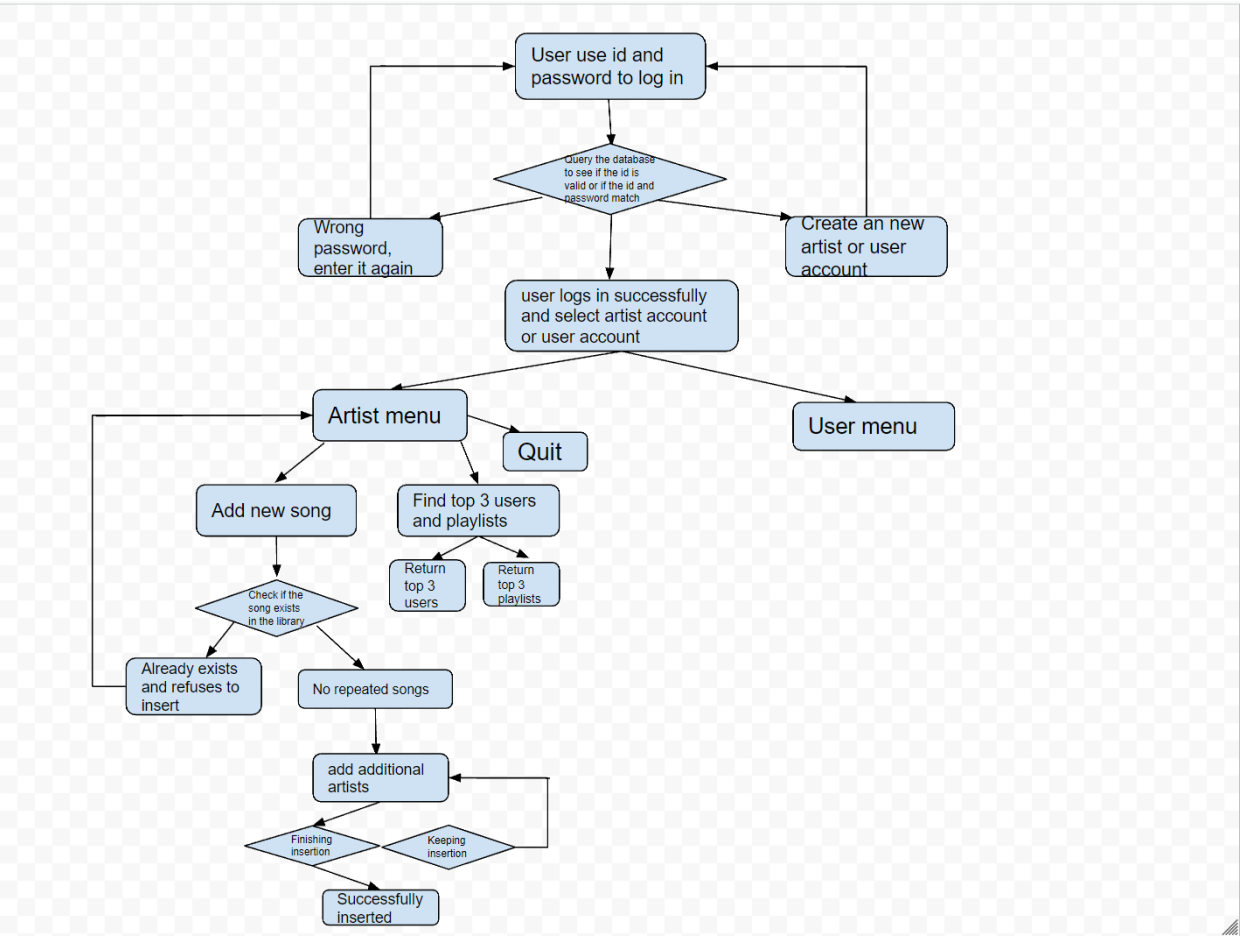
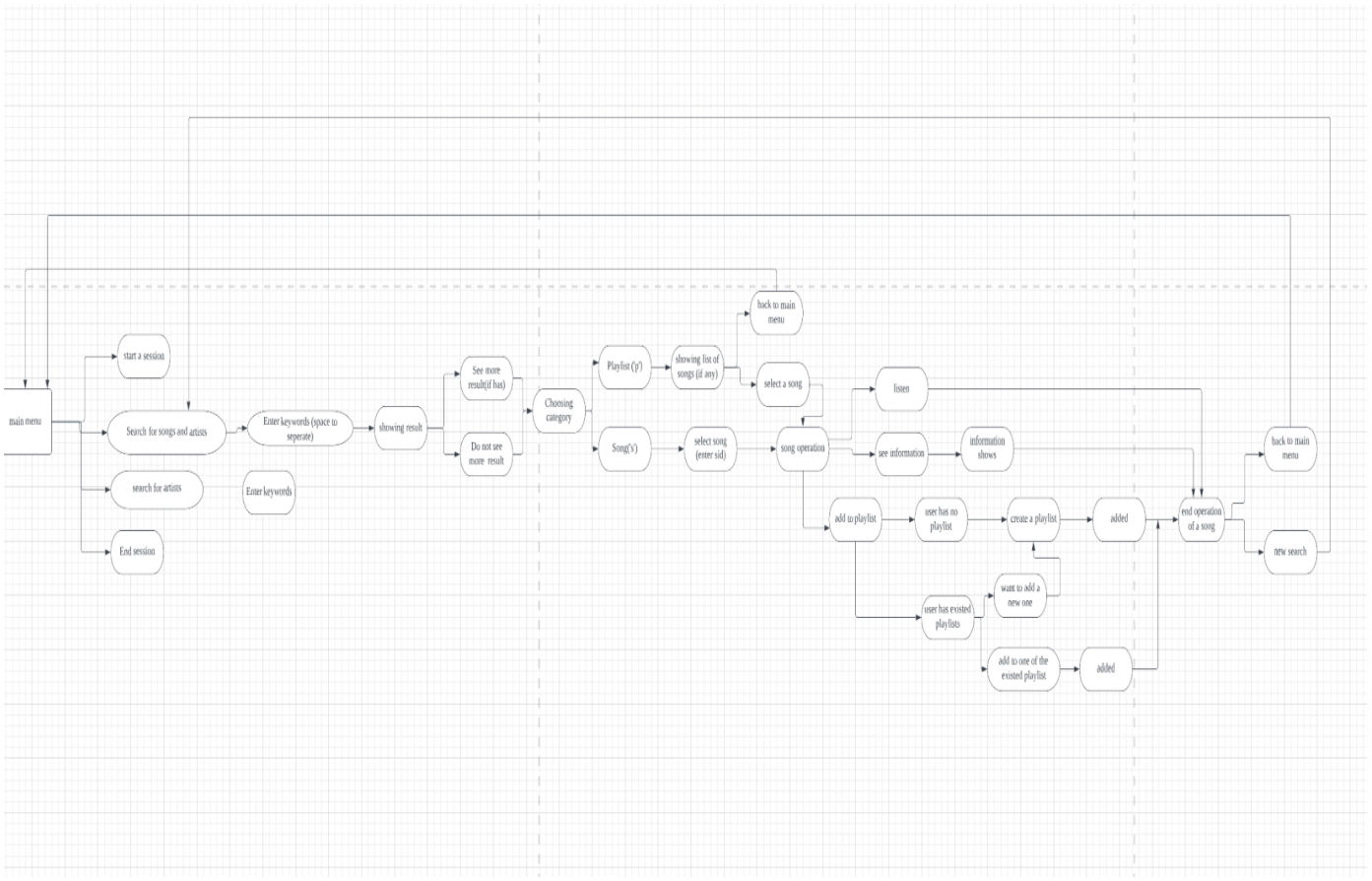
a) Login design:

1. user must enter database path first, and system will link to that pash and open the database
2. By using tkinter (python built-in GUI function), A visual window is created where the user can enter the username and password inside the text-bar. All passwords are not visible. The two buttons are Login and Create User.
    a. When the user selects the login button, the system will automatically jump to login_check() and pass the account and password entered into this funtion,By matching the existing_account() , the corresponding uid or aid will be returned and the current user will be displayed in user mode or artist mode.Then the user will see a main menu where the user can make selections,and open the user menu or the artist menu accordingly.
    b. Create User: The user enters his id (must less than 4 strings) and password, then the system will match it with existing_account(), if the id created by the user is unique, it will be added to the system, otherwise the system will show that the user already exists. Diagram:

(artist)



(user)



b) detailed design:

User action design:

1. Start the session:

   User manually select “1” from the main menu and the system will generate a integer for as the session number for the user. If the user has a session number before, then the new session number will the maximum session number +1. And if the user does not have a session number before, then a random number will be generated. This design avoid the situation that a user could not get a unique number due to the limited number of the choosing range of the session number.

2. Search for songs and playlist:

   The user will be prompt to enter a sequence of keywork (using “space” to separate those keywords). Then the database will fetch all the songs and playlists from the users. Then the result will be print on the screen. If there are more than 5 searching result, then the top first based on the number of matching keywords will be shown and the user will be asked to choose (“y” or “n”) to see if they want to see the rest of the searching results. But it there are less than 5, then the limited result will be shown on the screen.

   Then, user need to choose a category (“p” or “s”) and if “p”, which represents playlists is choosing, the songs in this playlist will be shown and the user could choose 2 to perform a song action or choose 1 to back to the main menu (all 4 user tasks)

In this part we use 3 error checking method:

3. Search for artists: The user enters keyword(s), unlimited number of keywords, then when the user enters 'exit' the system will automatically exit the input and start the search. The system will match the songs and artist names in the database by the keywords entered by the user. The system searches by keywords, for example, if you enter "plan", the system will match the song "god's plan". System will show the artist name if user input matched an artist, if the user input matched a song, the system will show which song is matched and this song is performed by which artist. After the system finishes matching, the user can select one artist by input the number, the system will show 5 matched artists of information each time, and the user needs to input “next” to the rest of the match artists. The system will also sort by times of the keywords matched. When the user selects an artist, the system returns all the songs that the user has sung, and if the singer has not sung any songs, the system lets the user choose. If you want to restart the search enter '1', if you want to return to the main menu enter '2'. When the song is selected, it will enter the song action function
4. End the sessions:

   When the user logout, the database will set the end in the session table to the current date.

Song action:

1. Listen (**listen (self_uid, song_sid)**): first check if the user has a current session number. If the user does not have a session number before, then the program calls start_session(self_uid) and now update the current_session global variable. Then, check if the user listened to this song within the same session using database query. If there is a record, means the users listen to this song and use “UPDATE” query to update the “cnt” field of the listen table. If there is no record, then insert a record into listen table
2. see information **(see_information(song_sid)) :** show the song sid, title, and name of the artist, duration of the song and the playlist if existed. Since we could not make sure there is a playlist contains this song, so we use LEFT JOIN on playlist and LEFT JOIN plinclude.
3. Add to playlist (**add_to_playlist(self_uid, song_sid)**): We first check if the user has a playlist or not. If there is a record, then we listed all the playlists the users have and prompt him choose one playlist he wants to add the songs in and use sqlite query “INSERT” to add the song (id, title, order) into plinclude. And even the use has an exist pid, we design he could also add the song into a new playlists by providing a new playlist name. And if there does not exist a record, we prompt the user to first create the playlist by providing a valid playlist name and then add the playlist into playlist table using “INSERT”. And besides insert the playlist into the playlist table. We also add the song into “plinclude”.

Artist action design:

For the add song part, first, we have to let the user input the title and duration of the song because there may exist a song with the same title and duration, which means that song may have already been inserted into the database. In order to avoid data duplication, I created a list that includes all the duration of the songs performed by the user, and the title of the songs is entered by the user. I used SQL to select these data and insert them into a list and let the user's duration match it. If the input duration is in the list, it means there already exists a song performed by the user. Then the system will remind that the song already exists in the database and deny insertion. After that, the user will go back to the artist menu and decide to add a song again or find the top users and top playlist or exist. If the duration

is not in the list, it demonstrates that the input is a new song. I used random function and while loop to generate unique sid. To make a match, I will use a list called sid, which includes all songs' sids from the database. If the sid already exists, then random generation will be performed again.Next, we will ask the user if there exist any additional artists to add. If the song is performed by the user only, the user's aid will insert into the list called the aid_perform. The user can type 2 and enter the other aids if additional artists exist. We make a while loop to check if the user enters a repeated aid or an aid that does not exist in the database. Also, we set a while loop to allow the user to enter several additional artists; every time the user wants to add an aid, they should type 2 and repeat the inspection process described above until they do not want. After that, we will use SQL to insert the duration, unique sid, and title into the songs table. Additionally, the unique sid and aid will also be inserted into the perform table. Because I use a list to store these aids, I use a for loop to add aid and unique sid. Finally, close the program.

For the find top 3 users and playlists part, I just wrote two SQL to implement the function under a while loop. For the top 3 users, I ask users for input 1 to start finding. I select uid from listen table and combine listen table, songs table and perform table together through sid. Then group these uids by user's id. To judge the top fans, I use the sum times the duration as the listening time, and order it by decreasing order. Therefore, the top fan who listen to users songs for the longest time will stay at the top. Finally, we use the limit to ensure that there will be no more than three top fans. For the top 3 playlists, I ask users user enter 2 to start finding. I use the user id to filter the songs performed by the user in the playlists and count the number of sids; if the count number is larger, it shows that many user's songs are in the playlist. We print them in decreasing order to make the playlists containing more songs from the users above. At last, we use the limit function to make sure that the top playlists will not be more than 3. If the user types 3, the whole program is finished.

c) Testing strategy:

We create our own database and for the query parts, we first write down the query and test it on our database and find if it shows the right output. And we also use other database like the database we use in the assignment2. Besides, we did a lot of error checking to avoid the user break the system. For example, the string matching and the case insensitive. And when an error occurs, there will have a prompt and let the user enter again their choice.

d) Group work breakdown strategy:

Anbang Li: Working on the aritists actions which include add a song and find top fans and playlists. Time spent: More than ten hours.Method of coordination: Talking with group members and and discussing the problems together.

Xuemeng Wei: works on the user start sessions, users search for songs and playlists, songs actions and end the sessions.

Youming Zheng: I am responsible for the integration of the entire program, the user interaction pages, and the user login process, and artist search function after user logged in. Spent more than 20 hours.