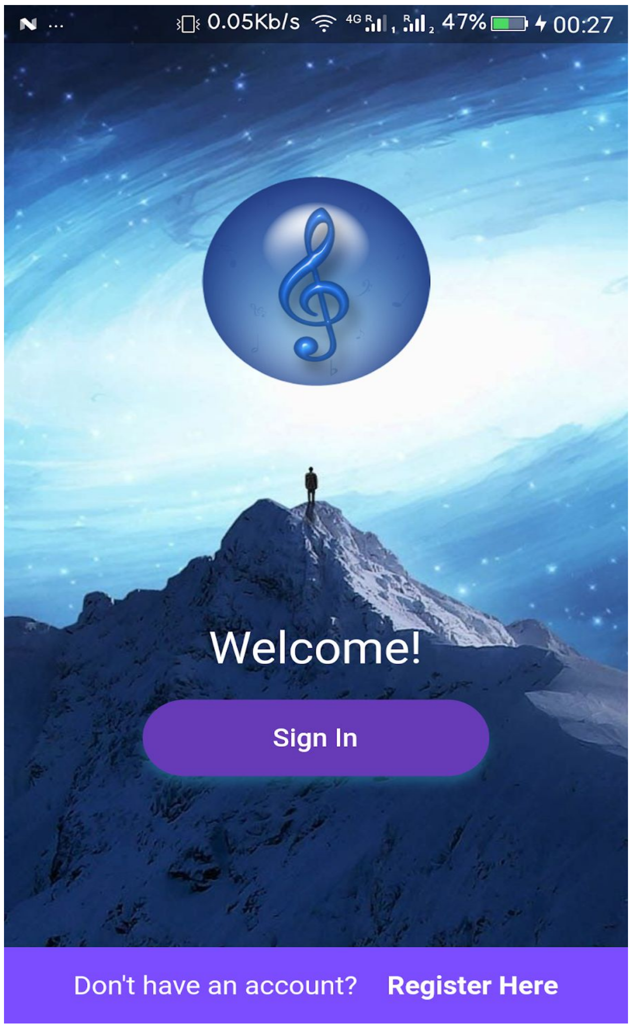
MusiQuest

Rajat Rathi	160050015
Abhro Bhuniya	160050017
Tummidi Nikhil	160050096
Gurparkash Singh	160050112

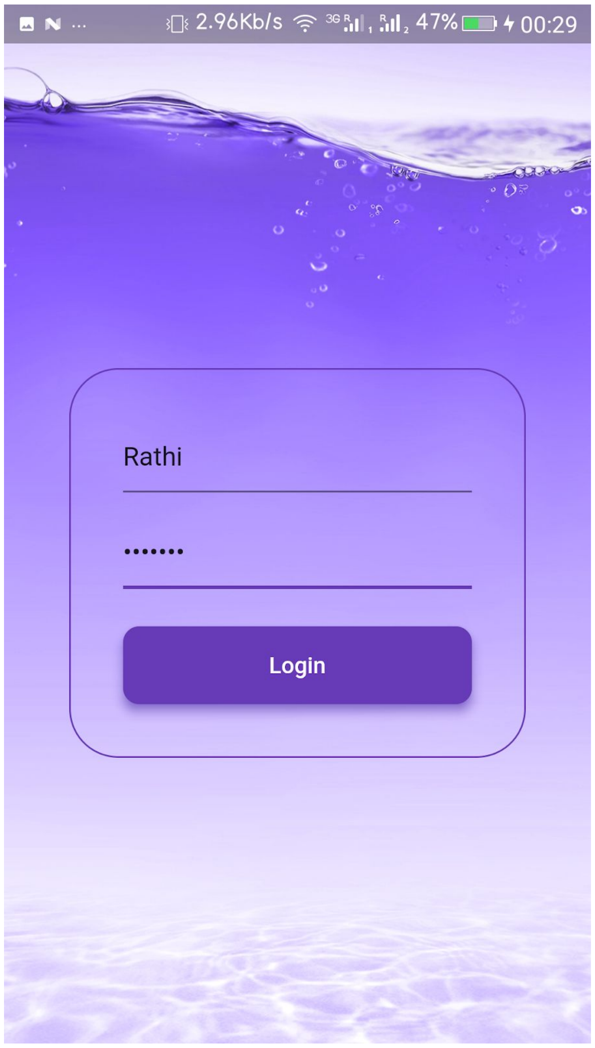
OVERVIEW

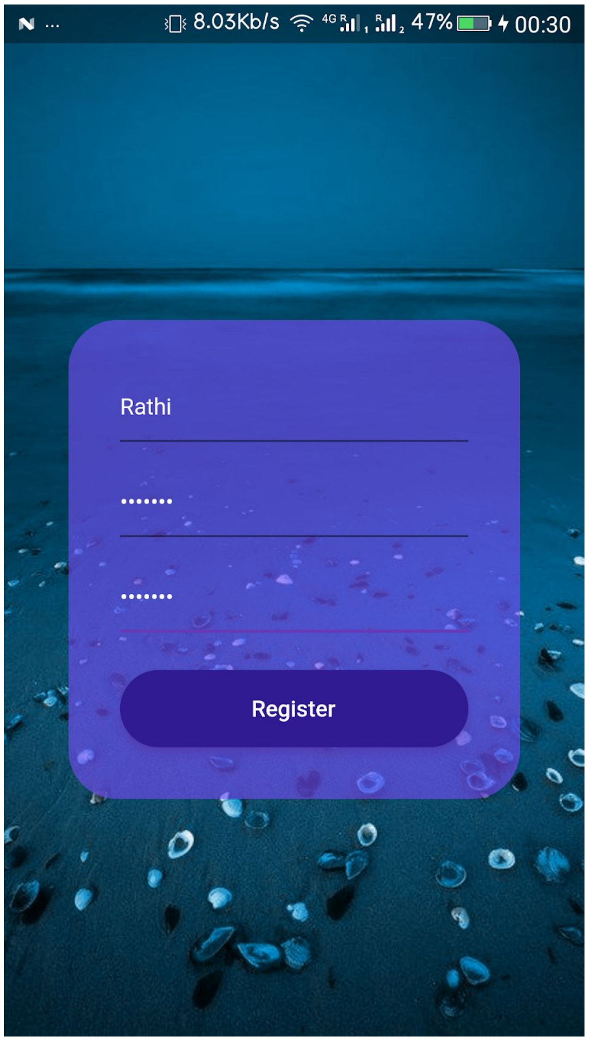
MusiQuest is a Flutter application, which primarily serves as a general purpose application to listen to / search for music similar to various other apps such as *Spotify* and *Saavn*. Similar to these apps we have also build a recommendation system which suggests songs for the user. What sets us apart from all the available apps is that we also provide a provision for adding lyrics for songs as well as searching over the lyrics.



FUNCTIONAL & INTERFACE SPECIFICATIONS

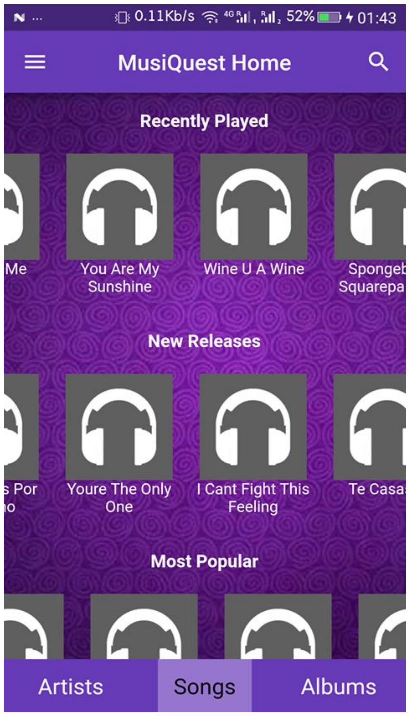
- **Login Screen / Register Screen**
  - When the user signs in the main screen, he/she is greeted with a login screen on which he/she can log in with his username and password.
  - If the user is not registered, he/she can go to register screen from the main screen.

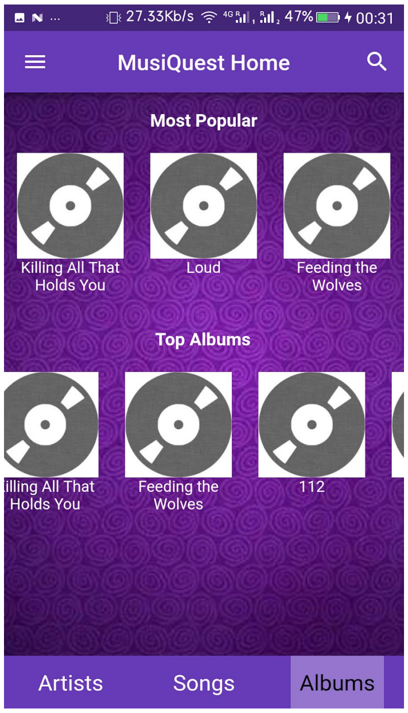


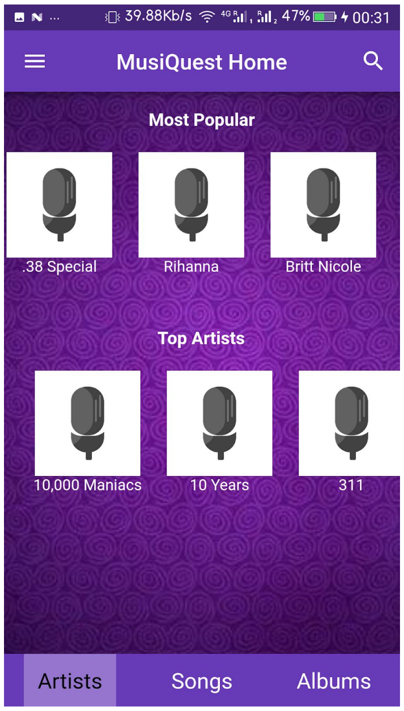


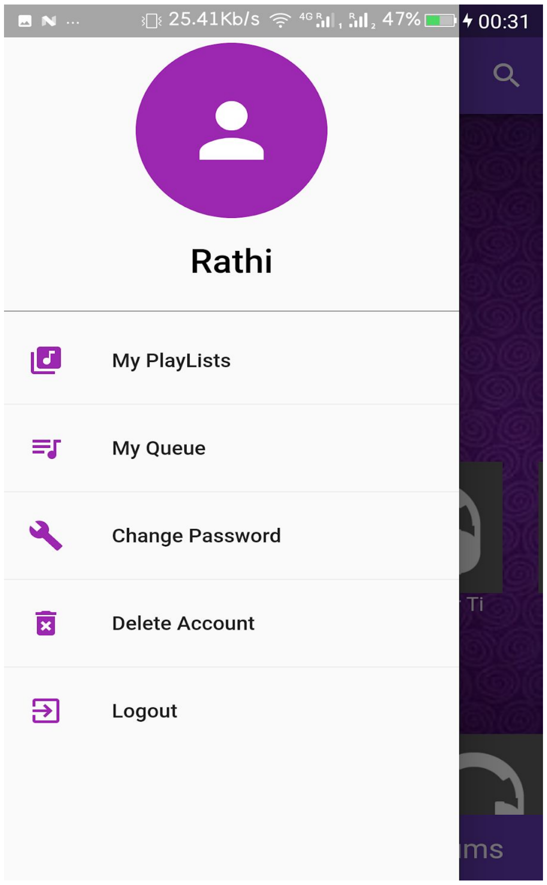
- **Homepage**
  - On the homepage, we display the most popular and most liked songs, albums and artists, which can be explored via the bottom panel. We also display the most recently played songs (by the user) and the latest released (in terms of release date) songs.

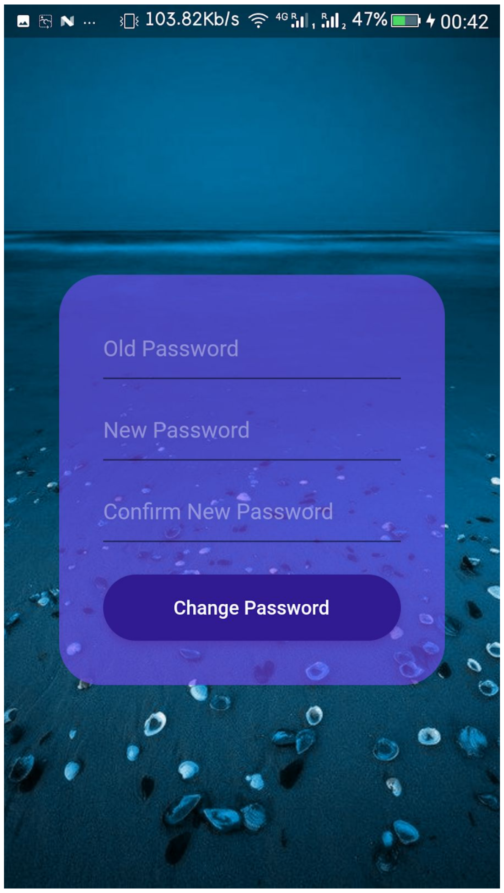
- A side navigation panel links the user to the *Logout, Change Password* and *Delete Account* options, and his personal library (*My Queue* and *My Playlists).*



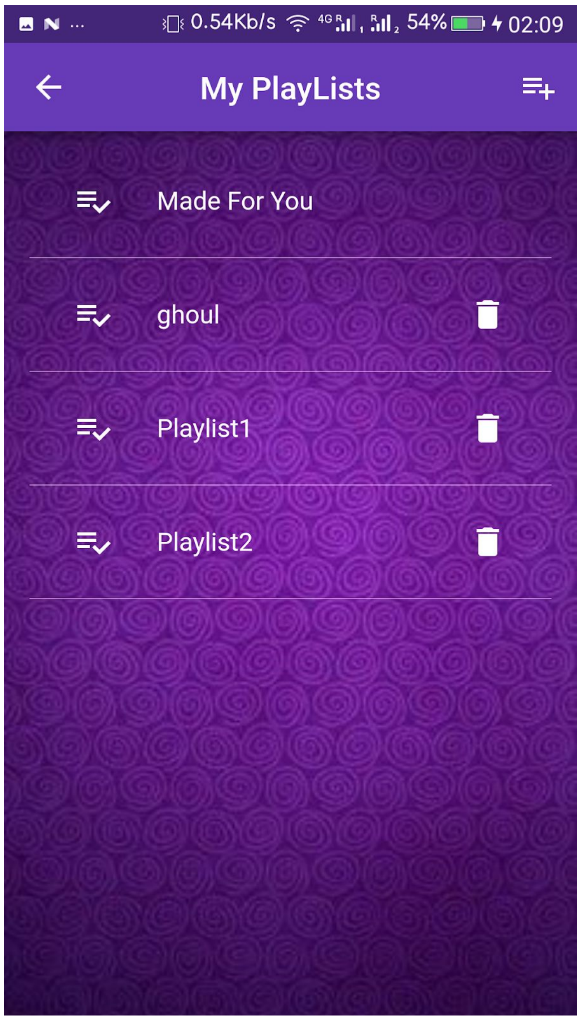


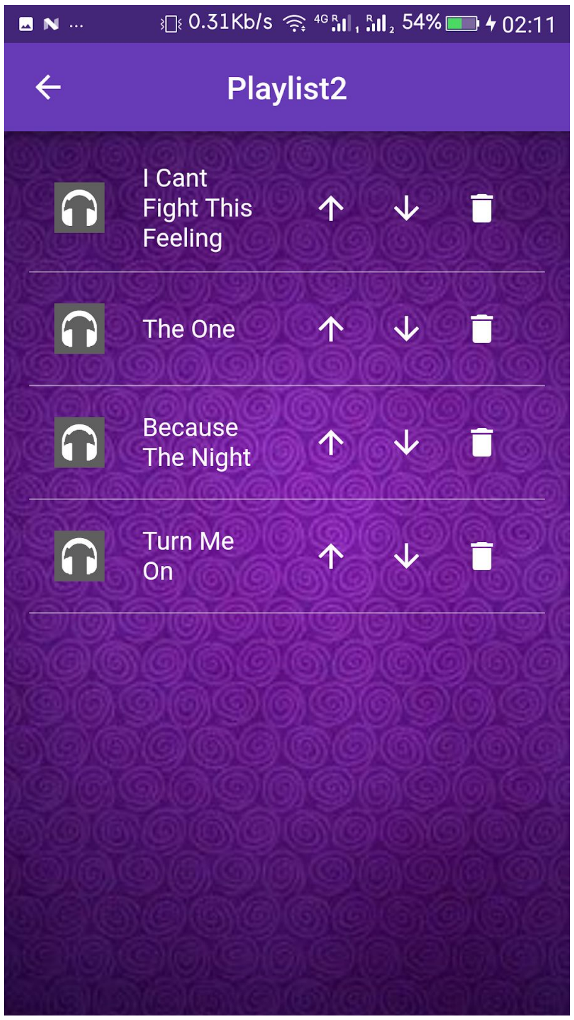


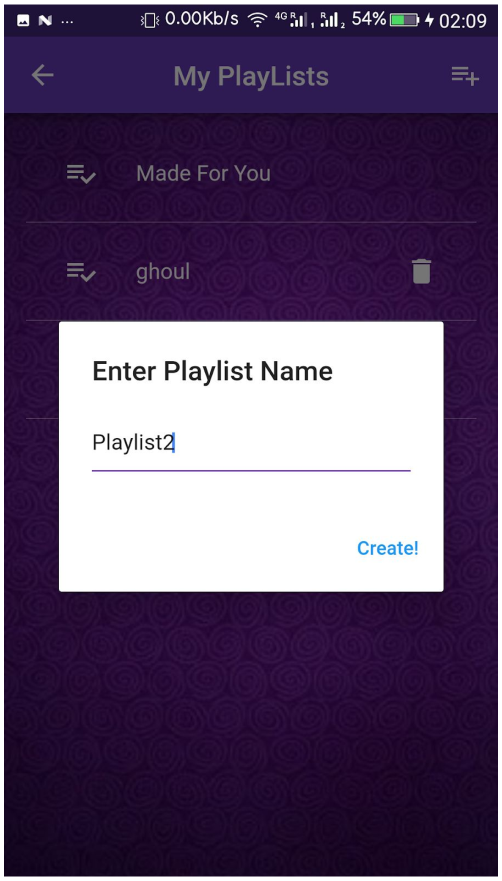


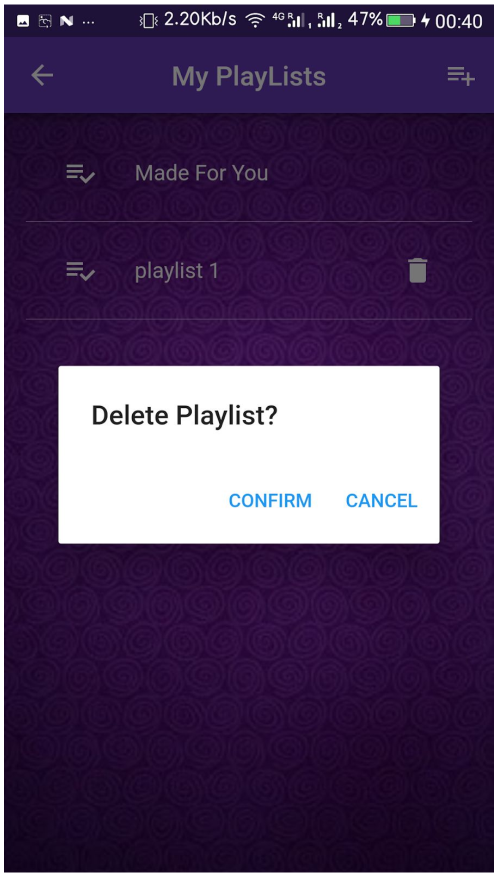


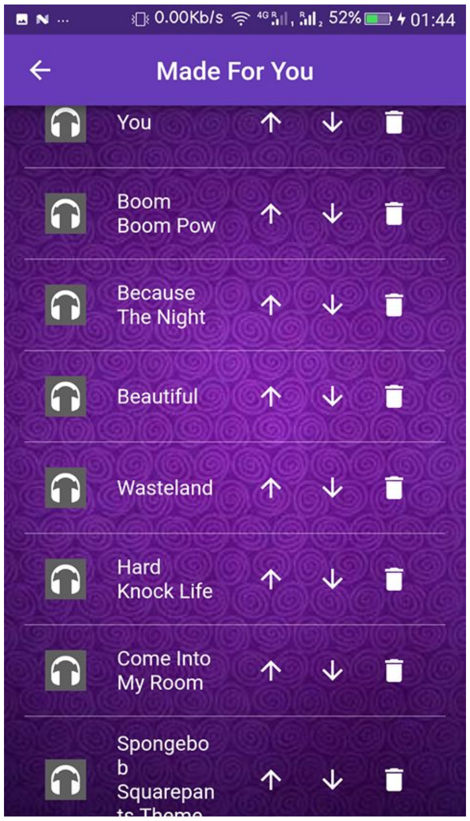
- **<u>My Playlists</u>**
    - These are the playlists that are created by the user and are permanent (saved across login sessions). These can be directly created by the user or come up as a result of making a queue permanent. Songs in the playlist can be reordered, can be deleted.
    - Among these playlists, is also a *Made For You* playlist which contains the songs suggested for the user based on his viewing and liking/disliking history. The details of the algorithm are explained in a later section. This playlist can't be deleted by the user but songs in it can be edited if he/she doesn't like them.

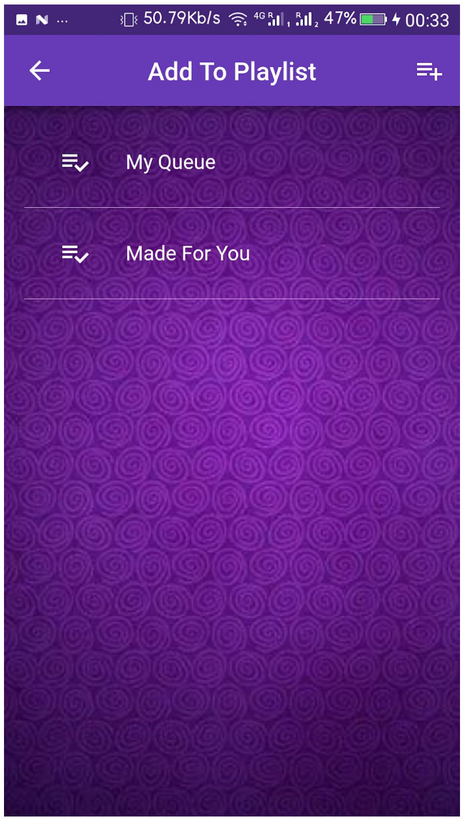




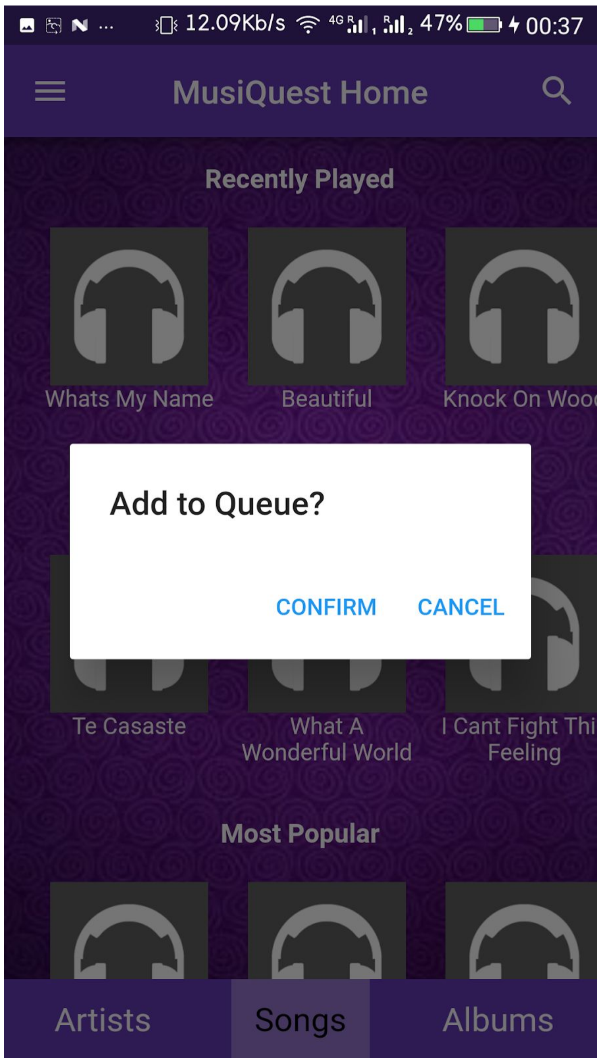

0.00Kb/s 54% 02:09
My PlayLists
Made For You
ghoul
Enter Playlist Name
Playlist2
Create!

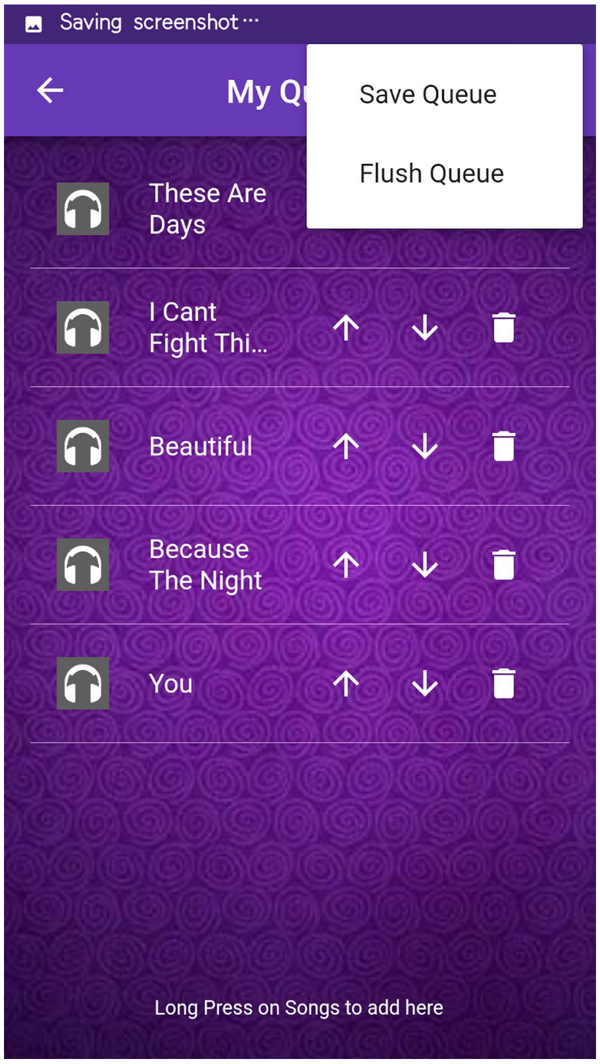

2.20Kb/s 47% 00:40
My PlayLists
Made For You
playlist 1
Delete Playlist?
CONFIRM
CANCEL


0.00Kb/s 52% 01:44
Made For You
You
Boom Boom Pow
Because The Night
Beautiful
Wasteland
Hard Knock Life
Come Into My Room
Spongebob Squarepants Theme


50.79Kb/s 47% 00:33
Add To Playlist
My Queue
Made For You

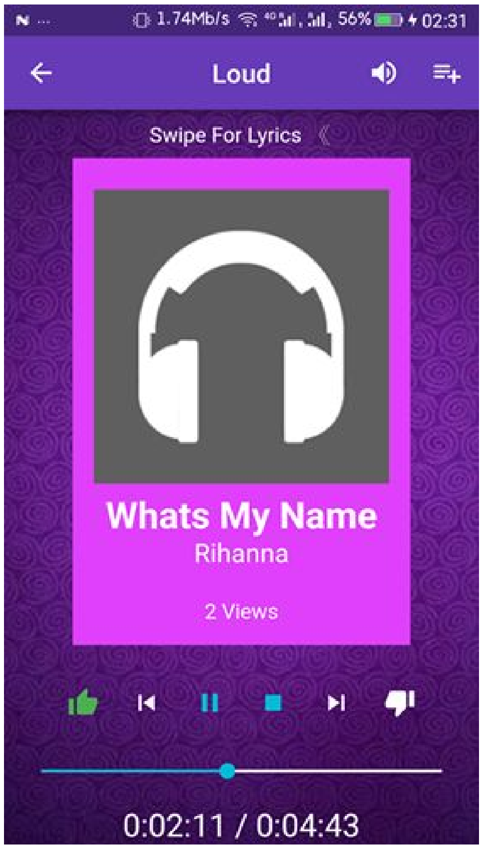
- **My Queue**
  - *My Queue* allows the user to maintain a temporary queue which he can listen to and make permanent if he wants to. This queue is temporary and is flushed on logout.
  - To add a song to this queue, a user just needs to long press a song and then click on add to the queue. Also from the song screen, he can choose to add to any playlist or the user queue.
  - The temporary queue can be saved as a permanent playlist or can be flushed. The songs in the queue can be reordered.

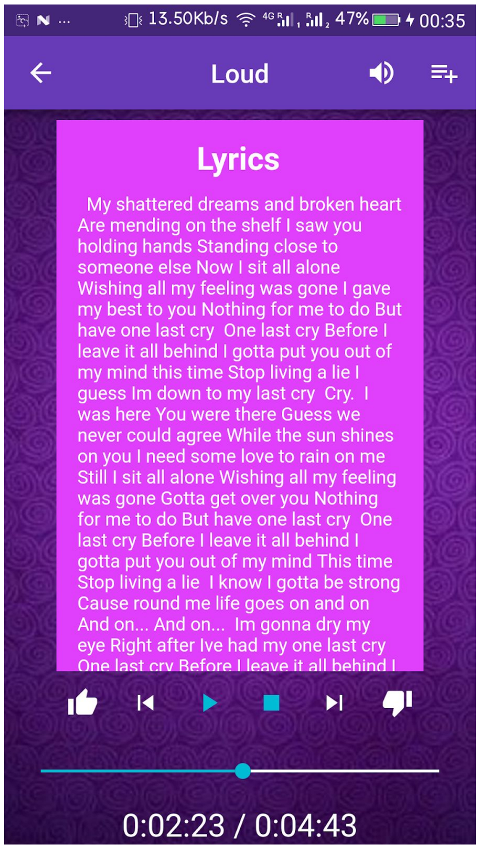




- **Song Screen**
    - There are two parts of the Song Screen. The Song Body shows the image for the song, the song's name, its artist and its number of views. Performing a Left Swipe on the Body shows the corresponding lyrics of the song, if available. A further Right Swipe restores the screen.
    - The second part, the Media Player actually plays the song. It has the Play-Pause-Stop functionality and the Seek functionality. By seeking a position on the Slider, the song will start playing from the corresponding time. There is added functionality of moving to the next or previous song in the queue/ playlist/ album. When a song ends, it automatically starts the next song (*autoplay*).
    - For user feedback on the song, we have the like/dislike functionality. A song can be liked if 'thumbs-up' is pressed, disliked with a 'thumbs-down' or kept neutral with no feedback. Every time a new song is played by a specific user, the total views of the songs increases by 1. Every time a song is viewed, the number of views specifically corresponding to the user for the song (a separate variable) increases (if a user views a song many times, it would feature high on his recommended song list). This feedback will help tune the Song Recommendation feature for the user. One user can like/ dislike a song only once.
    - The App bar for this page has the option to mute the song, and an option to add the current song to a Playlist or the user queue. The user will be directed to a new page to enable him/her to choose which Playlist to add to.

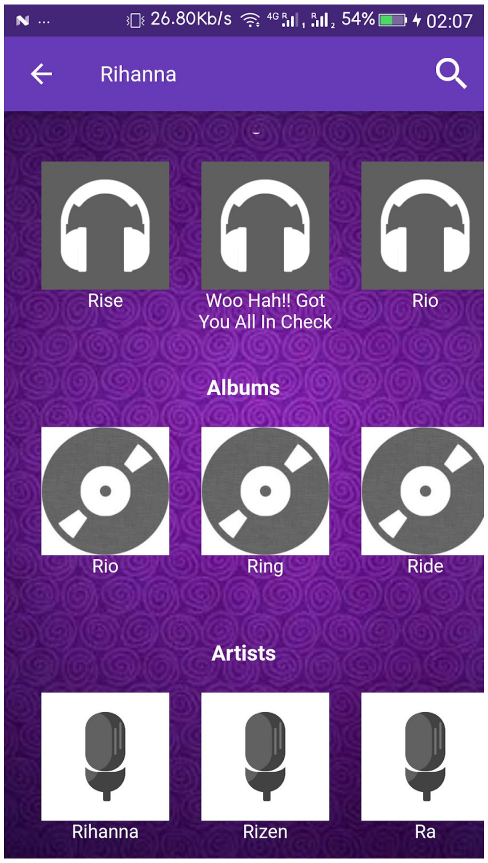
- **Search Screen**
    - This page can be accessed by pressing the search icon on the Home Screen. Entering a String into the Search Text Field, the app will show the user the top 8 songs, albums, artists whose names most match the String.
    - Additionally, Search result displays some songs whose lyrics contain strings similar to the searched string.



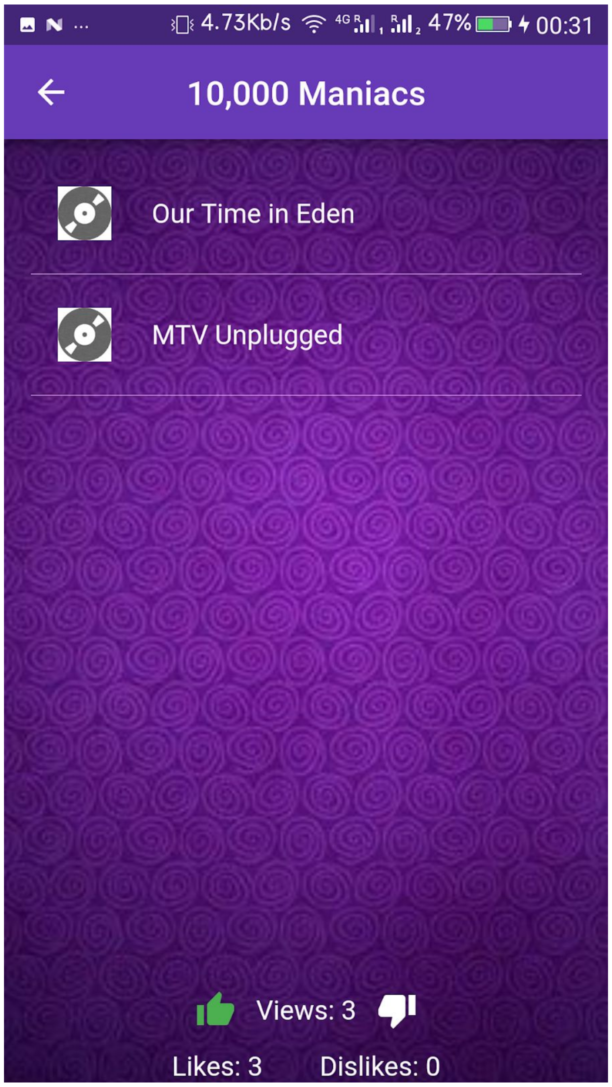


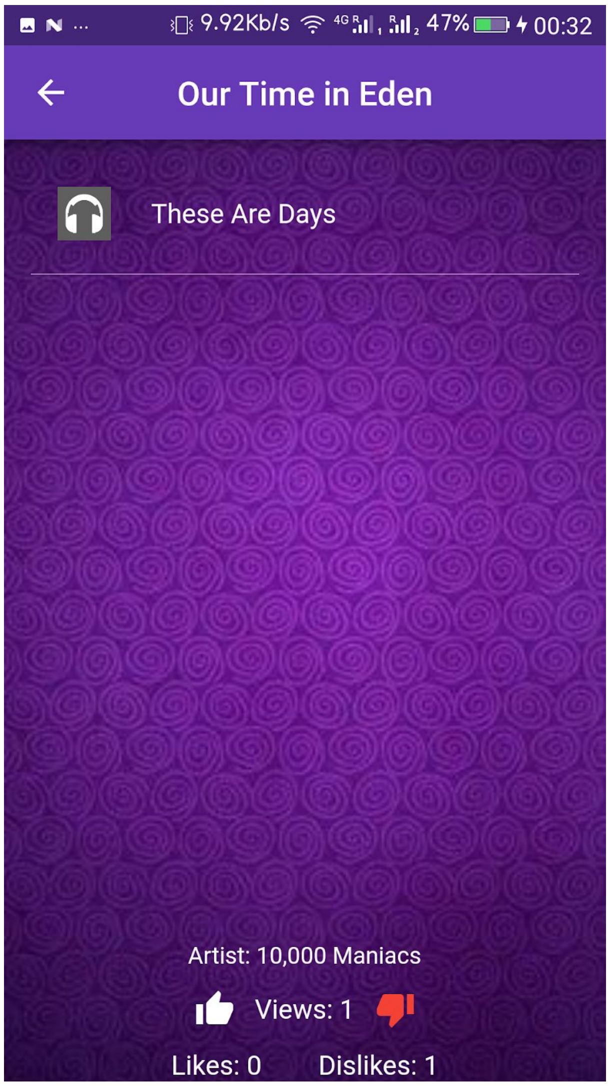
Lyrics

My shattered dreams and broken heart Are mending on the shelf I saw you holding hands Standing close to someone else Now I sit all alone Wishing all my feeling was gone I gave my best to you Nothing for me to do But have one last cry One last cry Before I leave it all behind I gotta put you out of my mind this time Stop living a lie I guess Im down to my last cry Cry. I was here You were there Guess we never could agree While the sun shines on you I need some love to rain on me Still I sit all alone Wishing all my feeling was gone Gotta get over you Nothing for me to do But have one last cry One last cry Before I leave it all behind I gotta put you out of my mind This time Stop living a lie I know I gotta be strong Cause round me life goes on and on And on... And on... Im gonna dry my eye Right after Ive had my one last cry One last cry Before I leave it all behind I



- **Artist Page**
    - This page displays all Albums available for the particular artist. There's an added functionality for the user to give feedback to the artist by like/ dislike icons.
- **Album Page**
    - This page displays all Songs available in the particular album. There's an added functionality for the user to give feedback to the album by like/ dislike icons.





RECOMMENDATION ALGORITHM

The aim of the recommendation algorithm used is to find the users which have a similar taste in music and to suggest the songs which are most viewed/liked by those users. Following are the details of the algorithm that we used.

1. First, for every song that a user has viewed we give a score based on whether he has liked or disliked it. This score is:
    a. Equal to the number of times he has viewed *(n)* it if he has neither liked nor disliked the song
    b. (n-10)*0.5 if the user has disliked the song.
    c. (n+10)*1.5 if the user has liked the song.
2. Next, we calculate an index for a pair of users indicating the similarity between the two. To calculate this index, we sum the squares of the difference of the scores given by the two users for every song common between the two users. Next we divide this sum by $N^{3/2}$ where N is the number of songs common between the two. So far we a distance metric between the users. To convert it into a similarity metric we add both the numerator and denominator by a constant and invert it.
3. Now, we choose top k users, and for every song in the set of songs we calculate a joint score for each song as follows. For each song, add for every user that listens to this song the similarity metric of the user multiplied by the score of the song as given by that user. Finally we recommend the top k songs as given by having the highest value of this sum.

SCRIPTS USED TO GENERATE DATASET

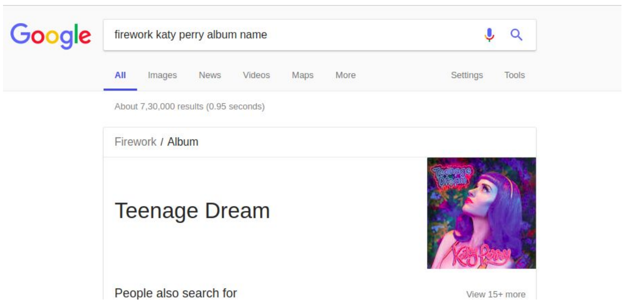
We used the Playlist Dataset [1] from Cornell which had a list of songs and their artists. For the purpose of our app we also needed their albums, release years and lyrics. We created Web Crawlers using the BeautifulSoup library in Python to achieve this.

ALBUM NAME

As we can see, given the artist and the song name, the album name always shows up on a fixed place in a Google search result. This property can be exploited using crawlers to fetch the contents of 'div' of a fixed class.

RELEASE DATE

Similar to the album name, the release date for a given album and artist always shows up at a fixed place.

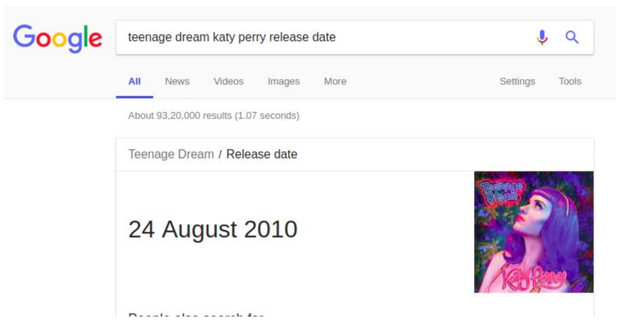

firework katy perry album name

About 7,30,000 results (0.95 seconds)

Firework / Album

Teenage Dream

People also search for


teenage dream katy perry release date

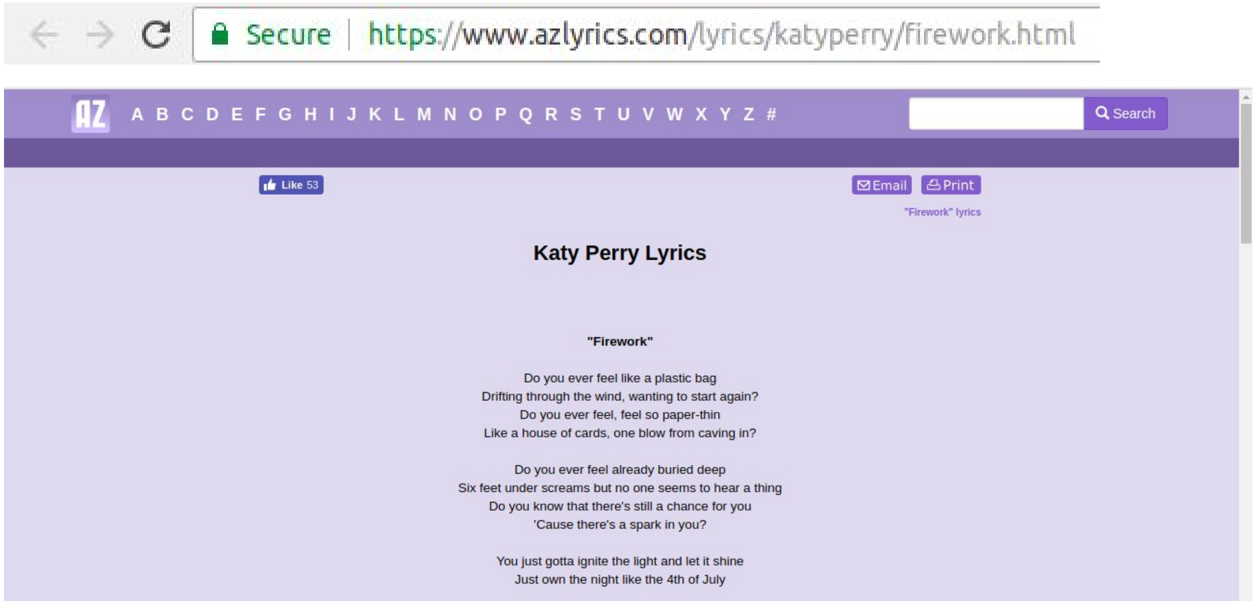
About 93,20,000 results (1.07 seconds)

Teenage Dream / Release date

24 August 2010

LYRICS

We used the website “azlyrics” to fetch the lyrics as it had a simple URL structure as well as a simple webpage to search for lyrics. As seen in the above screenshot, the URL for a song given song name and artist name can be generated in a very simple way. Also, the lyrics reside on a fixed place for every song, a property which can again be exploited easily using web-crawlers.


Secure | https://www.azlyrics.com/lyrics/katyperry/firework.html

Katy Perry Lyrics

"Firework"

Do you ever feel like a plastic bag
Drifting through the wind, wanting to start again?
Do you ever feel, feel so paper-thin
Like a house of cards, one blow from caving in?

Do you ever feel already buried deep
Six feet under screams but no one seems to hear a thing
Do you know that there's still a chance for you
'Cause there's a spark in you?

You just gotta ignite the light and let it shine
Just own the night like the 4th of July

FUNCTIONALITIES THAT CAN BE ADDED

Apart from the above-mentioned features that we plan to surely implement, there are certain features that we will look to implement if we have some time on our side. Following is the account of such points:

➔ **Admin Mode:** Privileged users can add and remove songs, albums, and artists.

➔ **Voice Search:** There are some libraries that can be used to match audio and return results. We might look to work with these libraries so that a user can give a recording of a small part of a song and we can return the matching results.

➔ **Privacy of user playlists:** There could be an option to the users to make their playlist public, that is, viewable but unmalleable to others. Also, we can store 'priority' for each song to allow the user to reorder songs in the playlist when playing in cyclic order (as compared to the usual case where we sort by 'timestamp' when the song was added to the playlist).

➔ **Discussion Forum / Reviews:** This will enable users to add comments and reply to comments on Songs, Albums and Artists and review the song.

FUTURE SCOPE

There are several features that can be incorporated into our App, but we have decided to exclude them either because of the time crunch or because of lack of available data. Listed below is an account of such features along with the way to add them if required.

➔ **Information Window:** Most modern Music websites and Apps provide information and other trivia about various Songs, Artists and Albums, however, it would be nearly impossible for a small team to gather and maintain such a database. However, logistically speaking, provided such data, we can easily incorporate it into our Relational Schema.

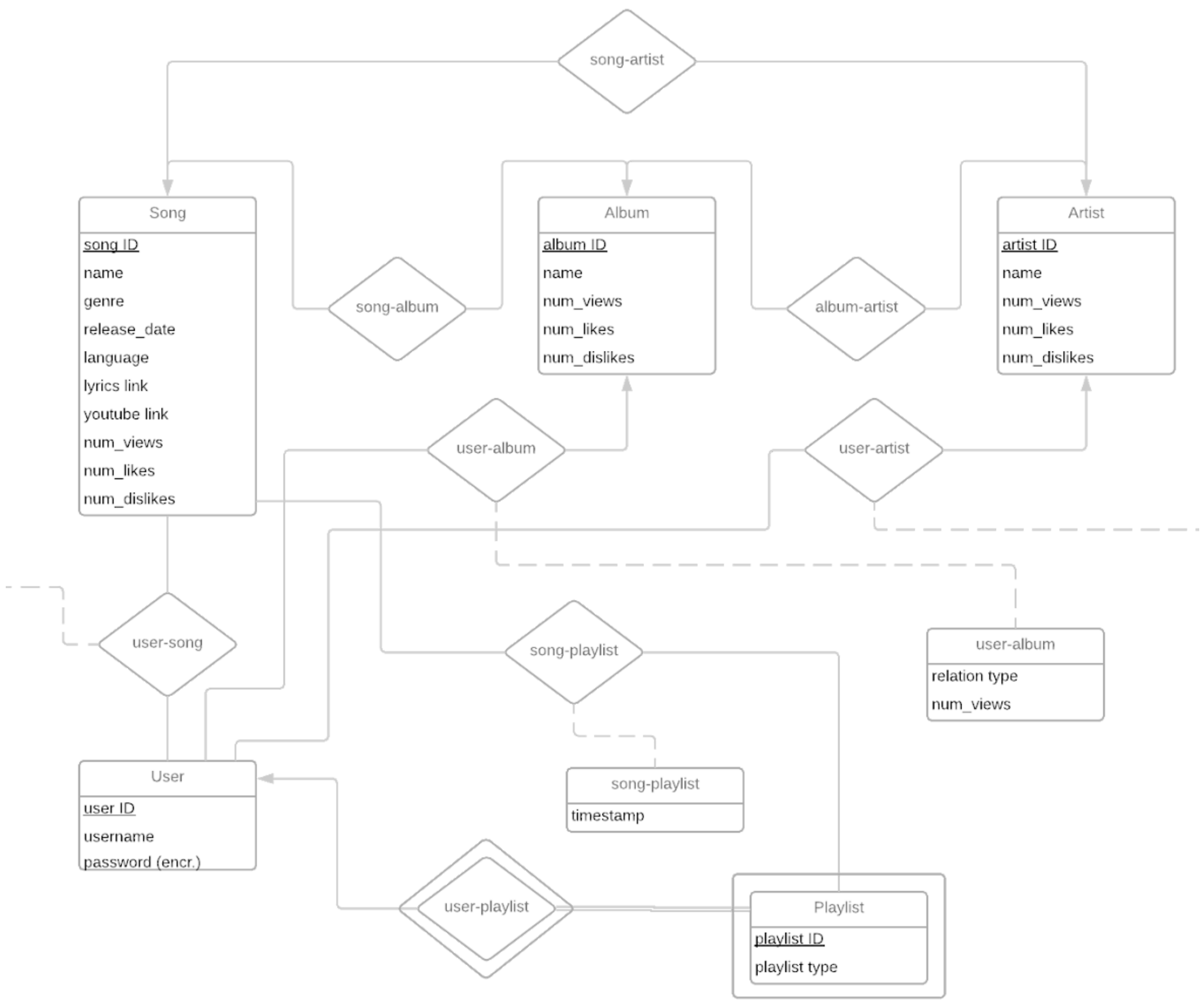
REFERENCES

1. https://www.cs.cornell.edu/~shuochen/lme/data_page.html

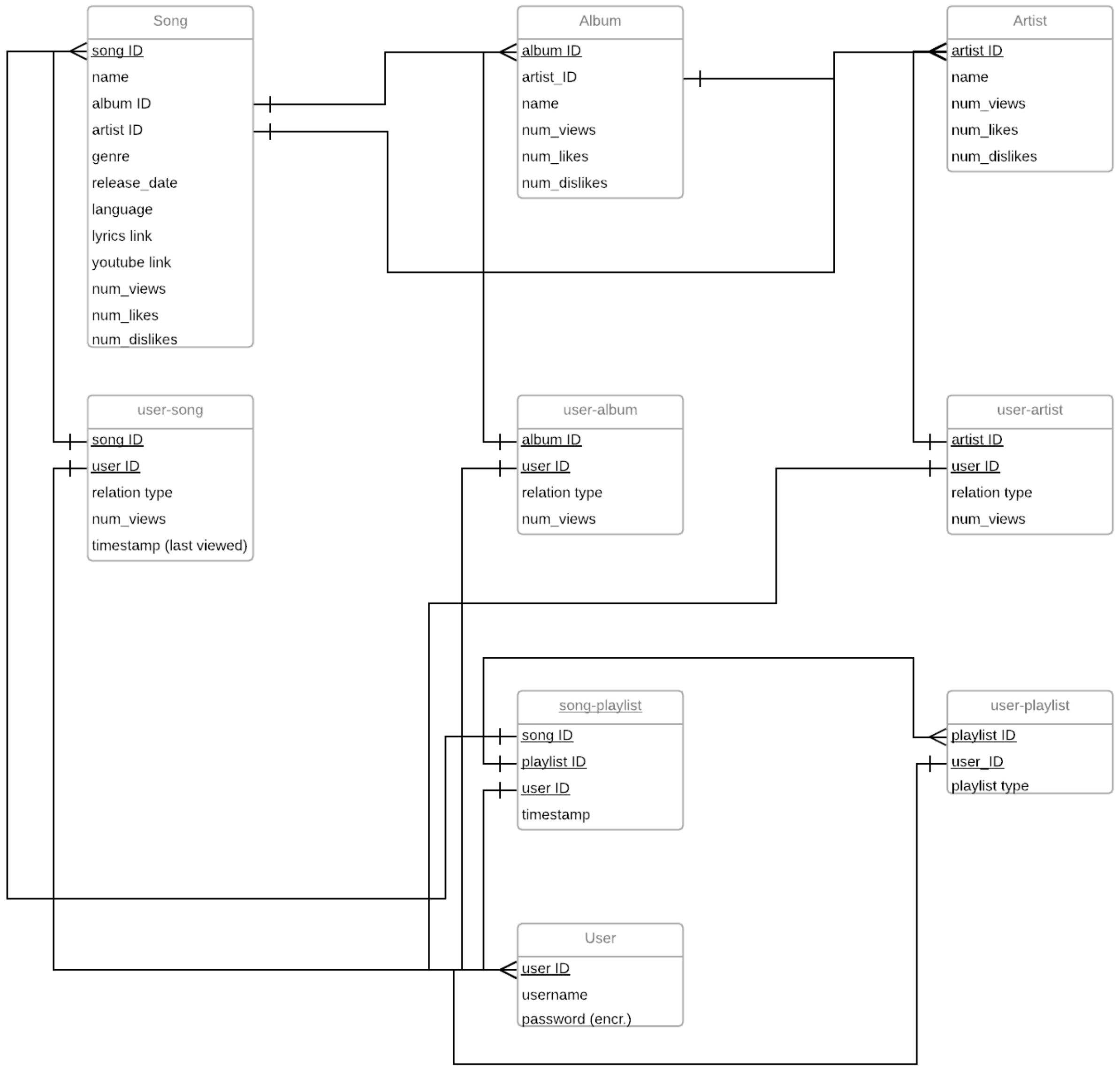
2. https://stackoverflow.com/questions/6120657/how-to-generate-a-unique-hash-code-for-string-input-in-android

3. https://pub.dartlang.org/packages/audioplayers

ER DIAGRAM


song-artist
Song
song ID
name
genre
release_date
language
lyrics link
youtube link
num_views
num_likes
num_dislikes
song-album
Album
album ID
name
num_views
num_likes
num_dislikes
album-artist
Artist
artist ID
name
num_views
num_likes
num_dislikes
user-album
user-artist
user-song
song-playlist
user-album
relation type
num_views
song-playlist
timestamp
User
user ID
username
password (encr.)
user-playlist
Playlist
playlist ID
playlist type

SCHEMA (TABLE DESIGN)


Song
song ID
name
album ID
artist ID
genre
release_date
language
lyrics link
youtube link
num_views
num_likes
num_dislikes
Album
album ID
artist_ID
name
num_views
num_likes
num_dislikes
Artist
artist ID
name
num_views
num_likes
num_dislikes
user-song
song ID
user ID
relation type
num_views
timestamp (last viewed)
user-album
album ID
user ID
relation type
num_views
user-artist
artist ID
user ID
relation type
num_views
song-playlist
song ID
playlist ID
user ID
timestamp
user-playlist
playlist ID
user_ID
playlist type
User
user ID
username
password (encr.)