MINI PROJECT 1

Collaborators: Purnapushkala Hariharan, Tarush Shankar and Khalid

Summary

The given project aims to bring the finest experience of music to the concerned users. The users are categorized into artists and regular users. The regular users can sign up, play music, and utilize the extensiveness of SQL databases to play around with available features. A regular user can also login, start a new session, listen to songs, search for songs and playlists as well as for their choice of artists. For an artist, our project system enables them to log in and start an artist interface. The artists can add new songs and find top fans and playlists of their songs.

Assumptions:

a) When login in as an Artist, your ID should already be in the pre-existing database.
b) The user needs to start an active session.
c) The user id (whether regular or artist) and the password should match with the record in the Database. Should the user ID and password not match the user will be prompted to enter a password again. Same UIDS with different passwords are treated differently.
d) Every artist is a user and unless specified otherwise, the users are not the artists.

Guidance to the User

Before starting the program, the user needs to install 'maskpass' module to properly execute the functionality of our system. This can be done by using the following command 'pip install maskpass' on the command terminal being used.

After ensuring that the module is installed, run the command python3 main.py <database_name> on the terminal. Depending on if the database already exists, either a new database is created or a pre-existing one is loaded.

Once the program has started, the user will be presented with an option to login with their id and password. If matched the user will be prompted to the main page of our program. If not, the user will be prompted to enter the correct pairing of password and the uid.

The user can search for songs and playlists with user entered keywords, search for artists with the keywords for their choice. Start (and end) a session and logout.

Sessions created by the user are used to track the songs you listen to, and the count of each song listened.

Addressing the main functionality, the user is prompted with a list of options the user can do with a song. The user can listen to the song, some additional info about the song and add the song to a playlist.

For an artist, the system presents them with the option to add a song or list top 3 fans or playlists. Should the artist decide to add a song, the system checks if a song of artist entered name and duration exists or not. If it happens to be the case, the user can choose to add the song regardless or not.

Finally, to exit a program, the user can choose the logout option which will successfully end the program and make sure that the database is updated with the relevant information.

Software Design

The system functionalities are written in python3, and it utilises an SQLite database to store data. The tables and the fields in them are coherent with the database schema provided to us on eClass.

Our Python structure comprises of four python files which share and delegate their functionality with each other. They are main.py, songactions.py, userAction.py, and artist.py.

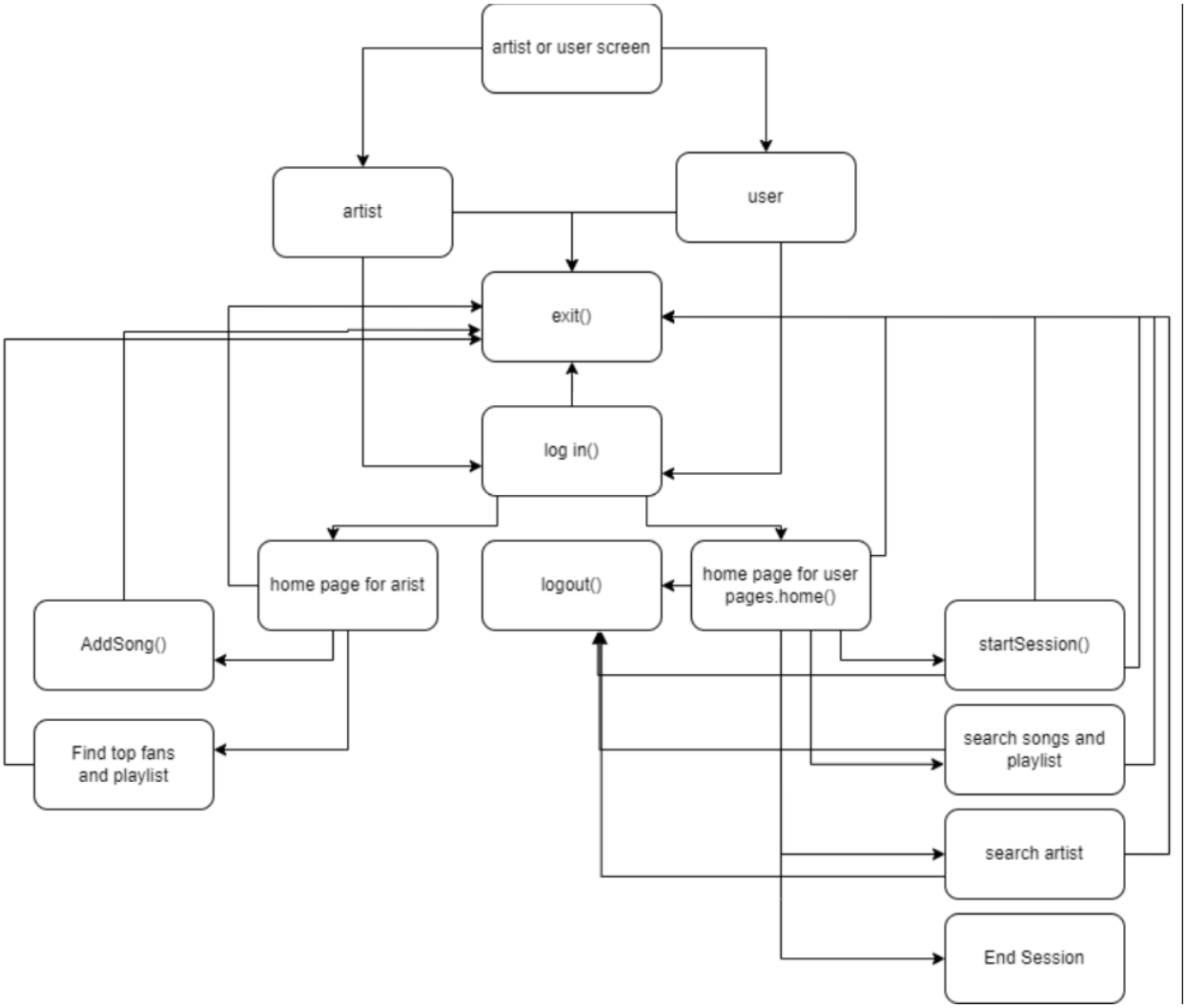
main.py (): The main python file which presents the user screen with the list of options. The file also glues the logic implemented in the other three python files to ensure proper functioning of the project.

songactions.py (): implements the logic of viewing the details of a song, adding the song to a playlist and listening to the song.

userAction.py (): Implements and Deals with the logic of starting a session, ending a session, and searching the songs, playlists, or artists.

artist.py (): The following python program implements the logic of helping the artist add a song, find top fans and playlists and logging out of the session.

The flowchart below shows the flow of functions and how the functions interact with each other.



Testing

We tested our modular functionalities by using the edge cases and usual cases. We did long and extensive sessions of deep debugging and testing to ensure that our system doesn't crash and deal with unexpected errors. We

tested the functionalities of main.py first since that acted as a base and a glue logic class. After passing the base cases we tested the functionalities of artist.py which dealt with the artist screen, and userAction.py which dealt with the user screen and interface. Finally, we tested the songactions.py which dealt with listening, viewing, and adding the song to a playlist. We used Aaron Gu's testing data provided.

Work Breakdown

Tarush: Worked on the userAction.py, helped with debugging, and made the report.pdf with consultation from group member. Hours contributed ~ 20

Purna: Worked on artist.py and songactions.py, helped with debugging. Hours contributed ~ 20

Khalid: Set up the git repository for the project, helped extensively with debugging and setting up main.py. Hours contributed ~ 20