2020

<Yanmei Zeng/s>

<Yonathan /s>

9/30/2020

Music Sentiment Analysis

A picture containing indoor, dark, sitting, table

Description automatically generated

**NAME / STUDENT NO.**

Yanmei Zeng / 10307389

Yonathan Cahyadi / 10149953

Contents

[Introduction 2](#_Toc54478793)

[Purpose & description 2](#_Toc54478794)

[Services used 2](#_Toc54478795)

[Spotify API 2](#_Toc54478796)

[Natural Node package 2](#_Toc54478797)

[REACT Node package 3](#_Toc54478798)

[AWS S3 bucket 3](#_Toc54478799)

[REDIS Node package 3](#_Toc54478800)

[Use cases 3](#_Toc54478801)

[US 1 (See Appendix B, figure 2) 3](#_Toc54478802)

[US 2 (See Appendix B, figure 3) 3](#_Toc54478803)

[Technical breakdown 3](#_Toc54478804)

[Architecture 3](#_Toc54478805)

[Architecture Diagram 4](#_Toc54478806)

[Process flow Diagram 4](#_Toc54478807)

[Client / server demarcation of responsibilities 4](#_Toc54478808)

[Response filtering / data object correlation 5](#_Toc54478809)

[Scaling and Performance 6](#_Toc54478810)

[Scaling policy 7](#_Toc54478811)

[Scaling Performance 7](#_Toc54478812)

[Test plan 7](#_Toc54478813)

[Difficulties / Exclusions / unresolved & persistent errors / 8](#_Toc54478814)

[Difficulties: 8](#_Toc54478815)

[Extensions 8](#_Toc54478816)

[User guide 8](#_Toc54478817)

[Appendix A 11](#_Toc54478818)

[Appendix B 13](#_Toc54478819)

# 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).

A screenshot of a computer

Description automatically generatedGraphical user interface, website

Description automatically generated

## 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/>

#### 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

Diagram, engineering drawing

Description automatically generated

#### Process flow Diagram

Diagram

Description automatically generated

#### 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:

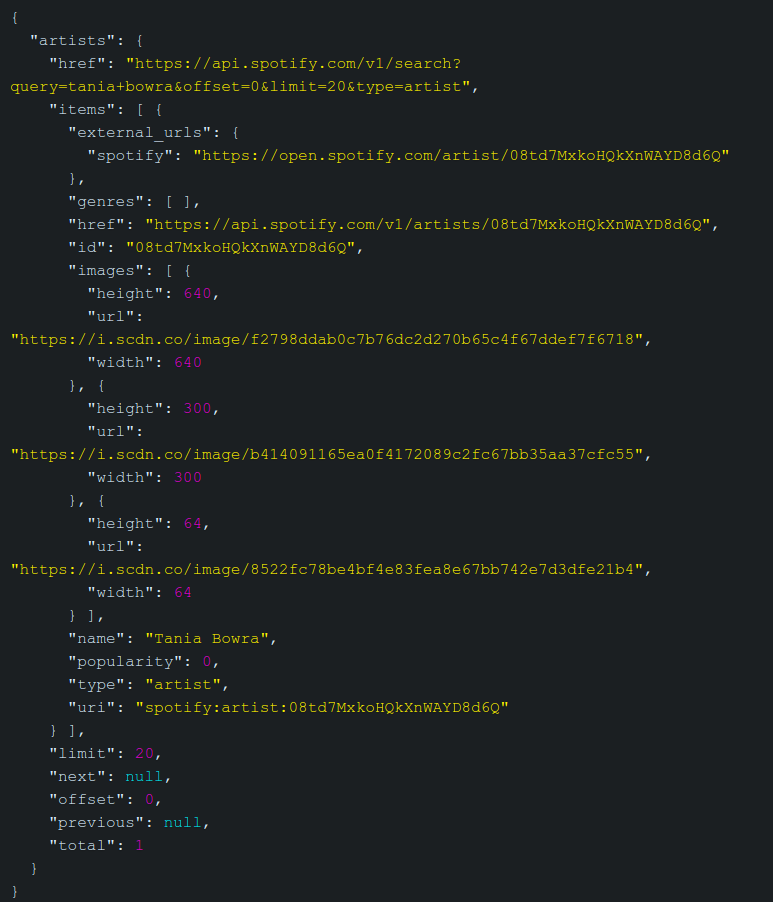
* The client is responsible for getting user authorization, getting user query and displaying the output of the server.

#### Response filtering / data object correlation

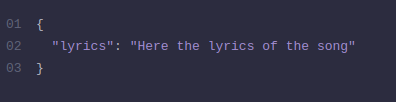
*For this project the data source is coming from 2 APIs. The first API is from Spotify, by using this API we can get the details of our music query. Such details include the song title, artist, popularity, preview url, song images, available market. The second API is from lyric.ovh, we are using this 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.*

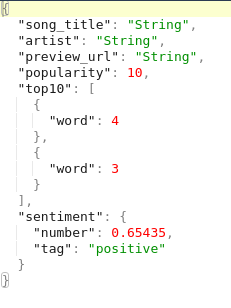
*Spotify Response:*



*Lyric.ovh Response:*



*The Response sent to the Client:*

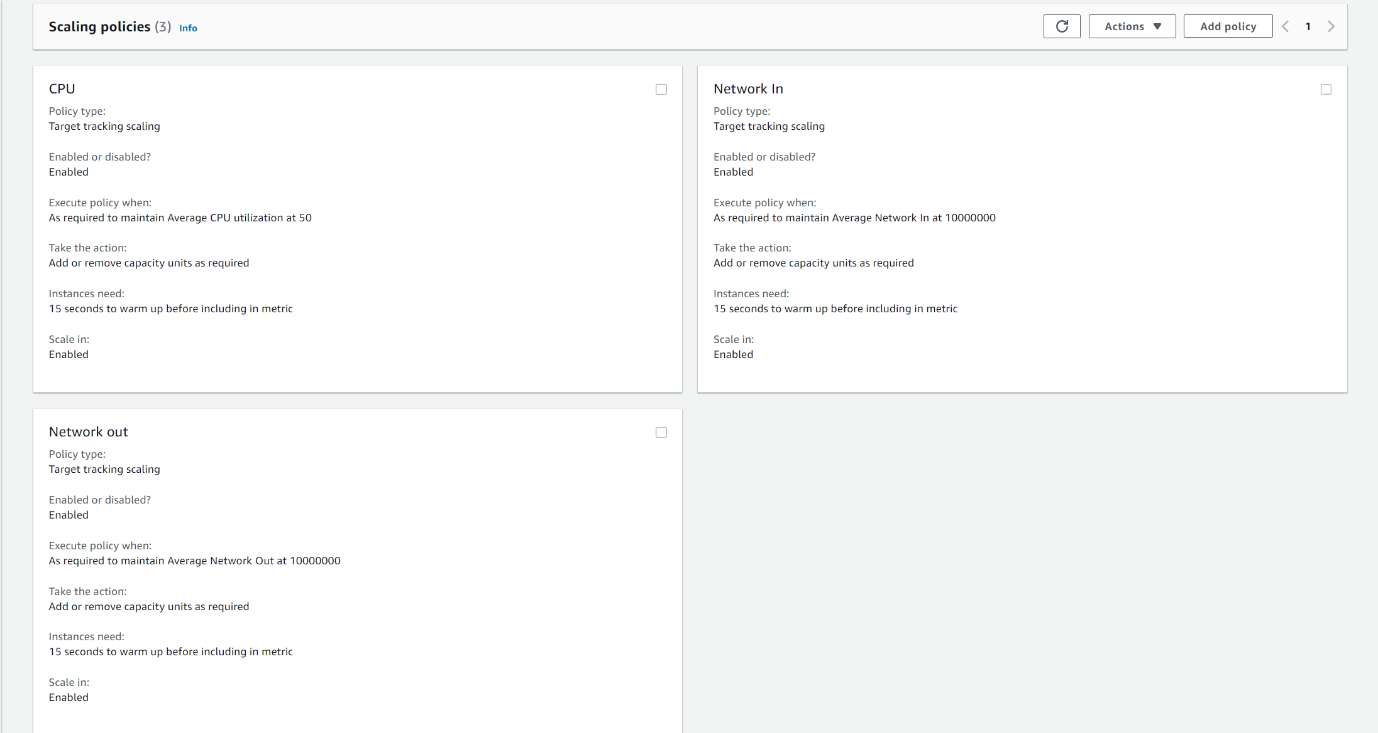


## Scaling and Performance

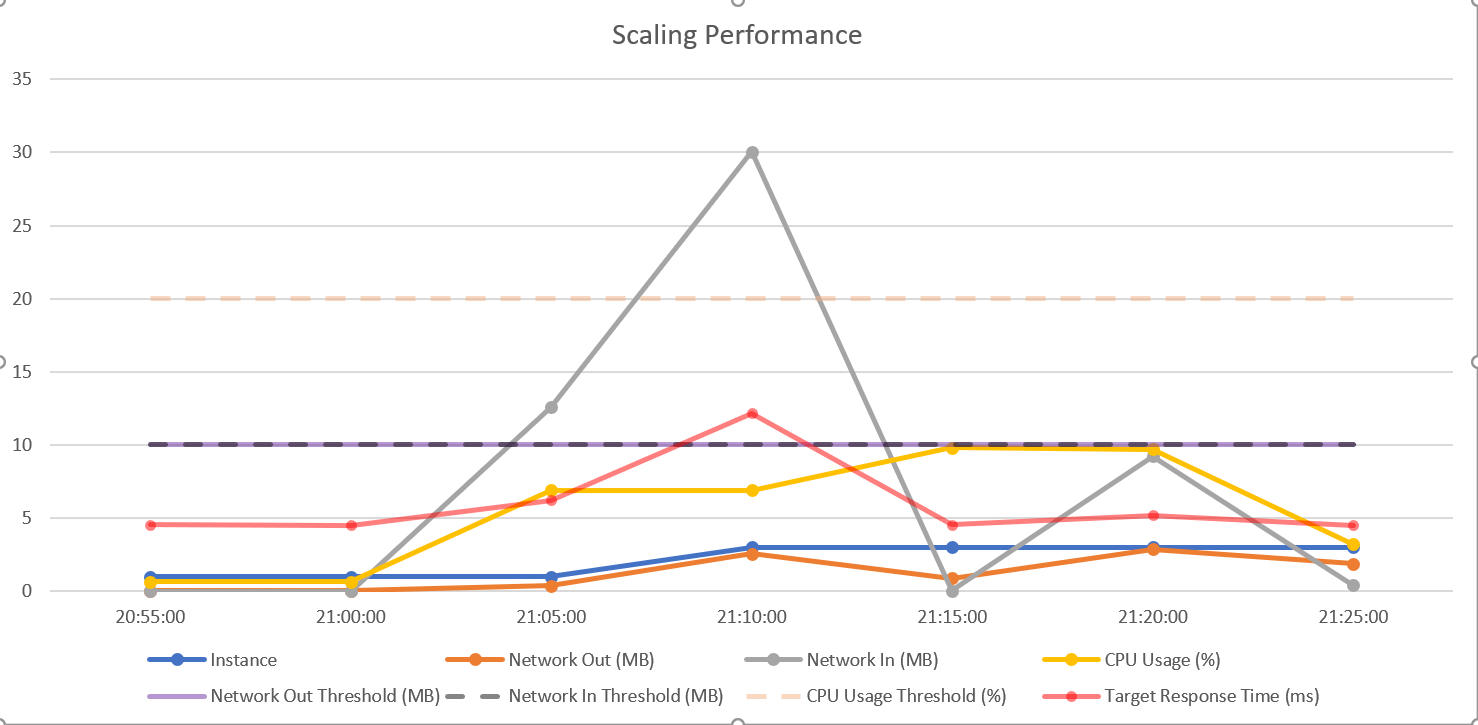
*For this project scaling policies, we decided that the policy is based on CPU utilization, Network In and Network Out. The reason why we choose this policy is:*

* + ***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 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 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

Graphical user interface, application

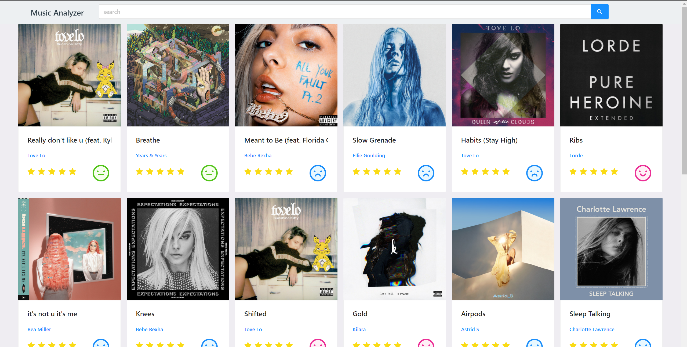
Description automatically generated

1. Login Spotify account

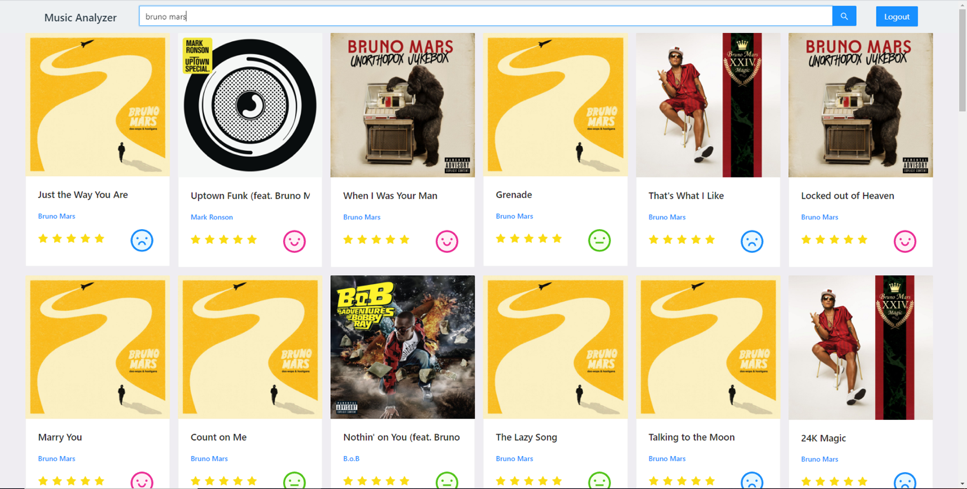
Graphical user interface, text, application, chat or text message

Description automatically generated

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



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



1. 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

Graphical user interface, application

Description automatically generated

1. Link to Spotify music playing page

A screenshot of a computer screen

Description automatically generated

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

Graphical user interface, website

Description automatically generated

# Appendix A

Text

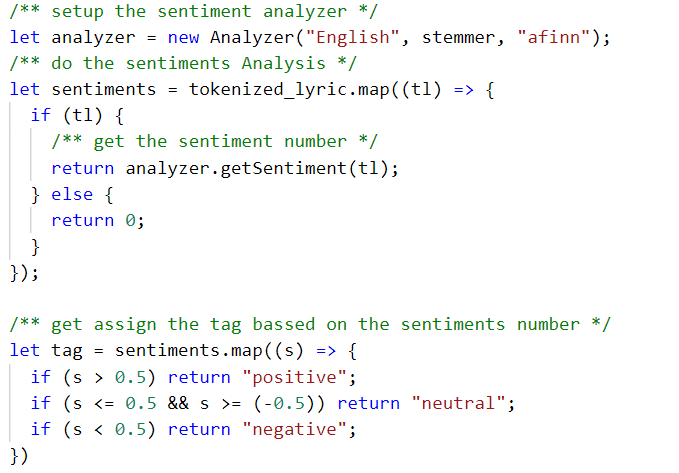
Description automatically generated

Figure

Text

Description automatically generated

Figure



Figure

Text

Description automatically generated

Figure

Text

Description automatically generated

Figure

# Appendix B

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure

Graphical user interface, website

Description automatically generated

Figure

Graphical user interface, application

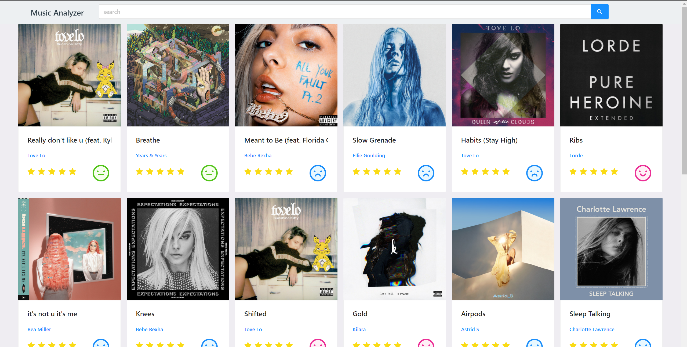
Description automatically generated

Figure

A screenshot of a computer screen

Description automatically generated

Figure



Figure

A screenshot of a computer

Description automatically generated

Figure

Graphical user interface, application

Description automatically generated

Figure