

# CMPSC 250 Final Project: Final Report

Instructor: Dr. Mohan

*Xingbang Liu, Matt Jones, Travis Thomas*

*02 May 2018*

---

## Motivation

Our team was inspired by existing music suggestion platforms (ie. Spotify, Pandora, Apple Music, etc.) to mimic their systems and add our own creative twist on current song suggesting algorithms. This includes features similar to other platforms including: “favorites” or “liked music”, “DailyMix” and “Recommended for You”, sorting features such as “by artist” and “by genre”, and a “cloud database\*” that would hold the list of all possible songs that the user can choose from.

Our work also builds directly off the previous work submitted in this course, as well as building off existing algorithms in the field. We built directly off of the music-sorting lab we submitted earlier this year, to sort the users liked songs. The creative twist we added to our algorithm suggestion takes a look at how we listen to music from a different angle. Because music provides a “rhythm or beat” to many of the things we do (whether it’s dancing, work, etc.), we decided to try to rate a user based on the speed of the rhythm (also known as Tempo).

## Why is it important

This work acts as valuable research to anyone in the music industry. Specifically, anyone who is looking at new ways to incorporate aspects of songs into a user-song-suggestion platform. While current music platforms are doing a good job of suggesting songs (usually based off other real peoples songs), there’s still songs that users find out-of-the-blue on their own that they really enjoy. Exploring new ways to relate aspects of songs to the user can only help find these outlying songs, and suggest them to the user.

## How is it useful

Millions of people use music everyday. Whether it’s to wake up to their favorite song in the morning, make their commute to their job better, or provide some-type of rhythm to everyday tasks, music makes our lives better! So, a tool which helps you find music you like, and save music you like would undoubtedly be useful to a lot of people.

## Background for problem

There are many algorithms to find similar songs. Here are two examples of the Collaborative filtering:

### Memory-based

Generally, user would be grouped in this algorithm. In this method, the algorithm will first track down people who have similar activities, then recommend songs to each other by their playlists. For example, action score could be assigned as:

Repeat song	Share	Like	Play	Played completely	Skip	Dislike
5	4	3	2	1	-1	-5

Then, a person's preference would be a N-Dimensional vector. Here N is the number of songs by default. By using the cosine of the vector angle, which is generated by two vectors, we can know how similar two users can be. The cosine of 0 degree, which means two people are exactly the same, is 1. The cosine of 180 degree, which means two people have the opposite preference, is -1. The equation is:

Assume we have vector  $\vec{a}$  and  $\vec{b}$ ,

$$\cos(A) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \times |\vec{b}|}$$

This prediction method is very accurate, but there are too many calculations and it is not accurate for a new user.

### Model-based

As for the second method, the algorithm will analyze the playlist of a person, then do recommendations. Models such as the latent factor are used in this method. In latent factor model, user's like list will be analyzed by dividing into different tag factors, for example, classic and indie. By signing scores for different factors, we can have a matrix which can be divided into two different matrices:

$$\begin{aligned} R &= QP^T \\ R_{user,song} &= Q_{user,factor} P_{factor,song}^T \\ &= \begin{pmatrix} q_{user_1,factor_1} & q_{user_1,factor_2} & \cdots & q_{user_1,factor_n} \\ q_{user_2,factor_1} & q_{user_2,factor_2} & \cdots & q_{user_2,factor_n} \\ \vdots & \vdots & \ddots & \vdots \\ q_{user_n,factor_1} & q_{user_n,factor_2} & \cdots & q_{user_n,factor_n} \end{pmatrix} \begin{pmatrix} p_{factor_1,song_1} & p_{factor_1,song_2} & \cdots & p_{factor_1,song_n} \\ p_{factor_2,song_1} & p_{factor_2,song_2} & \cdots & p_{factor_2,song_n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{factor_n,song_1} & p_{factor_n,song_2} & \cdots & p_{factor_n,song_n} \end{pmatrix}^T \end{aligned}$$

From the matrix R, we can get estimated score. After that, we can ignore the song which the user already listened, and recommend him the song he has not listened.

Of course, in the real world, if we implement algorithm based on song, the second dimension factor will rapidly increase. Plus there will be too many songs. Therefore, it will be more efficient if we calculate based on tags of the song. For example, for "The Sound of Silence" by Simon & Garfunkel, tags can be Folk Rock, 60s, and Movie Track. Due to the time we have is limited, we would not build tags databases, therefore we would analyze by songs.

In the real situation, based on different requirements, we can pick one of them or use them together. It is also an option to mix these two algorithms and form a new hybrid algorithm when there are not enough songs in a playlist.

### How could it be improved

These algorithms are not perfect, they could be limited when there are not enough users. For example, if a start up music service company wants to suggest songs to user, they have to use an algorithm which works when there are not enough users and user action data. In this situation, it would be better if the company could depend on the music only, for example, music metadata, rather than depend on user data. In our algorithm, we hard coded melody scores, From 1 to 10, the faster the higher, for each songs. This score

simulated music classification data. By getting a mean score from user liked list, we can find more songs by their mean score. This method is more like a weaker version of the combination of nature language processing and raw audio analyze. In nature language processing, the program would analyze the lyrics, then get the emotion data. In raw audio analyze, the program would analyze the audio frames, and finally get an score for audio characteristic.

## Program and Algorithm Description

### Pseudocode

```
For each liked song:
    append to likedlist;
mean = (sum of melodies)/likedListLength;
for exploreSong['melody'] == mean:
    return song;
```

This is a very basic description of the algorithm but it explains the though process behind the algorithm used to return songs. Essentially, we ask the user pick songs that they like. This will add those liked songs to the liked list. Using, the songs from the liked list, we find the mean of the melodies within the liked list. This mean is then used to search the main music database and return a set of songs that match the mean melody score of the liked list. The melody score is just one way we can return a set of recommended songs, there are other variables that can be and will be included in the future. The inclusion of these variables will not only give us a more desirable list of recommended songs for user but will also allow for options when returning a set of songs. For example, if a user wants a playlist to workout to, we may want to return songs with high melody scores that are more upbeat.

### Diagram

**Figure 1** is a basic outline of the interaction between classes in our program. The `music.csv` file is fed into `explore.java`, returning random songs with its `explore` function and also sends the liked songs to `liked.csv` with its `add` function. The liked songs are used by `DailySelection.java` and `Like.java`. `DailySelection` uses the algorithm discussed in the pseudocode to return songs that match the mean melody rating of the songs found in the liked list. The `Like` class sorts the liked list by either song, genre, artist, or year. `Like.java` calls `SortInt` to sort the year and `SortString.java` to sort the other categories. `MusicSug.java` is the main method and gives the user a command-line interface to interact with music recommender system, calling the functions mentioned above.

### Time Complexities

**Figure2** Represents the time complexities of the Bubble sort and Exchange sort after four runs. `SortInt.java` utilizes the bubble sort method to sort the year songs within the liked list. `SortString.java` utilizes exchange sort to sort each of the other categories. We found a worst-case time complexity of  $O(n^2)$  for both sorting algorithms. We decided to used these algorithms because we do not anticipate large scale growth in our liked list in its current state. That is the program only accounts for a single user and the user will not produce enough songs on the liked list to mandate a different algorithm.

## Conclusion

### Results

While analyzing our results, note that our program doesn't take into account things like artist, genre, instruments used, etc. when suggesting songs. However, the program does take a unique approach to song suggesting, and runs without any bugs (caused by the user and/or built into our system).

**Figure 3** shows the Main Menu of our program. It offers four options: Like (prints liked songs), Explore (Finds and adds new songs to the users liked songs), Daily Selection (Makes a playlist similar to the songs in the users liked-list), Quit (Exits the program)

**Figure 4** shows the first feature: Print Liked Songs & Sort them.

**Figure 5** shows the second feature: Finds 5 songs for the user, and the user picks a song (in this case, song 3) to add to their liked songs. Then **figure 6** shows the updated liked songs with new addition from feature #2.

**Figure 7** shows the third feature: Gives you two song selections similar to your liked list tempo, and then the user adds these songs to their liked songs.

**Figure 8** shows updated liked songs with new addition from feature #3.

**Figure 9** shows the fourth feature: Quit/exits the program.

## Takeaways

There are a few notable takeaways from researching and implementing our final project. First, collaborative filtering algorithms are complex and take into account many, many variables. There are many components that make up a song, thus creating accurate predictions requires a lot of data. Second, collaborative is difficult to implement if multiple users are not involved. Our program only accounts for a single user and single liked list. An optimal implementation would call for multiple user's liked lists to be compared across the program, to find similar user interests so we can recommend songs based off of similar user's playlists and interests. Overall, our program is a solid first-step in implementing a music recommender that is capable of taking in unique parameters, such as melody score, to recommend songs to users.

## Challenges

There were a few obvious problems that our group encountered that was to be expected. Because our program has many different working parts, we had to split it up and then either comment the code in a way so that our other group members could explain it. Alternatively to commenting every bit of code, we also met many times to try to explain our code to the rest of the group. The biggest challenge our group had was working with the "cloud music list" (aka our Music.csv) and figuring out how to edit this list without messing up the csv file and the rest of our program. Our solution was to make a "working csv file" (Test.csv) that was the file we actually made edits to, while the "cloud file" (Music.csv) was left alone so that the edits didn't cause any problems with the rest of the program.

## Rewards

The process of this project has been rewarding at every step. The research that went into the project to learn about collaborative filtering and common algorithms used to recommend songs was interesting as we all frequently use these music apps. So, learning how these apps recommend daily playlists and songs that we may like is intriguing. Implementation was rewarding in its challenges. Every obstacle overcome was its own reward. We have newfound respect for programmers that are responsible for implementing and maintaining these algorithms because we know the amount of data and variables that goes into proper implementation.

## Index

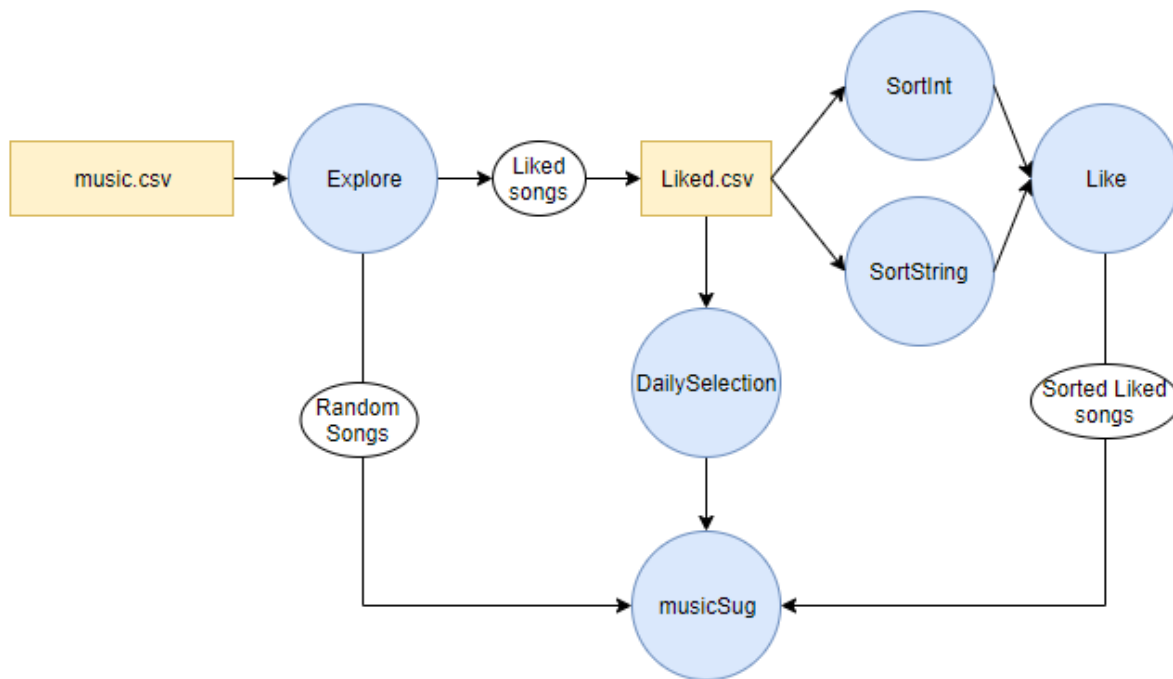


Figure 1: diagram

	in nano seconds	bubble sort	exchange sort
1.	186,525	106,856	
2.	156,695	111,009	
3.	156,318	110,631	
4.	239,386	110,254	
mean	184,731	109,688	
	$O(n^2)$		

Figure 2: Time Complexities

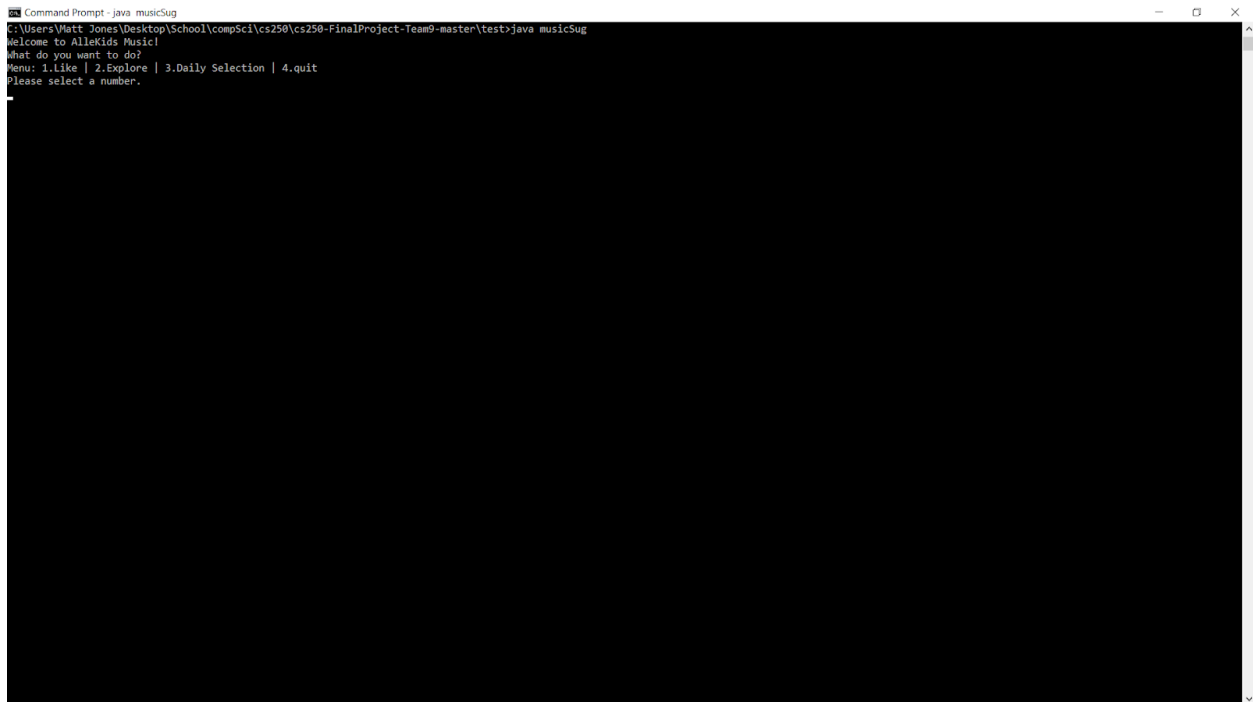


Figure 3: menu

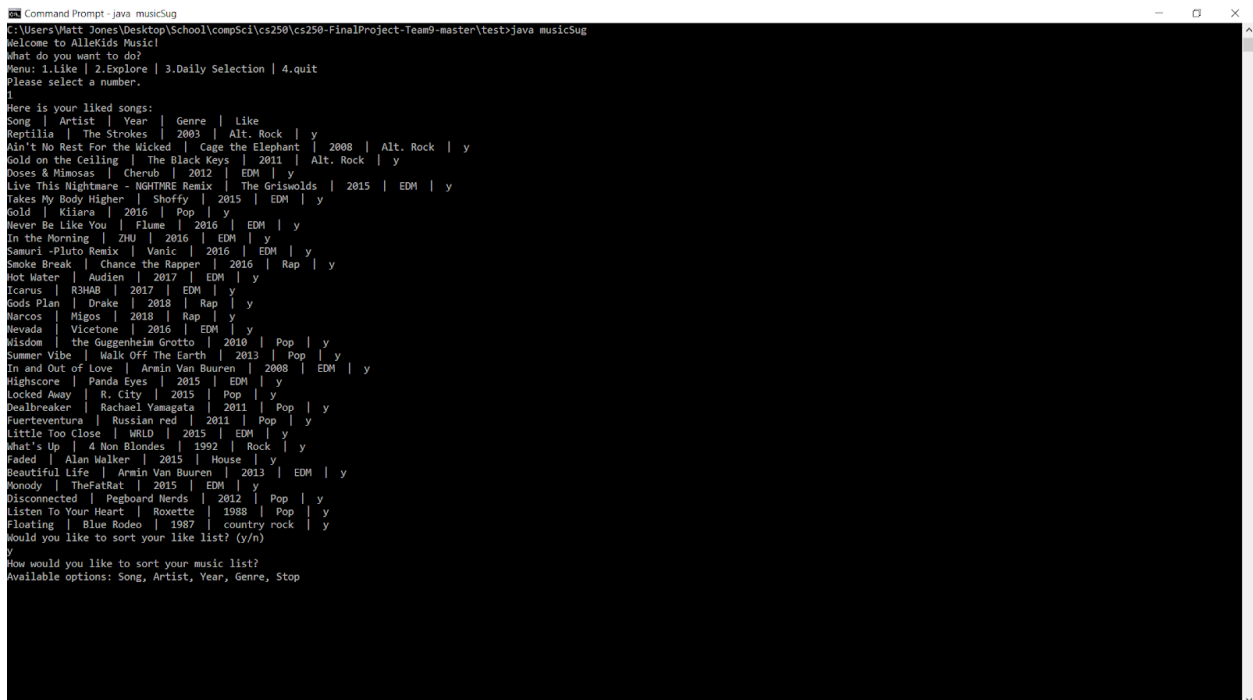


Figure 4: print Liked Songs & Sort them

```
Command Prompt - java musicSug
n
C:\Users\Matt Jones\Desktop\School\compSci\cs250\cs250-FinalProject-Team9-master\test>java musicSug
Welcome to AlleKids Music!
What do you want to do?
Menu: 1.Like | 2.Explore | 3.Daily Selection | 4.quit
Please select a number.
2
Song Artist Year Genre Like
1: Dying Under A Binary Star | Crematorium | 2005 | death core | n |
2: I denna ljuva sommartid | Sofia Karlsson | 2002 | chanson | n |
3: Baby Momma (Chopped & Screwed Version) | Lil Boosie & Webbie | 0 | gangster rap | n |
4: Touch You | Sandy Posey | 1966 | ballad | n |
5: White Jetta | Casiotone For The Painfully Alone | 2009 | trip hop | n |
Do you like any of these songs?(y/n)
y
What is the number of the song?
3
Baby Momma (Chopped & Screwed Version) | Lil Boosie & Webbie | 0 | gangster rap | y
The song has been added to your like list!
You are at menu now
```

Figure 5: explore

```
Command Prompt - java musicSug
C:\Users\Matt Jones\Desktop\School\compSci\cs250\cs250-FinalProject-Team9-master\test>java musicSug
Welcome to AlleKids Music!
What do you want to do?
Menu: 1.Like | 2.Explore | 3.Daily Selection | 4.quit
Please select a number.
1
Here is your liked songs:
Song Artist Year Genre Like
Reptilia | The Strokes | 2003 | Alt. Rock | y
Ain't No Rest For the Wicked | Cage the Elephant | 2008 | Alt. Rock | y
Gold on the Ceiling | The Black Keys | 2011 | Alt. Rock | y
Doses & Mimosas | Cherub | 2012 | EDM | y
Live This Nightmare - NIGHTMRE Remix | The Griswolds | 2015 | EDM | y
Takes My Body Higher | Shoffy | 2015 | EDM | y
Gold | Kiiara | 2016 | Pop | y
Never Be Like You | Flume | 2016 | EDM | y
In the Morning | DJJ | 2016 | EDM | y
Samurai -Pluto Remix | Vanic | 2016 | EDM | y
Smoke Break | Chance the Rapper | 2016 | Rap | y
Hot Water | Audien | 2017 | EDM | y
Icarus | R3HAB | 2017 | EDM | y
Gods Plan | Drake | 2018 | Rap | y
Narcos | Migos | 2018 | Rap | y
Nevada | Vicetone | 2016 | EDM | y
Wisdom | the Guggenheim Grotto | 2010 | Pop | y
Summer Vibe | Walk Off The Earth | 2013 | Pop | y
In and Out of Love | Armin Van Buuren | 2008 | EDM | y
Nightcore | Panda Eyes | 2015 | EDM | y
Locked Away | R. City | 2015 | Pop | y
Dealbreaker | Rachael Yamagata | 2011 | Pop | y
Fuerteventura | Russian red | 2011 | Pop | y
Little Too Close | WRLD | 2015 | EDM | y
What's Up | 4 Non Blondes | 1992 | Rock | y
Faded | Alan Walker | 2015 | House | y
Beautiful Life | Armin Van Buuren | 2013 | EDM | y
Monody | TheFatRat | 2015 | EDM | y
Disconnected | Pegboard Nerds | 2012 | Pop | y
Listen To Your Heart | Roxette | 1988 | Pop | y
Floating | Blue Rodeo | 1987 | country/rock | y
Baby Momma (Chopped & Screwed Version) | Lil Boosie & Webbie | 0 | gangster rap | y
Would you like to sort your like list? (y/n)
```

Figure 6: updated liked songs

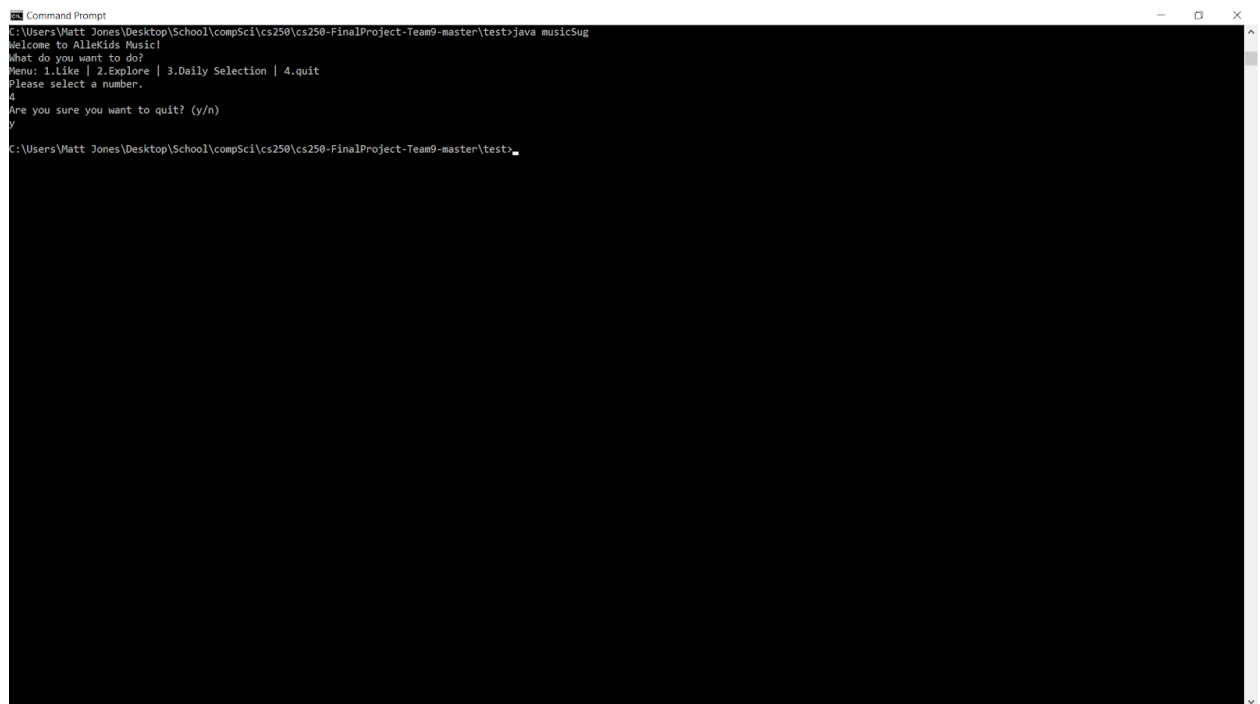
```
Command Prompt - java musicSug
C:\Users\Matt Jones\Desktop\cs250-FinalProject-Team9-master\test>java musicSug
Welcome to All4Kids Music!
What do you want to do?
Menu: 1.Like | 2.Explore | 3.Daily Selection | 4.quit
Please select a number.
3
Song | Artist | Year | Genre | Like
1: Spanish Grease | Willie Bobo | 1997 | latin jazz | n |
2: The Moon And I (Ordinary Day Album Version) | Jeff And Sheri Easter | 0 | southern gospel | n |
Do you like any of these songs?(y/n)
y
What is the number of the song?
1
Spanish Grease | Willie Bobo | 1997 | latin jazz | y
The song has been added to your like list!
You are at menu now
```

Figure 7: suggestion

```
Command Prompt - java musicSug
C:\Users\Matt Jones\Desktop\cs250-FinalProject-Team9-master\test>java musicSug
Welcome to All4Kids Music!
What do you want to do?
Menu: 1.Like | 2.Explore | 3.Daily Selection | 4.quit
Please select a number.
1
Here is your liked songs:
Song | Artist | Year | Genre | Like
Reptilia | The Strokes | 2003 | Alt. Rock | y
Ain't No Rest For the Wicked | Cage the Elephant | 2008 | Alt. Rock | y
Gold on the Ceiling | The Black Keys | 2011 | Alt. Rock | y
Doses & Mimosas | Cherub | 2012 | EDM | y
Live This Nightmare - NIGHTMRE Remix | The Griswolds | 2015 | EDM | y
Takes My Body Higher | Shoffy | 2015 | EDM | y
Gold | Kiiara | 2016 | Pop | y
Never Be Like You | Flume | 2016 | EDM | y
In the Morning | DJJ | 2016 | EDM | y
Samuri -Pluto Remix | Vanic | 2016 | EDM | y
Smoke Break | Chance the Rapper | 2016 | Rap | y
Hot Water | Audien | 2017 | EDM | y
Icarus | R3HAB | 2017 | EDM | y
Gods Plan | Drake | 2018 | Rap | y
Narcos | Migos | 2018 | Rap | y
Nevada | Vicetone | 2016 | EDM | y
Wisdom | the Guggenheim Grotto | 2010 | Pop | y
Summer Vibe | Walk Off The Earth | 2013 | Pop | y
In and Out of Love | Armin Van Buuren | 2008 | EDM | y
Nightscore | Panda Eyes | 2015 | EDM | y
Locked Away | R. City | 2015 | Pop | y
Dealbreaker | Rachael Yamagata | 2011 | Pop | y
Fuerteventura | Russian red | 2011 | Pop | y
Little Too Close | WRLD | 2015 | EDM | y
What's Up | 4 Non Blondes | 1992 | Rock | y
Faded | Alan Walker | 2015 | House | y
Beautiful Life | Armin Van Buuren | 2013 | EDM | y
Monody | TheFatRat | 2015 | EDM | y
Disconnected | Pegboard Nerds | 2012 | Pop | y
Listen To Your Heart | Roxette | 1988 | Pop | y
Floating | Blue Rodeo | 1987 | country/rock | y
Spanish Grease | Willie Bobo | 1997 | latin jazz | y
Would you like to sort your like list? (y/n)
```

Figure 8: updated liked songs





```
Command Prompt
C:\Users\Matt Jones\Desktop\School\compSci\cs250\FinalProject-Team9-master\test>java musicSug
Welcome to AlleKids Music!
What do you want to do?
Menu: 1.Like | 2.Explore | 3.Daily Selection | 4.quit
Please select a number.
4
Are you sure you want to quit? (y/n)
y
C:\Users\Matt Jones\Desktop\School\compSci\cs250\FinalProject-Team9-master\test>
```

Figure 9: Quit/exits