a. General overview of the system & a small user guide

Overview

The program we created simulates a music app in which allows listener and content creators to manage their own data and profiles. On the user side, this program allows them to search for songs by keywords, listen to songs and manage their own playlists. On the artist side, this program allows them to add new songs they created and search for their top users and playlists.

User guide

Upon starting the program, a prompt will show asking a user for primary actions, **login**, **sign up** and **exit**. Signup allows the user to create a new account and use it to login later. When prompted to login, user must enter their user id followed by their password. Note that the user id is not the name of the user, but the actual id. If the credentials are correct, the user will be logged in as either user or artist. If an end user is both a user and an artist, the program will ask which role the end user would like to log in as.

Upon logged in as either a user or an artist, a menu will show which consists of choices, users simply need to type the corresponding number to signal which action the program should take. From this point on, the program will give specific guidelines for the constraints on the input. For example, if an artist chooses to add a song, the program will ask the artist to enter the title and the duration, separated by spaces, and it also prompt the artist that the title may not be empty, and the duration may not be a negative integer.

b. Design (see graphical design at the last page)

The program is divided into two major parts, artist mode and user mode, since the end user may only access one of the roles at a time. In another word, the program can execute smoothly with the other part missing as the codes are isolated. Both roles consist of their own SQL command file, UI design file and a functionality file. When you are logged in as a user, you have no access to the files that belongs to artists.

The SQL command file stores all SQL queries in which the role calls to connect to the database, for example, artist modes have queries that adds a song to the database while user mode have queries for starting a session. These files define SQL queries stored as stings, we have one SQL string for almost every query we need, but when it comes to searching with key words, since the number of key words is dynamic, we used a for loop to concatenate all keywords into the query, thus defining a unique function specifically for this job.

The UI design files stores the UI design in the terminal, precisely, it defines how the information will be present to the user in the terminal and what it will look like. It consists of the menu format

that the users will see, which is defined as a constant string, as well as individual rows when printing the results. As the number of results are also dynamic, a for loop is used here to ensure that only 5 rows may be presented at a time.

Last and most importantly, the functionality files define the mode classes and their corresponding functions, specifically, each of the functions defined under the mode classes corresponds to the main function needed for this role.

Another login.py defines the interaction at login page. It mainly consists of a while loop in which ask the user to give a proper selection of choices. Afterwards, it will invoke the login function in which asks for credentials or the signup function in which asks user to provide a unique id and password for registration based which was chosen. For users, search functions shall take and validate the inputs in a while loop before getting the SQL query for connection. The database will later return the results and the search function will then pass the results to UI function for printing in the terminal. Same mechanism is used for artists. As for adding a song or starting/finishing a session, the corresponding function simply invokes a SQL query as no complicated query or display is needed. If the user wishes to logout or exit the program, the program will always end any current opened session before any further action is taken. We aim to make each functionalities have their own function in the program.

c. Testing strategy

Each part of the program (login, user mode, artist mode) were tested separately with input combinations that covers all statements (lines of code) & boundary cases. The following strategies were used to test each part of the program efficiently:

- unit testing: each method & function in script were tested with a set of input combinations that could reach every single line of code (statement coverage 100%)
- functional testing: for each stage that requires end user to enter the input, test cases were designed as a set of input combinations that covers all possible outcomes of the program behavior; for example, when testing the functionality of “find top fans and playlists” in artist mode, 4 types (artist that returns more than 3 fans & playlists, artist that returns more than 3 fans but less than 3 playlists, artist that returns less than 3 fans but more than 3 playlists, and artist that returns less than 3 fans & playlists) of artists were selected to perform the combinatorial testing
- boundary value analysis: for each input stage, boundary test cases were designed (e.g., empty input, invalid input) to verify the correctness of program in error handling

d. Group work break-down strategy

1. Jianxi is in charge of the structure of overall project as well as the User mode functionalities. He planned out what the system should look like, which functions are supposed to organized inside the same file and what classes will be needed for the project. He also completed all functions needed when logged in as a user, as well as its UI designs. Jianxi spent approximately 15 hours on working the project.
2. Jiemin is in charge of the login page and Songs class. He created the logics in the login page and implements them, he also made the Songs class with fields storing necessary properties as well as functions to change the state of a Song object. He also completed all the UI designs at the login page. Jiemin spent approximately 14 hours on working the project.
3. Yihe is in charge of the Artist mode functionalities and design documents. He completed all functions needed when logged in as an artist, as well as its UI designs. He also wrote the design documents and created data flow diagrams when for users and artists. Yihe spent approximately 13 hours on working the project.

Communications were made through in-person meeting, Discord group chats, & GitHub project management; works are partitioned into less-coupling branches (login, user mode, artist mode) & distributed at the beginning of the project; once all branches were done, the parts were merged & connected to finalize the final project. All three members involved in testing of one another's