Title: CS 411 Music Recommendation System

Summary/Description: We want to build a user-generated music recommendation system. The idea of the application is that users can get song recommendations from our database based on the genres of music that they listen to and are interested in. We want to foster a strong community of individuals that can stay engaged through the song ranking system we propose as a functionality.

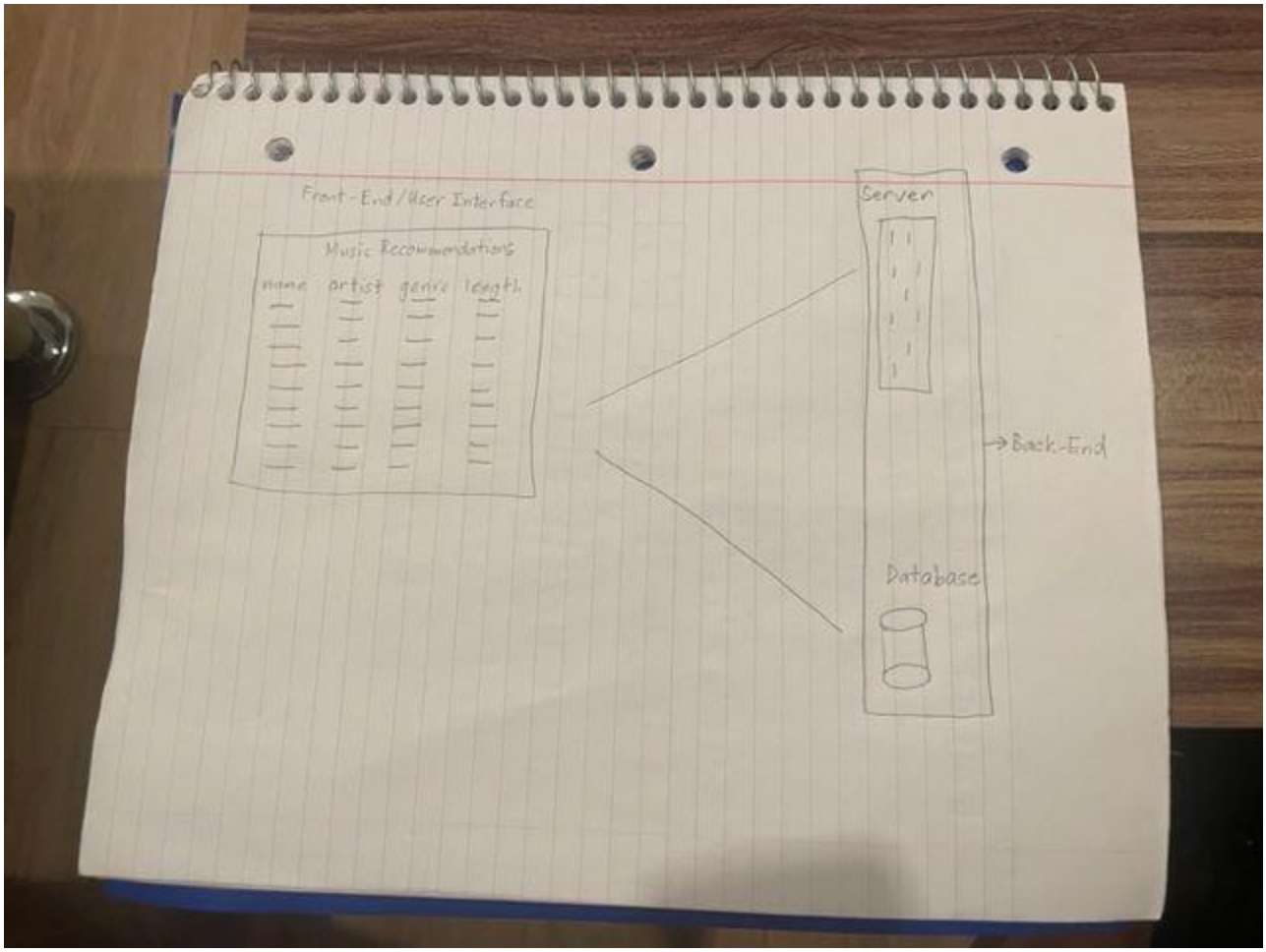
Usefulness: This music recommendation system can be a powerful tool for users to rate their song choices thereby allowing users accessing the database to be informed of song recommendations based on their listening preferences. Additionally, users are exposed to music across genres and across regions, allowing them to widen their listening palette.

Realness: In our database, we will be storing a collection of popular music. There are a few CSV files on Kaggle containing data of Spotify's hits throughout the decades that we can use as the starting point for our application. The following attributes can be considered important information when it comes to storing records of songs: song name, artist, genre, duration, release year, explicitness, popularity, tempo, etc. Here are links to some of the datasets we found: https://www.kaggle.com/datasets/kapturovalexander/spotify-data-from-pyspark-course, https://www.kaggle.com/datasets/josephinelsy/spotify-top-hit-playlist-2010-2022.

Functionalities: The basic functionalities of our application include users searching for song recommendations based on a certain set of attributes (genre, artist, etc.). We can also give users the option to input any songs that they feel people can be interested in.

One cool feature that we think we could implement is a functionality where users can rate songs that they listen to (1-10, 0-5 stars, etc.). When a user gives a good rating for a song, it is more likely to get recommended to people who are looking for a new song to enjoy. We can achieve this by editing our database to allow for an option where users can input a rating for each song they listen to, and therefore take the average rating and recommend based off of the number.

Below is a picture of how we believe our user interface to look like:



Our front-end user interface will most likely be a React.js webpage that will interact with a server and a SQL database on the backend. The webpage consists of all pertinent information regarding the database's collection of songs, and will display the attributes that users search for. SQL queries made through the backend will either insert information regarding music into the database or return records for people to peruse.

Work Distribution: These responsibilities are not set yet, but ideally we would have two people working on the front-end (React.js webpage) and two working on the backend database. Each person will work on both, but one aspect will be the primary aspect. This is a breakdown of the work distribution for each functionality.

React.js Front-end: Kevin, Aakarsh
SQL Back-end: Aniketh, Asuthosh
Creation, Insertion, and Retrieval of Records: Kevin, Asuthosh
Rating System: Aniketh, Aakarsh