2020

Music Sentiment Analysis

NAME / STUDENT NO.

Yanmei Zeng / 10307389

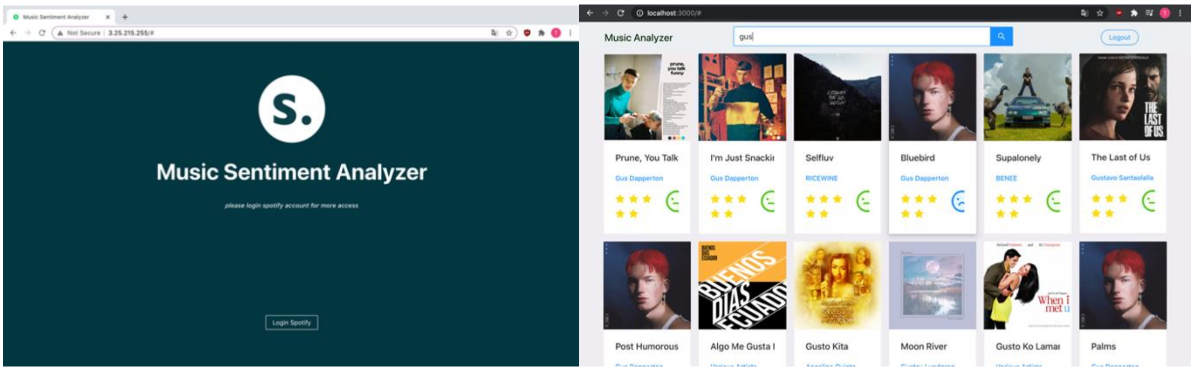
Yonathan Cahyadi / 10149953

Contents

Introduction

Purpose & description

Music sentiment analysis app allows user to search a song by music name, artist name or album name, then the app will display the lyrics of the song and using a AI sentiment analysis library make the sentiment tag after user login the Spotify account, it helping users to get to more about the song, and choses the music based on the mood (positive, negative and neutral).



Services used

Spotify API

Get user’s Authorization and recommend music

- Authorization Endpoint: https://accounts.spotify.com/authorize
- Recommended Music Endpoint: https://api.spotify.com/v1/recommendations
- Search Endpoint: https://api.spotify.com/v1/search
- Authorization Docs: https://developer.spotify.com/documentation/general/guides/authorization-guide/%23authorization-flows
- Recommended Music Docs: https://developer.spotify.com/documentation/web-api/reference/browse/get-recommendations/
- Search Docs: https://developer.spotify.com/documentation/web-api/reference/search/search/

Lyric API

Get song lyric

- Docs: https://lyricsovh.docs.apiary.io/

Natural Node package

"Natural" is a general natural language facility for node.js. This app using it to compute each worlds of a song's lyrics and calculate, the end can get the sentiment result: negative, positive and neutral.

- Docs: https://www.npmjs.com/package/natural/v/0.5.6#pos-tagger

REACT Node package

For building up the client side and give user a better view of the application.

- Docs: https://reactjs.org/docs/getting-started.html

AWS S3 bucket

Store data on the cloud, for long-term storage

- Docs: https://aws.amazon.com/s3/

REDIS Node package

Store data on the Redis for a short-term storage

- Docs: https://redis.io/documentation

Use cases

US 1 (See Appendix B, figure 2)

As a	Music Enthusiast
I want	Search a song by the name, album name or artist name
So that	I can find my favourite music

US 2 (See Appendix B, figure 3)

As a	Music Enthusiast
I want	Find a song's lyrics by searching this song's name
So that	Understand this song better and enjoying the music better

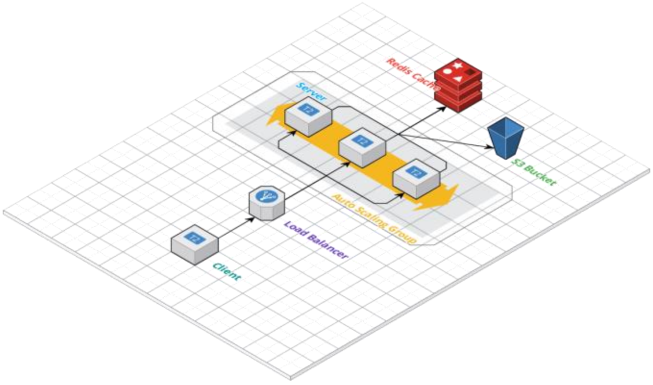
Technical breakdown

Architecture

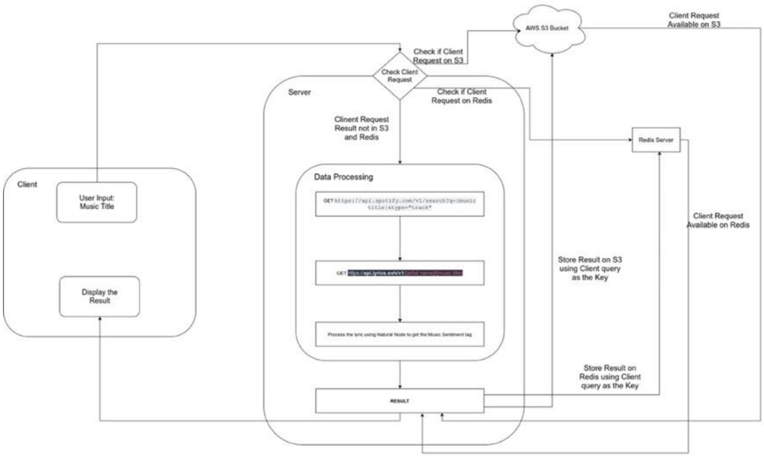
For the architecture for our project we choose a simple one. In this project we have our client connected to our server auto scaling group load balancer and our server is connected to both S3 Bucket for long term storage and ElastiCache Redis for short term storage, and our server will scale based on the CPU utilization, Network In and Network Out policy. While our Server scale up, our ElastiCache Redis and S3 Bucket will not scale together with the server, therefore every instance in the Auto Scaling group is connected to the same

ElastiCache Redis and S3 Bucket (Stateless). The reason for why we only scale the server is, because most of the data processing will be happened on the server side, therefore the server will require a lot of resources. As for the client side will only handle user authorization and getting user query, which not requiring a lot of resources.

Architecture Diagram



Process flow Diagram


AWS S3 Bucket
Client Request Available on S3
Check if Client Request on S3
Check Client Request
Server
Check if Client Request on Redis
Clinent Request Result not in S3 and Redis
Redis Server
Client
User Input: Music Title
Data Processing
GET https://api.spotify.com/v1/search?q={music title}&type="track"
GET https://api.lyrics.ovh/v1/
Process the lyric using Natural Node to get the Music Sentiment tag
Client Request Available on Redis
Store Result on S3 using Client query as the Key
Display the Result
Store Result on Redis using Client query as the Key
RESULT

Client / server demarcation of responsibilities

Server (see Appendix A, figure 1, 2, 3, 4, 5):

- The server is responsible for handling user query and processing the data received from the APIs.
- Store the processed data into the S3 Bucket and Redis cache. The data is stored based on the user query.

Client:

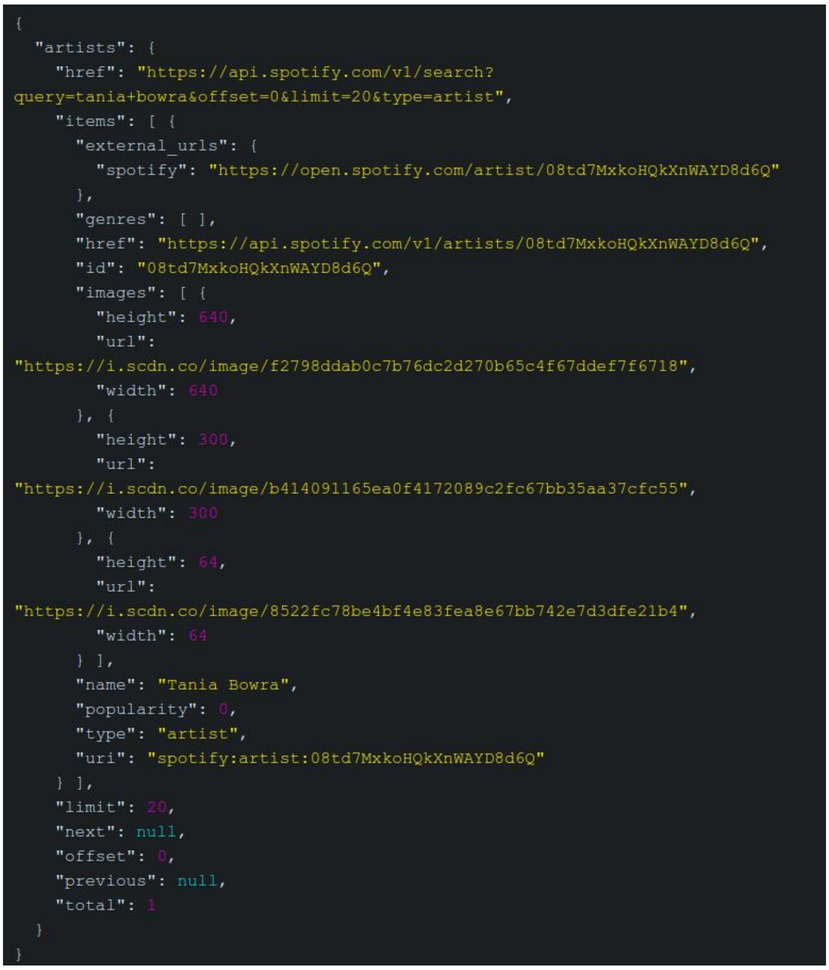
- The client is responsible for getting user authorization, getting user query and displaying the output of the server.

Response filtering / data object correlation

For this project the data source is coming from 2 APIs. The first API is from Spotify, by using this API we can get the details of our music query. Such details include the song title, artist, popularity, preview url, song images, available market. The second API is from lyric.ovh, we are using this API for getting the lyric of the music. Then using the lyric, we got from our second API we can do sentiments analysis using the Node Natural library. From the analysis we got number specifying the sentiment of the music. The sentiments number will be ranging from positive to negative number, as for the sentiment tag we can decide that based on the number we got, for example positive number will be tagged as "positive" and negative number will be tagged as "negative" as for "neutral" tag it will be a number between 0.05 to – 0.05. The lyric is also analyzed based on the word frequency.

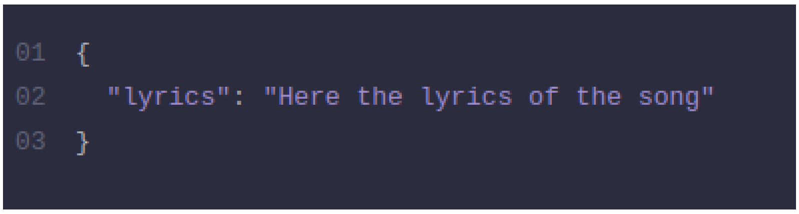
As for the response we are sending, it will consist of song title, artist, preview url, song images, sentiment tag, sentiment number, top 10 most used words.

Spotify Response:



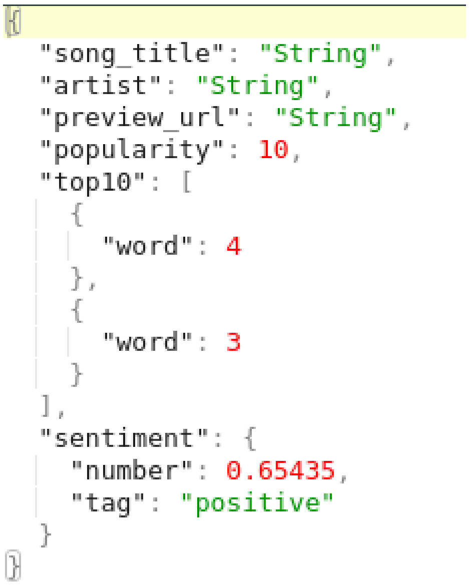
```
{
  "artists": {
    "href": "https://api.spotify.com/v1/search?
query=tania+bowra&offset=0&limit=20&type=artist",
    "items": [ {
      "external_urls": {
        "spotify": "https://open.spotify.com/artist/08td7MxkoHQkXnWAYD8d6Q"
      },
      "genres": [ ],
      "href": "https://api.spotify.com/v1/artists/08td7MxkoHQkXnWAYD8d6Q",
      "id": "08td7MxkoHQkXnWAYD8d6Q",
      "images": [ {
        "height": 640,
        "url":
"https://i.scdn.co/image/f2798ddab0c7b76dc2d270b65c4f67ddef7f6718",
        "width": 640
      }, {
        "height": 300,
        "url":
"https://i.scdn.co/image/b414091165ea0f4172089c2fc67bb35aa37cfc55",
        "width": 300
      }, {
        "height": 64,
        "url":
"https://i.scdn.co/image/8522fc78be4bf4e83fea8e67bb742e7d3dfe21b4",
        "width": 64
      } ],
      "name": "Tania Bowra",
      "popularity": 0,
      "type": "artist",
      "uri": "spotify:artist:08td7MxkoHQkXnWAYD8d6Q"
    } ],
    "limit": 20,
    "next": null,
    "offset": 0,
    "previous": null,
    "total": 1
  }
}
```

Lyric.ovh Response:



```
{
  "lyrics": "Here the lyrics of the song"
}
```

The Response sent to the Client:



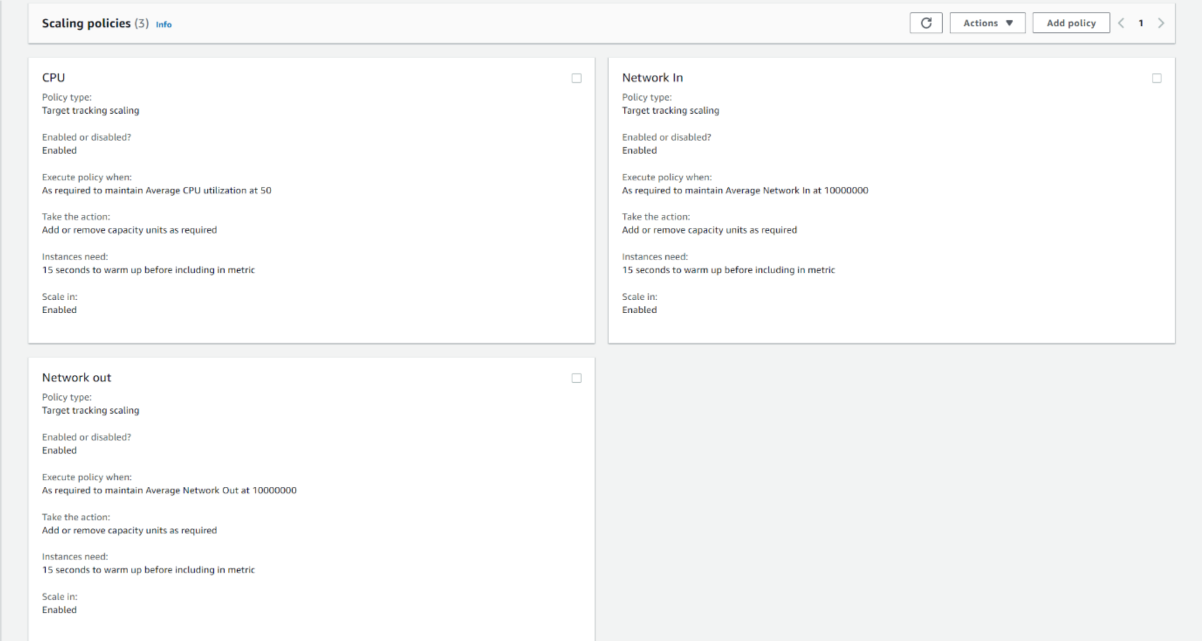
```
{
  "song_title": "String",
  "artist": "String",
  "preview_url": "String",
  "popularity": 10,
  "top10": [
    {
      "word": 4
    },
    {
      "word": 3
    }
  ],
  "sentiment": {
    "number": 0.65435,
    "tag": "positive"
  }
}
```

Scaling and Performance

For this project scaling policies, we decided that the policy is based on CPU utilization, Network In and Network Out. The reason why we choose this policy is:

- ***For CPU utilization policy**: the load will be coming from processing the song lyric. We decide the threshold to be 50% CPU utilization. The reason for this is because, when the user search for music sometimes the music doesn't have a lyric, therefore there is a lot less data to process. We also find that threshold of 50% is a good balance for our scaling policy.*
- ***For Network In policy**: the load will be coming from the APIs that we used. Our app using a lot of network bandwidths, for example for every user request the server is sending around 50 requests to our APIs. As for the threshold we decide on 10MB, because for every JSON response we get from the APIs had around 20KB worth of data totaling in around 1MB worth of data, since for every user request we are sending around 50 requests to our APIs. For visualization, we can see from the scaling performance graph below, it shows that at peak our server has around 30MB worth of data to receive and process.*
- ***For Network Out policy**: the load will be coming from our server response to the client. The data we are sending from the server to the client will not be as large as the data we receive from the APIs, the reason being the data is already processed and filtered. Therefore, the amount of data that we need to send to the client is significantly smaller. The data we sent to the client is arounds 50KB, and the threshold for our policy is around 10MB.*

Scaling policy



Scaling policies (3) Info

CPU

Policy type:
Target tracking scaling

Enabled or disabled?
Enabled

Execute policy when:
As required to maintain Average CPU utilization at 50

Take the action:
Add or remove capacity units as required

Instances need:
15 seconds to warm up before including in metric

Scale in:
Enabled

Network In

Policy type:
Target tracking scaling

Enabled or disabled?
Enabled

Execute policy when:
As required to maintain Average Network In at 10000000

Take the action:
Add or remove capacity units as required

Instances need:
15 seconds to warm up before including in metric

Scale in:
Enabled

Network out

Policy type:
Target tracking scaling

Enabled or disabled?
Enabled

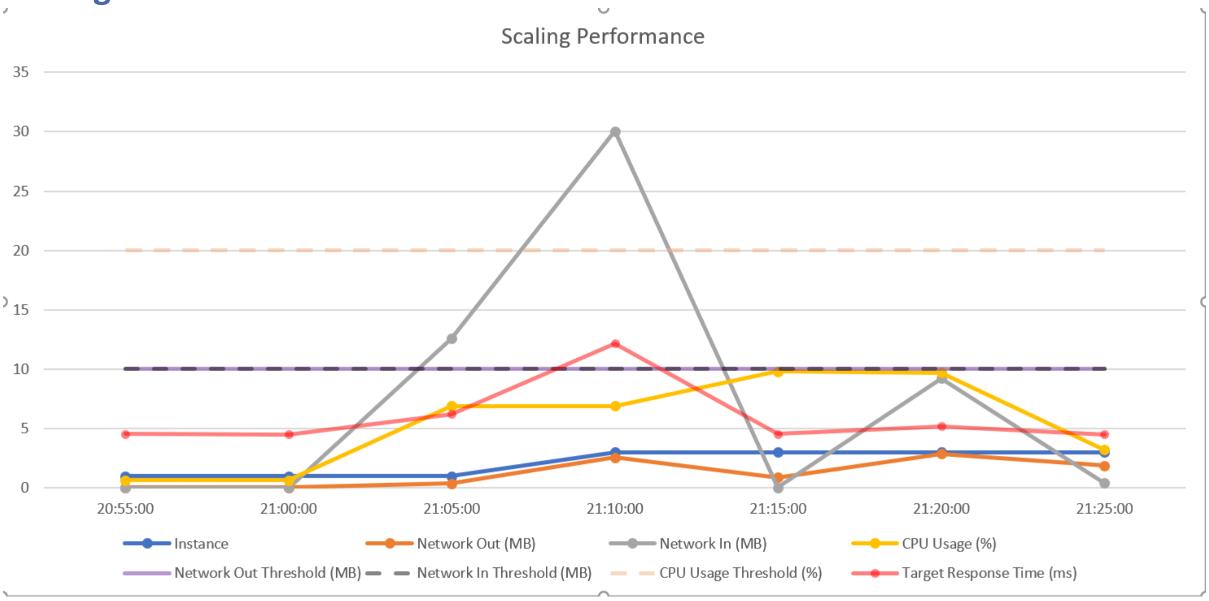
Execute policy when:
As required to maintain Average Network Out at 10000000

Take the action:
Add or remove capacity units as required

Instances need:
15 seconds to warm up before including in metric

Scale in:
Enabled

Scaling Performance



Test plan

Task	Expected outcomes	Result	Screenshots (Appendix B)
Login Spotify account	User can be authorized	PASS	1
Search music	Display music based on the search result	PASS	2
Play a preview version of the song	Music playing on the page	PASS	3
Link to the Spotify music playing page	Play the full vision of the song at Spotify	PASS	4

Display the Sentiment Tags of the music	Sentiment Tags displayed with sentiment value	PASS	3
Display the recommend music if no input is given	Recommend music list displayed	PASS	5
Logout button clicked	Go back to the login page	PASS	6
Handle top 10 words in a song's lyrics	Show a pie graph of 10 most used words	PASS	3
Handle no lyric available	Display info message	PASS	7
Handle no top 10 words	Display info message	PASS	7
Handle no preview available	Display info message and disable the buttons	PASS	7

Difficulties / Exclusions / unresolved & persistent errors /

Difficulties:

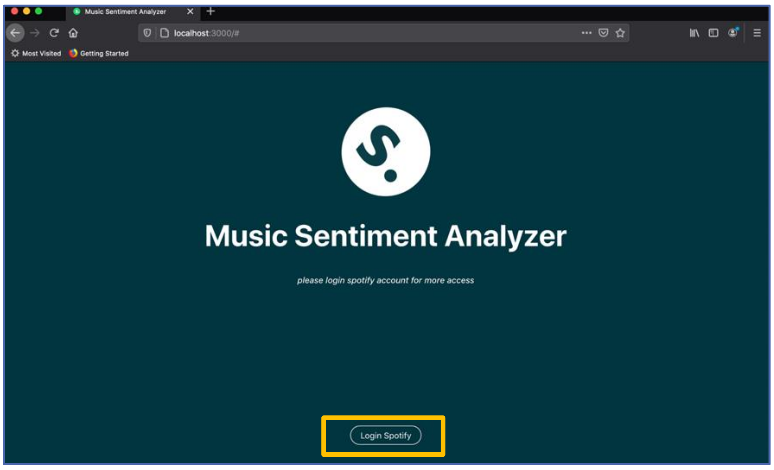
Deploying ElastiCache is kind of tricky, we need some time to figure out how to connect our server to the ElastiCache endpoint.

Extensions

For the future opportunities, we'd love to add a filter function and analysis Spotify users' daily and weekly recommend playlist, for the optimal user experience, user can filter the list by the sentiment tag "Positive", "Negative" and "Natural".

User guide

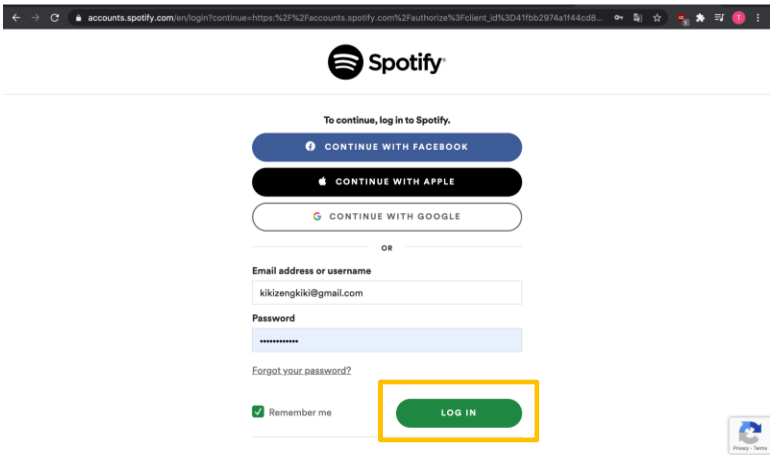
1. *Click Login to Spotify authorization page*



Music Sentiment Analyzer

please login spotify account for more access

2. *Login Spotify account*


accounts.spotify.com/en/login?continue=https:%2F%2Faccounts.spotify.com%2Fauthorize%3Fclient_id%3D41fbb2974a1f44cd8...
Spotify
CONTINUE WITH FACEBOOK
CONTINUE WITH APPLE
CONTINUE WITH GOOGLE
OR
Forgot your password?
LOG IN
Privacy - Terms

To continue, log in to Spotify.

Email address or username

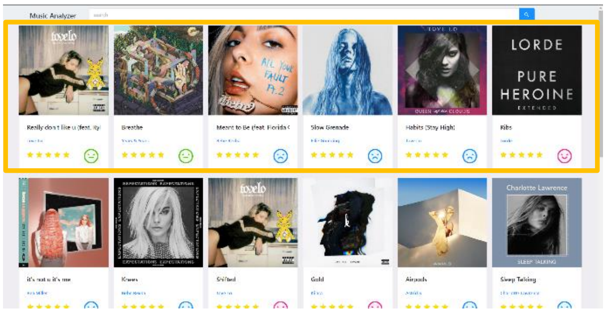
kikizengkiki@gmail.com

Password

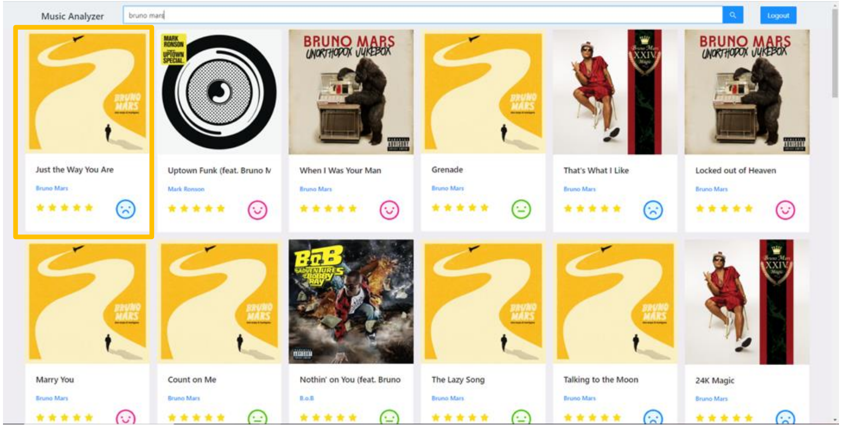
••••••••••••

☑ Remember me

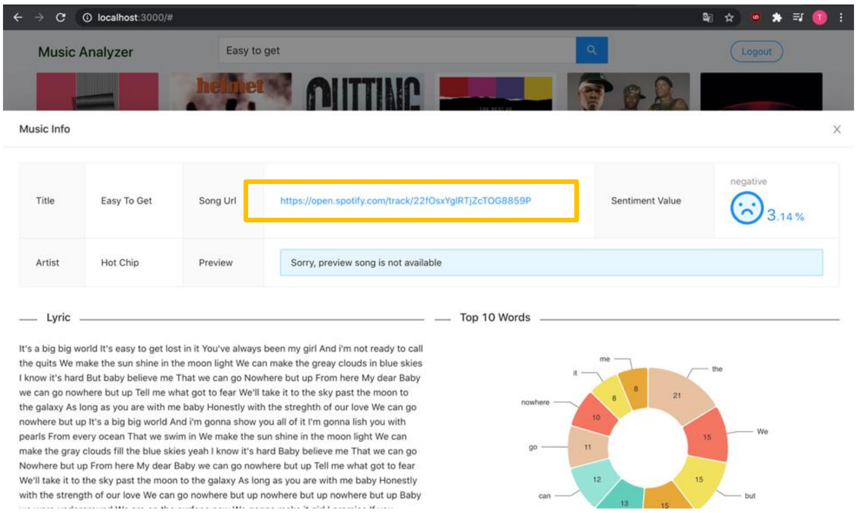
3. *After logged in will land on the home page with the recommend music list from Spotify with the popularity rates and sentiment tags.*



4. *User can search music name, album name or artist name, the result will be displayed, user can click the music for more information*



5. *User can check the lyrics, a summary of the top 10 words of this song, preview of the song if its available and a link to check full version of this song by clicking the url*



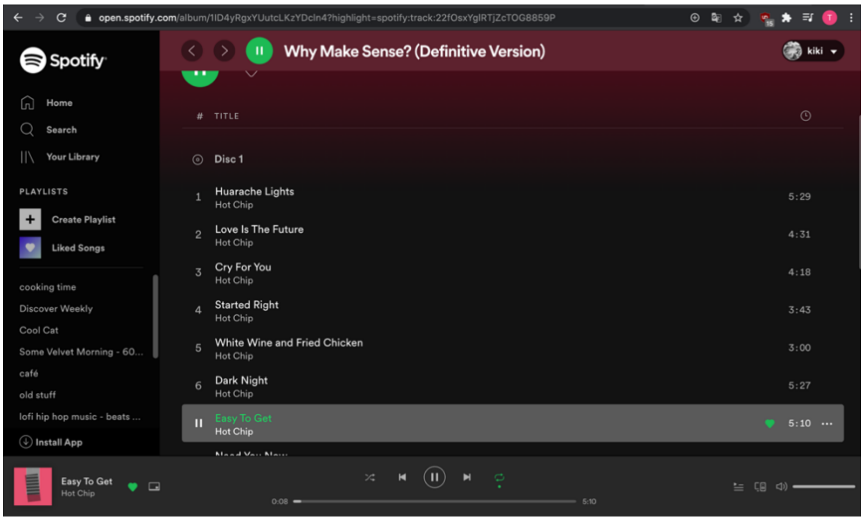
Title	Easy To Get	Song Url	https://open.spotify.com/track/22fOsxYglRTjZcTOG8859P	Sentiment Value	negative 3.14 %
Artist	Hot Chip	Preview	Sorry, preview song is not available		

Lyric

It's a big big world It's easy to get lost in it You've always been my girl And i'm not ready to call the quits We make the sun shine in the moon light We can make the greay clouds in blue skies I know it's hard But baby believe me That we can go Nowhere but up From here My dear Baby we can go nowhere but up Tell me what got to fear We'll take it to the sky past the moon to the galaxy As long as you are with me baby Honestly with the streghth of our love We can go nowhere but up It's a big big world And i'm gonna show you all of it I'm gonna lish you with pearls From every ocean That we swim in We make the sun shine in the moon light We can make the gray clouds fill the blue skies yeah I know it's hard Baby believe me That we can go Nowhere but up From here My dear Baby we can go nowhere but up Tell me what got to fear We'll take it to the sky past the moon to the galaxy As long as you are with me baby Honestly with the strength of our love We can go nowhere but up nowhere but up nowhere but up Baby

Top 10 Words

6. *Link to Spotify music playing page*

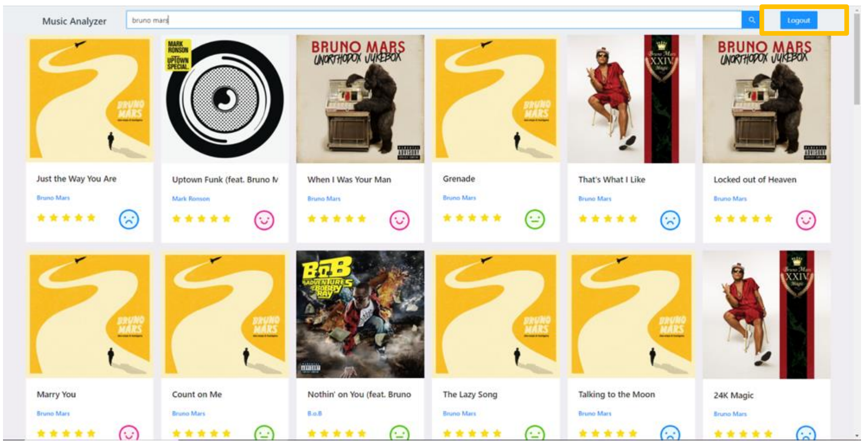

open.spotify.com/album/1ID4yRgxYUutcLKzYDcln4?highlight=spotify:track:22fOsxYglRTjZcTOG8859P
Spotify
Home
Search
Your Library
PLAYLISTS
Create Playlist
Liked Songs
cooking time
Discover Weekly
Cool Cat
Some Velvet Morning - 60...
café
old stuff
lofi hip hop music - beats ...
Install App
kiki

Why Make Sense? (Definitive Version)

#	TITLE	
	Disc 1	
1	Huarache Lights Hot Chip	5:29
2	Love Is The Future Hot Chip	4:31
3	Cry For You Hot Chip	4:18
4	Started Right Hot Chip	3:43
5	White Wine and Fried Chicken Hot Chip	3:00
6	Dark Night Hot Chip	5:27
	Easy To Get Hot Chip	5:10

Easy To Get
Hot Chip
0:08
5:10

7. *User can logout the Spotify account safely by clicking the Logout button, it will back to the App Login page*



Just the Way You Are
Bruno Mars

Uptown Funk (feat. Bruno M
Mark Ronson

When I Was Your Man
Bruno Mars

Grenade
Bruno Mars

That's What I Like
Bruno Mars

Locked out of Heaven
Bruno Mars

Marry You
Bruno Mars

Count on Me
Bruno Mars

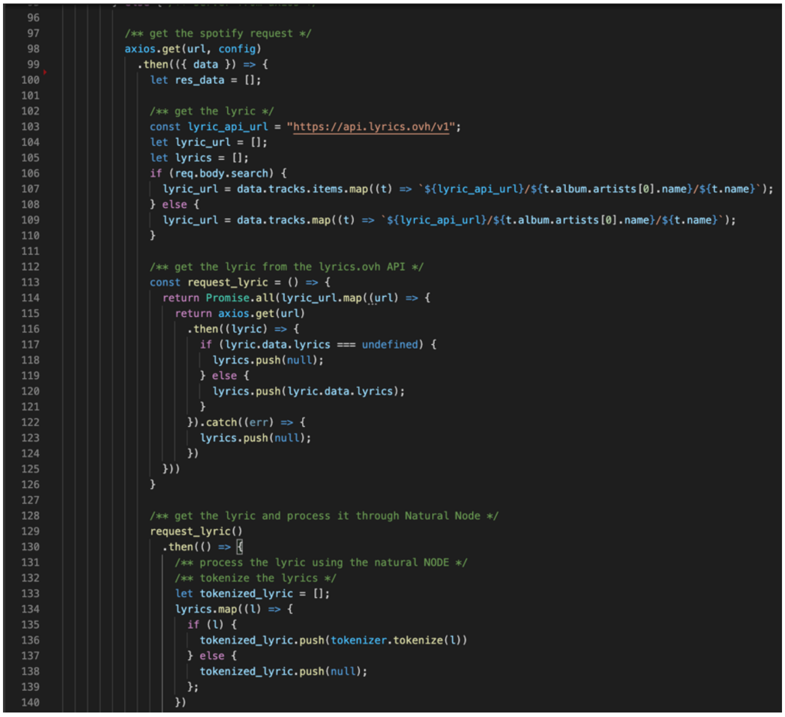
Nothin' on You (feat. Bruno
B.o.B

The Lazy Song
Bruno Mars

Talking to the Moon
Bruno Mars

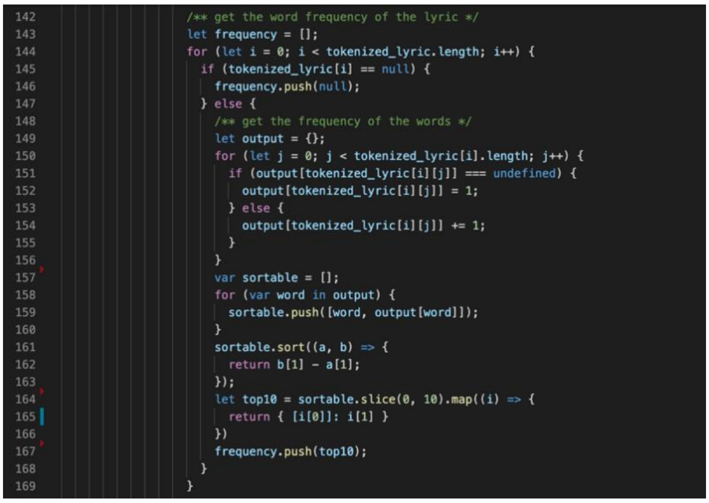
24K Magic
Bruno Mars

Appendix A



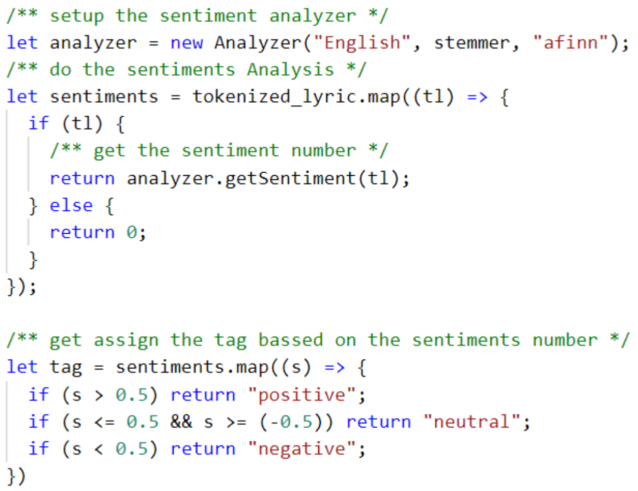
```
/** get the spotify request */
axios.get(url, config)
  .then(({ data }) => {
    let res_data = [];

    /** get the lyric */
    const lyric_api_url = "https://api.lyrics.ovh/v1";
    let lyric_url = [];
    let lyrics = [];
    if (req.body.search) {
      lyric_url = data.tracks.items.map((t) => `${lyric_api_url}/${t.album.artists[0].name}/${t.name}`);
    } else {
      lyric_url = data.tracks.map((t) => `${lyric_api_url}/${t.album.artists[0].name}/${t.name}`);
    }

    /** get the lyric from the lyrics.ovh API */
    const request_lyric = () => {
      return Promise.all(lyric_url.map((url) => {
        return axios.get(url)
          .then((lyric) => {
            if (lyric.data.lyrics === undefined) {
              lyrics.push(null);
            } else {
              lyrics.push(lyric.data.lyrics);
            }
          }).catch((err) => {
            lyrics.push(null);
          })
      }))
    }

    /** get the lyric and process it through Natural Node */
    request_lyric()
      .then(() => {
        /** process the lyric using the natural NODE */
        /** tokenize the lyrics */
        let tokenized_lyric = [];
        lyrics.map((l) => {
          if (l) {
            tokenized_lyric.push(tokenizer.tokenize(l))
          } else {
            tokenized_lyric.push(null);
          };
        })
```

Figure 1



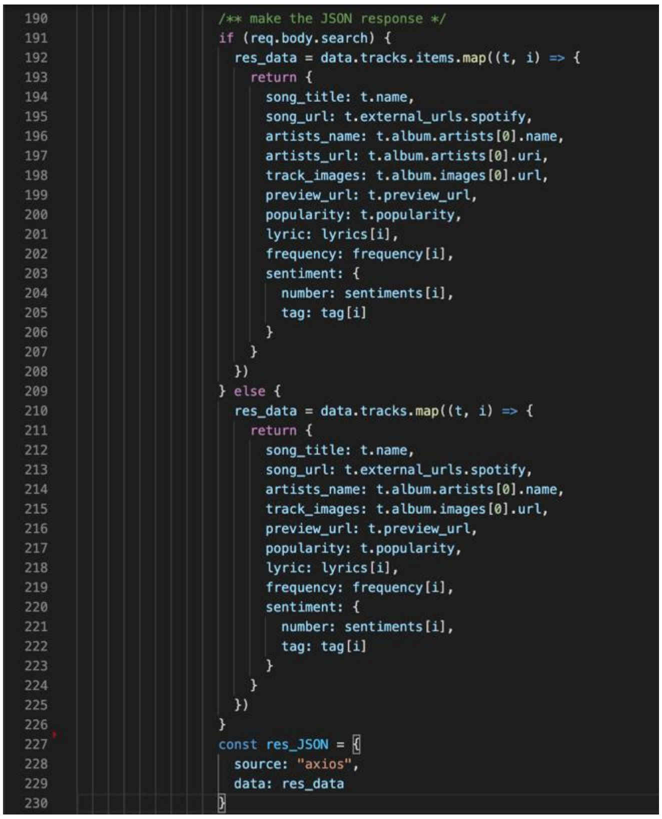
```
/** get the word frequency of the lyric */
let frequency = [];
for (let i = 0; i < tokenized_lyric.length; i++) {
  if (tokenized_lyric[i] == null) {
    frequency.push(null);
  } else {
    /** get the frequency of the words */
    let output = {};
    for (let j = 0; j < tokenized_lyric[i].length; j++) {
      if (output[tokenized_lyric[i][j]] === undefined) {
        output[tokenized_lyric[i][j]] = 1;
      } else {
        output[tokenized_lyric[i][j]] += 1;
      }
    }
    var sortable = [];
    for (var word in output) {
      sortable.push([word, output[word]]);
    }
    sortable.sort((a, b) => {
      return b[1] - a[1];
    });
    let top10 = sortable.slice(0, 10).map((i) => {
      return { [i[0]]: i[1] }
    })
    frequency.push(top10);
  }
}
```

Figure 2



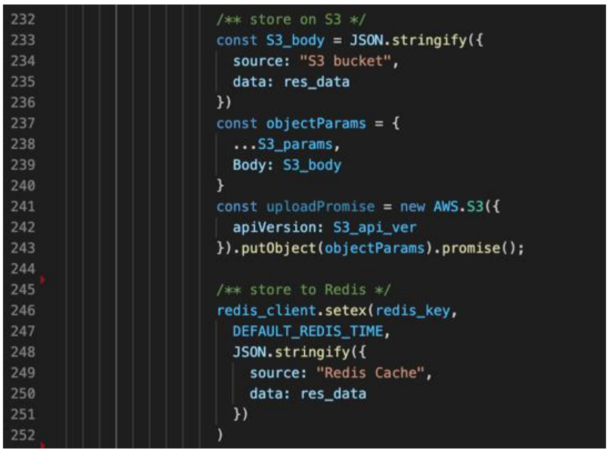
```
/** setup the sentiment analyzer */
let analyzer = new Analyzer("English", stemmer, "afinn");
/** do the sentiments Analysis */
let sentiments = tokenized_lyric.map((tl) => {
  if (tl) {
    /** get the sentiment number */
    return analyzer.getSentiment(tl);
  } else {
    return 0;
  }
});

/** get assign the tag bassed on the sentiments number */
let tag = sentiments.map((s) => {
  if (s > 0.5) return "positive";
  if (s <= 0.5 && s >= (-0.5)) return "neutral";
  if (s < 0.5) return "negative";
})
```

Figure 3



```
/** make the JSON response */
if (req.body.search) {
  res_data = data.tracks.items.map((t, i) => {
    return {
      song_title: t.name,
      song_url: t.external_urls.spotify,
      artists_name: t.album.artists[0].name,
      artists_url: t.album.artists[0].uri,
      track_images: t.album.images[0].url,
      preview_url: t.preview_url,
      popularity: t.popularity,
      lyric: lyrics[i],
      frequency: frequency[i],
      sentiment: {
        number: sentiments[i],
        tag: tag[i]
      }
    }
  })
} else {
  res_data = data.tracks.map((t, i) => {
    return {
      song_title: t.name,
      song_url: t.external_urls.spotify,
      artists_name: t.album.artists[0].name,
      track_images: t.album.images[0].url,
      preview_url: t.preview_url,
      popularity: t.popularity,
      lyric: lyrics[i],
      frequency: frequency[i],
      sentiment: {
        number: sentiments[i],
        tag: tag[i]
      }
    }
  })
}
const res_JSON = {
  source: "axios",
  data: res_data
}
```

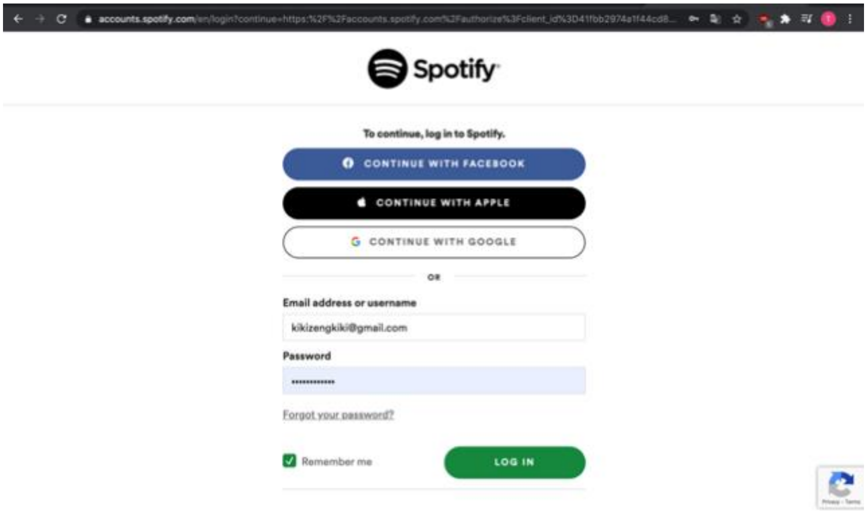
Figure 4



```
/** store on S3 */
const S3_body = JSON.stringify({
  source: "S3 bucket",
  data: res_data
})
const objectParams = {
  ...S3_params,
  Body: S3_body
}
const uploadPromise = new AWS.S3({
  apiVersion: S3_api_ver
}).putObject(objectParams).promise();

/** store to Redis */
redis_client.setex(redis_key,
  DEFAULT_REDIS_TIME,
  JSON.stringify({
    source: "Redis Cache",
    data: res_data
  })
)
```

Figure 5

Appendix B



accounts.spotify.com/en/login?continue=https:%2F%2Faccounts.spotify.com%2Fauthorize%3Fclient_id%3D41fbb2974a1f44cd8...
Spotify
CONTINUE WITH FACEBOOK
CONTINUE WITH APPLE
CONTINUE WITH GOOGLE
OR
Forgot your password?
LOG IN

To continue, log in to Spotify.

Email address or username

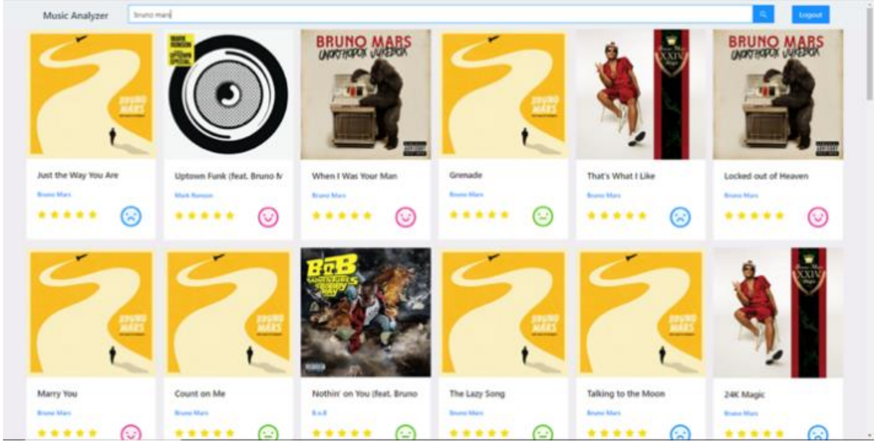
kikizengkiki@gmail.com

Password

••••••••••••

☑ Remember me

Figure 1



bruno mars

Just the Way You Are
Bruno Mars
★★★★★

Uptown Funk (feat. Bruno M
Mark Ronson
★★★★★

When I Was Your Man
Bruno Mars
★★★★★

Grenade
Bruno Mars
★★★★★

That's What I Like
Bruno Mars
★★★★★

Locked out of Heaven
Bruno Mars
★★★★★

Marry You
Bruno Mars
★★★★★

Count on Me
Bruno Mars
★★★★★

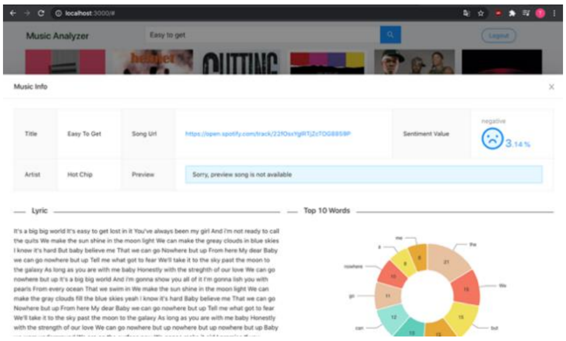
Nothin' on You (feat. Bruno
B.o.B
★★★★★

The Lazy Song
Bruno Mars
★★★★★

Talking to the Moon
Bruno Mars
★★★★★

24K Magic
Bruno Mars
★★★★★

Figure 2



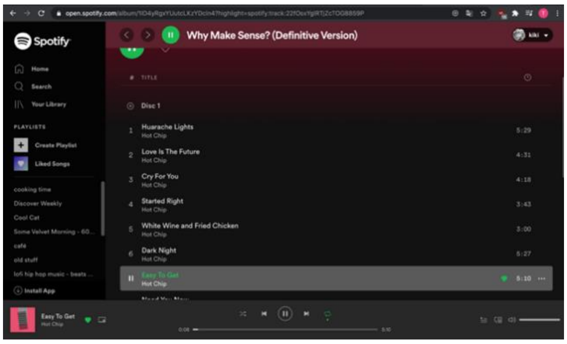
Title	Easy To Get	Song Url	https://open.spotify.com/track/22fOsxYglRTjZcTOG8859P	Sentiment Value	negative 3.14 %
Artist	Hot Chip	Preview	Sorry, preview song is not available		

Lyric

It's a big big world It's easy to get lost in it You've always been my girl And i'm not ready to call the quits We make the sun shine in the moon light We can make the greay clouds in blue skies I know it's hard But baby believe me That we can go Nowhere but up From here My dear Baby we can go nowhere but up Tell me what got to fear We'll take it to the sky past the moon to the galaxy As long as you are with me baby Honestly with the streghth of our love We can go nowhere but up It's a big big world And i'm gonna show you all of it I'm gonna lish you with pearls From every ocean That we swim in We make the sun shine in the moon light We can make the gray clouds fill the blue skies yeah I know it's hard Baby believe me That we can go Nowhere but up From here My dear Baby we can go nowhere but up Tell me what got to fear We'll take it to the sky past the moon to the galaxy As long as you are with me baby Honestly with the strength of our love We can go nowhere but up nowhere but up nowhere but up Baby

Top 10 Words

Figure 3



open.spotify.com/album/1O4yRgxYUutcLKzYDcln4?highlight=spotify:track:22fOsxYglRTjZcTOG8859P
Spotify
Home
Search
Your Library
PLAYLISTS
Create Playlist
Liked Songs
cooking time
Discover Weekly
Cool Cat
Some Velvet Morning - 60...
café
old stuff
lofi hip hop music - beats ...
Install App
kiki

Why Make Sense? (Definitive Version)

#	TITLE	
	Disc 1	
1	Huarache Lights Hot Chip	5:29
2	Love Is The Future Hot Chip	4:31
3	Cry For You Hot Chip	4:18
4	Started Right Hot Chip	3:43
5	White Wine and Fried Chicken Hot Chip	3:00
6	Dark Night Hot Chip	5:27
	Easy To Get Hot Chip	5:10

Figure 4

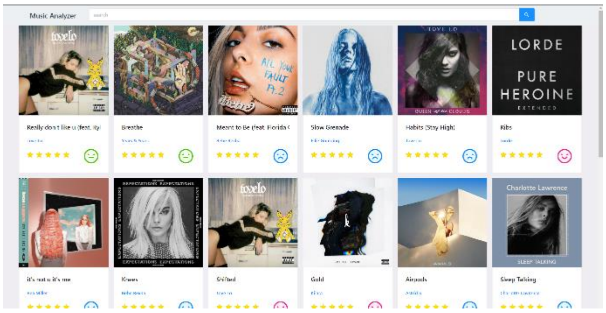
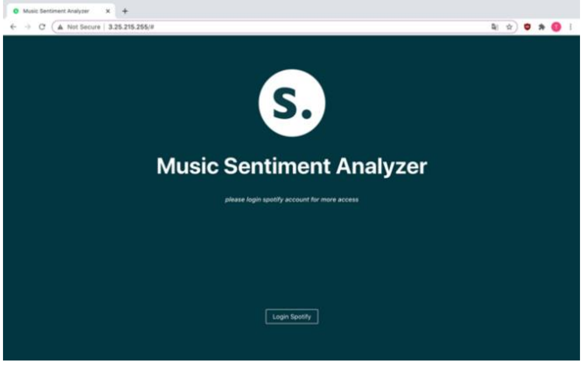


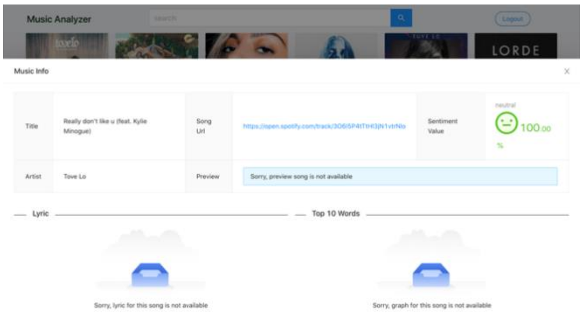
Figure 5



Music Sentiment Analyzer

please login spotify account for more access

Figure 6



Title	Really don't like u (feat. Kylie Minogue)	Song Url	https://open.spotify.com/track/3O6I5P4tTtHl3jN1vtrNlo	Sentiment Value	neutral 100.00 %
Artist	Tove Lo	Preview	Sorry, preview song is not available		

Sorry, lyric for this song is not available

Sorry, graph for this song is not available

Figure 7