# CIS 550 Project Proposal

Drive link: https://docs.google.com/document/d/1lfTHYFdZrr6g\_PkYuKMZsm29zeagxvALbBES7BU3Yoc/edit#heading=h.htqkuo31ph70

### Group members

|  |  |  |
| --- | --- | --- |
| **Name** | **Email** | **GitHub Username** |
| Mingwei Ruan | ruanm@seas.upenn.edu | mruan238 |
| Tsang Fonda Helen | fonda@seas.upenn.edu | fondatsang |
| Haoning Gong | ghaoning@seas.upenn.edu | ghaoning |
| Huaying Gu | huaying@seas.upenn.edu | kkkhora |

### Application/Website Description

This project will implement a website that provides following features with regard to Spotify:

1. **Choose from sad/happy playlists**. This module aims to display playlists that contain sad or happy songs. Those playlists are retrieved by sending a GET request to Spotify with keywords like “sad song” and “happy song”.
2. **Like/Dislike**. This module allows users to like or dislike a song by clicking the button on the page. If a user likes a song, the song will be added to a playlist named *My Favorite*. And if a user dislikes a song, the song will be removed from that playlist.
3. **Create a playlist**. This module allows users to add and remove songs from a playlist by clicking the button on the page. And users can add some descriptive information towards the playlist.
4. **Search for a song**. Users can search for a song from an existing list by input song name/singer name/year, etc.
5. **Recommendation for similar song/artist**. We will implement a recommendation algorithm to display a list of similar songs to what the user has chosen.
6. **Random-generated playlist**. This module allows users to generate a random list of songs.

### Dataset Description

#### Dataset 1

|  |  |
| --- | --- |
| **Table Name** | song\_artist\_album |
| **Source** | spotify.com |
| **Size** |  |
| **Number of Rows** | 36407 |
| **Number of Attributes** | 9 |

(metadata and statistics here)

|  |  |  |
| --- | --- | --- |
| **song\_artist\_album**  Based on the given unique songID in the kaggle dataset (detailed information of which is shown below), we pulled data from spotify to get relevant information about artist and album of the song. We are using a third-party tool called ‘spotipy’ which uses spotify api to get data. | **Attribute** | **Type** |
| songID | varchar |
| song name | varchar |
| song popularity | int |
| artist name | varchar |
| artist id | varchar |
| artist genres | varchar |
| album name | varchar |
| album id | varchar |
| album release date | varchar |

#### Dataset 2

|  |  |
| --- | --- |
| **Table Name** | song\_detail |
| **Source** | Kaggle |
| **Size** | 3.9MB |
| **Number of Rows** | 36407 |
| **Number of Attributes** | 13 |

(metadata and statistics here)

|  |  |  |
| --- | --- | --- |
| **Songs Features**  Spotify's API was used to search for playlists with happy and sad words, for example: (happy, dance, great) and (sad, crying, depressing). The h/s column shows whether the songs came from a happy or sad playlist. 1 for happy, 0 for sad.  Reference for other columns of audio features can be found here:  https://developer.spotify.com/documentation/web-api/reference/#/operations/get-audio-features | **Attribute** | **Type** |
| uri(AS songID) | varchar |
| acousticness | float |
| valence | float |
| danceability | float |
| energy | float |
| key | int |
| instrumentalness | float |
| liveness | float |
| loudness | float |
| speechiness | float |
| tempo | float |
| duration | int |
| h/s(happy or sad mood) | char(1) |

### Queries to Perform

#### Query 1: Choose from sad/happy playlists

SELECT list\_name, creator

FROM table\_of\_sad\_playlist, table\_of\_happy\_playlist

LIMIT 10;

#### Query 2: Like/Dislike

SELECT song\_name, artist, year

FROM table\_of\_songs

WHERE likes = ‘Yes’;

#### Query 3: Create a playlist

SELECT song\_name, artist, year

FROM table\_of\_songs

WHERE playlist = ‘name\_of\_playlist’;

#### Query 4: Search for a song

SELECT song\_name, artist, year

FROM table\_of\_songs

WHERE

song\_name LIKE ‘%song\_name%’ OR

artist LIKE ‘%artist\_name%’

LIMIT 100;

#### Query 5: Recommendation for similar song/artist

SELECT a.song\_name, a.artist, a.year

FROM table\_of\_songs a JOIN kaggle\_dataset b ON a.song\_id = b.song\_id

WHERE

DIFF(b.acousticness, target\_acousticness) < 0.1 AND

DIFF(b.valence, target\_valence) < 0.1 AND

DIFF(b.danceability , target\_danceability) < 0.1 AND

DIFF(b.energy, target\_energy) < 0.1

LIMIT 10;

#### Query 6: Random-generated playlist

SELECT song\_name, artist, year

FROM table\_of\_songs

WHERE song\_id IN random\_song\_id

LIMIT 10;