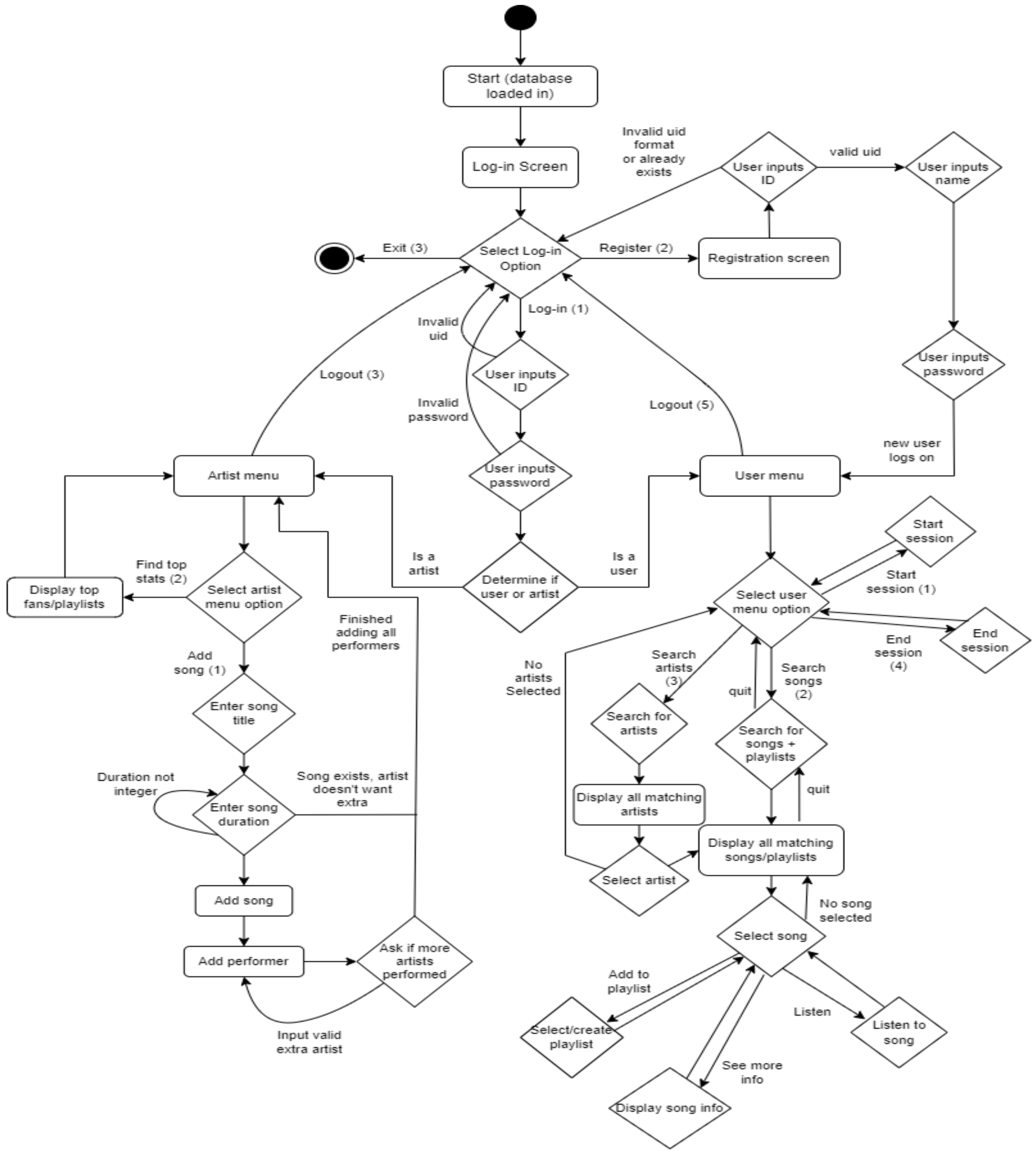
CMPUT 291 MINI PROJECT I

Design Document

Lawrence Wang (lxwang), Mohammed Saleh (msaleh), Kody Diep (kdiep)

GENERAL OVERVIEW

Attached below is a flowchart for our application:



User guide:
Once the application is open, the user is directed to a login screen where they can either login as a user/artist or they can register themselves as a new user.

If logged in as a user, the user is able to start a session and end any currently running session. Sessions are automatically ended when the user logs out.

Users can search for songs and playlists; they can enter space separated keywords which allows the system to search for similar songs matching these keywords. Playlists can be selected to display songs. Songs can get displayed, and the user can select any of these displayed songs to either listen to it, view more info or add it to a playlist.

Likewise, the user can search for artists and get songs from there as well.

If logged in as an artist, the artist can add a song to the database along with all required information for that song. In addition, the artist can extract their top 3 users and playlists.

Logging out of the application brings us back to the main page, where they can login as a different user, register or exit the program.

DETAILED DESIGN

Our application is written in Python and is contained in three files:
1. `projcloud.py`
2. `usermenu.py`
3. `artistmenu.py`

Where `projcloud.py i`s our main log-in application, `usermenu.py` is our user menu and all corresponding user functionality, and `artistmenu.py` is our artist menu and all corresponding artist functionality. Each file has multiple functions that work to build our application as a whole.

`projcloud.py`
- `main()`
  - Main control flow of the application. Controls whether the user is logging in or exiting the app.
- `register(id, name, password)`
  - Register the user, adding it to the database.

- `login(logintype, id, password)`
    - Attempt to login the user by verifying credentials via the database. Returns a boolean based on whether the credentials are valid or not.
- `loginScreen()`
    - Main control flow of the login screen. Allows the user to login, register or exit to the main menu.
- `verifyID(id)`
    - Verify if the ID inputted exists, and whether it is a user, artist or both. Returns a value that represents the type of ID.

`usermenu.py`

- `userMenu(uid, con, cursor)`
    - Main control flow of the user menu, including selection of options and exiting back to the main menu.
- `addSession(uid)`
    - Add a session to the current user.
- `endSession(uid, sno)`
    - End the session of the current user.
- `spSearch(input)`
    - Search for songs and playlists based on keywords, and displays these matching songs.
- `findArtist(keywords)`
    - Search for artists based on keywords, and displays these matching artists.
- `getSongs(name)`
    - Get songs that match the given artist name
- `getSongsPlaylists(pid)`
    - Get the songs in the playlist that match the given pid
- `songActions(uid, sno, sid)`
    - Main control flow for song actions, including the execution of all of them.

`artistmenu.py`

- `artistMenu()`
    - Main control flow of the artist menu, including selection of options and exiting back to the main menu.
- `addSong()`
    - Add a song to the database based on the artists' input.
- `findTopStats()`

- ○ Displays the top 3 users who listen to this artists songs, as well as the top 3 playlists containing songs from this artist.

TESTING STRATEGY

Our strategy for testing involved creating a `p1-queries.sql` file which queries rows from our database. We wrote a few test queries in this file to extract pertinent data such as songs, artists and playlists. After running our application, we would send the output of this file into a text file called `output.txt`. We can then check the relevant contents of our database through this file to confirm that our current code works as expected.

Each scenario given in the requirements document was a scenario to be tested (such as starting sessions, searching for songs/playlists/artists, adding songs etc). Other extra scenarios included logging out without ending session, case sensitive checks with searching, manual calculation to determine current top fans and checking the inside of the database to ensure all relations were proper.

Our test cases covered the entire functionality of the requirements, as well as a few edge cases regarding case sensitivity, incorrect input etc.

GROUP WORK BREAK-DOWN

Work items
Lawrence: artists menu, add song from artist, top fans/playlists, design document
Mohammed: login screen, user menu, session start/stop, song/playlist search
Kody: artist search, song actions (listen, info, add to playlist)

Time spent/progress
We each spent approximately 10 hours on our responsibilities. During this time we have fulfilled each of our corresponding work items.

Method of coordination
Discord