# ***Team OP Music***

***Dennis Contreras***

***Charley Wu***

## ***CSCI 201 Final Project***

# Table of Contents

## Project Proposal

Description:

OP Music will be a java application that allows multiple users to interact with a server. The server will maintain a database. The database will store the following data: Song titles, Cumulative Rating of a song averaged based on all users scores, usernames, user passwords, and personal user ratings for each song they choose to rate. The main page will display all songs available for the users to rate. A guest will experience limited functionality within the software and will be able to see the overall scores for all songs and also view the profiles of registered users, if they know their username, to see which songs they have rated and what score they have given that song. A registered user will have all guest functionality and will be able to log in and build a profile of songs based on the songs they have rated.

## Technical Specification

Login/Register (4 hours)

The user will be directed to a login page where they can choose to either log in, register an account, or continue as a guest. If they choose to register, they will need to input a unique username, and a password. The backend will save this information when they log in. Once registered, the users will be logged in and redirected to their album list page (profile page). If they choose to log in, they will need to provide a valid username and password. If valid, they will be redirected to the album list page. If invalid, an error will be displayed, and they will remain on the login page. The backend will handle the form validation. If they choose to continue as a guest, they will be taken to the album list page with guest permissions.

Music List Page (Main Page) (Guest) (3 hours)

The Music list page will list song names. Guests will be able to view the song list and the song ratings. Guests will have the ability to view all songs and rankings but will not have permission to rank any song. Guest can view registered users' profiles as well. They will not have access to a personal profile.

Music List Page + Rating a song (users) (6 hours)

Users will have access to all features that guests have. In addition to that, users will have an exclusive profile page, listing all the music that they have scored. Upon viewing the song list the user can choose to rate a song by typing in the command "add_score" followed by a number in the range 1-10. After scoring, the song will be pulled from the song list and added to the user's list (each user has their own user's list to share with each other called a profile page). This score will also be sent to the server, which will calculate the accumulated scores and rank the song relative to the other songs. The rank is defined by the song's position on the music list page with the song at the top of the list being the highest rank and the last song on the list being the lowest ranked. The user can move from profile page to Music List page by typing in the command "my_profile" and "music_library" respectively.

Profile Page (4 hours)
Users can rate any song provided by the application. Also, any song rated by the user will appear in their profile along with the score they gave that song. In addition, users will be able to view any other user profiles so long as they know their username. The advantage of this is that the user will be able to view the songs other user have rated and whether they like that song or not. The idea is that friend will be able to view each other's profiles and see what they thought about a song.

Ranking (6 hours)
The server will collect scores (1-10) from the users and calculate a cumulative score. It will then give ranks to the songs based on the cumulative scores. The songs scores will be displayed on the Music list page in descending order based on their scores. With their overall score shown alongside the song name. A songs position on the page determines it rank so the first song on the list is the highest ranked and the last song on the list is the lowest ranked. The score shown next to the song name is the average of all user scores.

Navigation (4 hours)
Users will type in commands followed by options to navigate the software. A description of the navigation options and their purpose follows. The verb "Type:" is used for clarity here and should not be typed into the software and the brackets "[ ]" surround a command that is to be specified by the user.

**The following commands apply to guests:**
Type: register [username] [password], provides user registration functionality.

**The following commands apply to guests and registered users:**
Type: login [username] [password], provides login functionality
Type: look_up [username], will allow both users and guests to view the specified user's profile.

**The following commands apply to registered users:**
Type: my_profile, will allow a user to view their profile
Type: add_score [song name] [score], will allow a user to add a score to a song, if that song exists in the music library. The song and the score they gave will be added to their profile and the score given will be factored into the overall score for the song withing the main music list
Type: music_library, will allow the user to view the full song list or music_library, also referred to as the "main page".

# Design Document

<u>Server Class</u>

Variables:

- List<ServerThread> serverThreads // keep track of the server threads created for each guest
- ServerSocket ss

Functions:

- Main // Create a new Server object
- Server() // Constructor, create a new ServerSocket, contain a while true loop to accept incoming guests and create a new ServerThread for each guest

<u>ServerThread Class</u>

Variables:

- PrintWriter dout // write to the guest
- BufferedReader din // read from the guest
- Server sv
- String user // store the username, after the guest logins or registers

Functions:

- ServerThread(Socket s, Server sv) // Constructor, establish connection with the guest and call start()
- printDatabse() // display the music library from the database
- login(String username, String password) // process login request, return 1 if succeeded and 0 if failed, update user if succeed
- register(String username, String password) // process register request, return 1 if succeeded and 0 if failed, update user if succeed
- add_score(String song,String score,String username) // add scoring record to the database (“profiles” table)
- recalcOverallScore(String song,int id) // update cumulative score of a song
- displayProfile(String username) // display user’s profile
- run() // while loop processing requests from the guest and calling the functions accordingly

<u>Guest Class</u>

Variables:

- PrintWriter dout // write to the ServerThread
- BufferedReader din // read from the ServerThread
- Socket

Functions:

- Guest(String hostname, int port) // call start(), contain a while true loop that reads and processes user input and writes to the ServerThread accordingly
- Run() // contain a while true loop that reads from the ServerThread and prints out what is written

MySQL Database Tables

Table 1: profiles

- username varchar(50) ,
- songName varchar(50) ,
- Score float,
- UNIQUE `unique_index`(`username`, `songname`)

| username | songName | Score |
|---|---|---|
| | | |

Table 2: songlist

- songID int primary key not null AUTO_INCREMENT,
- songName varchar(50),
- overallScore float

| songID | songName | overallScore |
|---|---|---|
| | | |

Table 3: usercred

- userID int primary key not null AUTO_INCREMENT,
- userName varchar(50) unique,
- password varchar(50)

| userID | userName | password |
|---|---|---|
| | | |

MySQL Functions & Procedures

- PROCEDURE login (IN name varchar(50), IN code varchar(50), OUT count1 INT)
    - Process login request, return the number of occurrence with matching username and password, 1 = login successful, 0 = login failed
- PROCEDURE register_check(IN name varchar(50), OUT count2 INT)
    - Check for duplicate username for register

# Testing Document

**A Verified working Program will perform accordingly:**

The program should allow the functionality of communication between users/guests and the server. The user/guest will request login access from the server, the server will then authenticate the user in the case that the guest wants to login with their credentials. This should be handled by the server class via the database, which will store the user credentials. If the guest attempts to login with invalid credentials the server will communicate to the user that the credentials were invalid. If the guest does not currently have credentials, the user can choose to register.

In this case the server will allow the user to enter a username and password. The username and password will then be stored in the database, this will be handled by the server. The username must be unique in order to verify that the user does not currently exist in the database and thus a user's credentials will not be overwritten. This can be handled by the database. Both registered users and guests should be able to view the entire music list which will be presented to the user via the server which will pull the entire music library from the database and present it to the user/guest.

Additionally, if a user decides to add a score to a song then the song and score given will be added to the users profile page. However, the overall score of the song will be recalculated with the score given by the user factored in and the overall score of the song will be updated in the database. Hence, the database will store the overall score calculated as an average of all user scores and users' personal profile page should display the exact score the user has given to that song. The users personal profile page will only display songs that the users themselves have added a score to, not the entirety of the music library. Guests will not have access to a personal profile page, thus the guest should not be allowed to add a score and will not have a personal profile page.

**Login/Register:**

| **Test #** | 1. |
|---|---|
| Description | Guest wants to register and becomes a user |
| Input | Enter a username and password and the username does not exist in the database |
| Expected Result | ● Guest is registered and logged in<br>● The guest’s username and password is added to the “usercred” table in the database<br>● Output a success message to the console saying “You are registered!” and “You are logged in!” |

| **Test #** | 2. |
|---|---|
| Description | Guest wants to register and becomes a user |
| Input | Username already exists in the database |
| Expected Result | ● Guest is not registered<br>● Output a failure message to the console saying “Username already exists. Please try again.” |

| **Test #** | 3. |
|---|---|
| Description | Guest wants to register and becomes a user |
| Input | Missing Username or password |
| Expected Result | ● Guest is not registered<br>● Output a failure message to the console saying “Invalid Register Info. Please Try Again.” |

| **Test #** | 4. |
|---|---|
| Description | Guest wants to login |
| Input | Correct username, correct password |
| Expected Result | ● Guest is logged in<br>● Output a success message to the console saying “You are logged in!”<br>● Guest can now use my_profile and add_score |

| **Test #** | 5. |
|---|---|
| Description | Guest wants to login |
| Input | Incorrect username or password |
| Expected Result | ● Guest is not logged in<br>● Output a failure message to the console saying “Login failed. Please try again” |

**Guest browsing**

| **Test #** | 6. |
|---|---|
| Description | Guest connects with the server |
| Input | Run a Guest Thread |
| Expected Result | ● Server pages prints “New guest comes in”<br>● Guest page prints the welcome message, the music library, and the options |

| **Test #** | 7. |
|---|---|
| Description | Guest wants to look up some user’s profile |
| Input | Valid username |
| Expected Result | ● User’s profile is printed out |

| **Test #** | 8. |
|---|---|
| Description | Guest wants to look up some user’s profile |
| Input | A username that does not exist in the database |
| Expected Result | ● Empty profile |

| **Test #** | 9. |
|---|---|
| Description | Guest wants to refresh the page (music library) |
| Input | Music_library request |
| Expected Result | ● Most current music library displayed |

| **Test #** | 10. |
|---|---|
| Description | Guest tries to access his or her profile |
| Input | My_profile request |
| Expected Result | ● Output a failure message saying “You must login in first” |

| **Test #** | 11. |
|---|---|
| Description | User tries to add score to a song |
| Input | Add_score request |
| Expected Result | • Output a failure message saying “You must login in order to give a score” |

**User Browsing**

| **Test #** | 12. |
|---|---|
| Description | User tries to add a score to a song |
| Input | Invalid song name |
| Expected Result | • Failure message printed notifying the user that song does not exist in the database |

| **Test #** | 13. |
|---|---|
| Description | User tries to add a score to a song |
| Input | Valid song name with a score |
| Expected Result | • Success message saying “You have scored”<br>• The record is added to “profiles” table in database<br>• Song and score will be displayed when the user enters my_profile<br>• The cumulative score of the song is updated |

| **Test #** | 14. |
|---|---|
| Description | User tries to access his or her profile |
| Input | My_profile |
| Expected Result | • User’s profile is displayed |

## Deployment Document

1. Setting up the database
    - Open MySQL Workbench and go inside of your local host
    - Click on File -> Open SQL script and select FP Database.sql from the project folder
    - Execute FP Database.sql (the entire file)
    - For testing, you can add more songs into the songlist table through SQL query (Ex. INSERT INTO songlist (songname, overallscore) VALUES (‘a’, 5))
    - Please make sure that the songName entered is one string (Ex.“Hello_world”) and that the score is between 1 and 10
2. Setting up Eclipse
    - Open Eclipse
    - Click on File -> Open Projects from File System
    - Choose the Final_Project folder and click finish
    - Make sure that you have the Java MySQL Connector JAR in the project’s class path
    - If you don’t, you can download it here: https://dev.mysql.com/downloads/connector/j/
    - To add the JAR file, right click on your Java Project -> Properties -> Buildpath -> Libraries -> Add External JAR and Select the JAR file you downloaded (Ex. mysql-connector-java-8.0.22-bin.jar)
3. Run the program
    - Open Guest.java, Server.java, and ServerThread.java
    - Run Server.java as a java program to set up the server
    - Run Guest.java multiple times to create some number of guests
    - Type requests inside of the console for Guest.java