Cover page

SPADE Application Report

Table of Contents

SPADE Application Report

Student Contribution Agreement

Name: Isaiah Cuzzupe	**Name: Jessiah Malik**
Student No: s3743803	**Student No: S3731166**

Contributions:
- Backend
  - Spotify API Calls
  - Google API Calls
  - Google Service implementation
  - Firebase
  - Analytics
  - Translate
	Contributions:
	- Frontend
	  - Application Pages
	  - Handling user interaction
	  - Handling Call Responses
	  - Styling
	- Backend
	  - Google Analytics
	  - Data Sanitisation

Contribution Percentage: 50%	**Contribution Percentage: 50%**
Signature: **Date:**	**Signature:** **Date:**

Project Links

Application on GCP: https://spade-274202.ts.r.appspot.com/
Github Repo: https://github.com/etricity/CCAssignment2.git

SPADE Application Report

Summary

The purpose of SPADE is to provide an easy to use track searching dashboard for Spotify users. With one simple search, SPADE allows users to see a broad range of data specific to their search on a single web page, including track, artist and album details. From there users can then preview their searches & add the track to their library if so they wish to.
Further, SPADE provides users with personalized information visualising their activity on both Spotify & SPADE. User's can view their favorite tracks & artists on Spotify, as well as their most searched & saved song on SPADE.
Lastly, SPADE doubles as a Spotify Web Player. Users can connect to the player via the Spotify app and use SPADE as a remote player to listen to any song available through spotify.

Introduction

Motivations:

After looking at multiple different APIs, the Spotify API excited me the most, especially since I use Spotify almost everyday! I wanted to understand how something I used everyday actually works. As for the functionality of SPADE, I wanted to create something that had relatively little user interaction, but through that interaction, gave the user a lot to do, essentially getting the most out of the user input. Further, looking at similar applications that use the Spotify API, I did not find any that would present multiple sets of related data to users based on a single search.

Uses:

1. Track Search

SPADE allows users to search tracks and view a collective of data regarding their search on a single web page. This includes information about the searched track itself, as well as the track's artist & album.

2. Personalized Data Analysis

Users are able to visualise their activity on both SPADE & Spotify.
Presented information includes:

- Spotify Data:
    - User's top rated tracks
    - User's top rated artists
- SPADE Data:
    - User's search history & patterns
    - User's login activity

3. Spotify Web Player

Implementing the Spotify Web API, SPADE has functionality to act as a remote for the Spotify application. Users can connect to SPADE's web player & listen to their tracks on their web browser.

SPADE Application Report

Why is it required?
SPADE excels at presenting data sets related to a user's search and data analysis. With one search, the user is able to see data that would require many more searches & time if done through Spotify. This makes the interaction of use of SPADE very easy. In addition, the user is able to see their own search patterns based on their search history, reflecting their likes & dislikes over time. This is done in a different fashion to how Spotify measures the user's likes & dislikes, thus providing a different perspective on the user's activity.

How it can be used as real-life application?

SPADE allows a real life insight into data as an analytic tool for music, providing users with valuable insights that spotify itself doesn't usually provide. Allowing access to the entire database of user searches we can find trends in people's curiosities, which popular music is searched, music genres and artists which are popular in each country. Artists can see how popular they are and how often people are querying their music. The app is also relevant in the real world to allow users to see their real time top songs, allowing the

v. The advantages/positive/new things of your application.
SPADE revolutionises the nature with which we listen to our favourite music, giving users the ability to combine their favourite music as a core to a playlist and develop that playlist with any available music, with an accessible front end allowing translation of key words over nearly every language and large clear buttons, using the site on a mobile platform is simple and coherent.

SPADE Application Report

Related Work

https://developer.spotify.com/community/showcase/

To gather ideas and inspiration, our teams looked at the community's showcase of applications that implement the Spotify API. These can be found on the Spotify website.

Notable Inspirations:

Application: Musical Data
Submitted by: Rutger Ruizendaal
Tools used: Spotify Web API

Description: *"Musical Data is a web app that allows users to quickly gather and visualize data of their favorite tracks, albums and artists. All you have to do is select the track you're interested in and Music Data will provide you details on the track and its corresponding album and artist. Users can get insights into audio features as well as compare the top tracks of the artist per country.*

Musical Data uses Spotify's Web API to gather information on audio features, artist's top tracks, the loudness per track and more data." (sourced from link below).

Link: https://developer.spotify.com/community/showcase/musical-data/

Application: Musical DataSpotify Audio Analysis
Submitted by: Hugh Rawlinson
Tools used: potify Web API, Web Playback SDK

Description: *Learn more about the audio properties of your favourite tracks, including detailed rhythmic information.*

You can see each section, bar, beat, segment, and tatum on a timeline, skip to each timestamp, and see the pitch and timbre vectors for the current segment.

To get these values, we use the Spotify API's Get Audio Analysis for a Track endpoint. (sourced from link below).

Link: https://developer.spotify.com/community/showcase/musical-data/

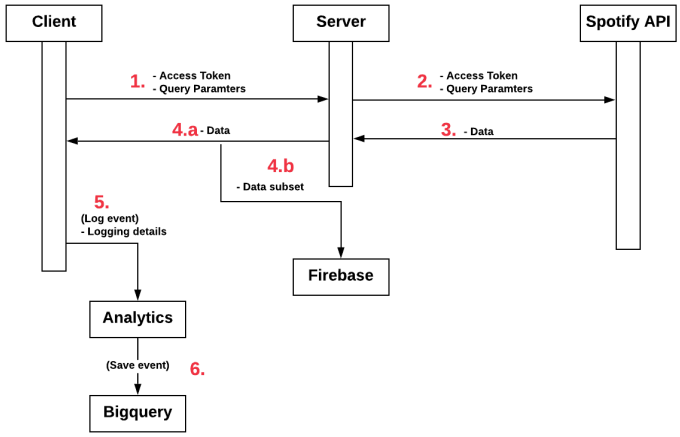
SPADE Application Report

Software Design/Architecture

Diagrams

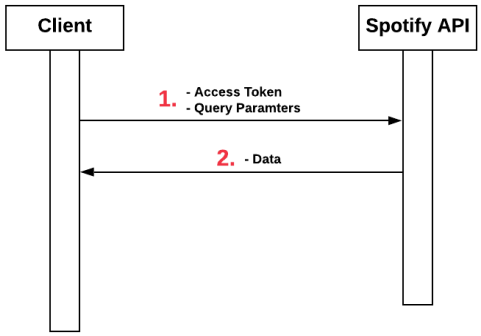
Below is an example of a request for data to the Spotify Endpoint. The same flow is used for a number of requests in our application. These include getting the user's Spotify Profile information & getting the search data. The example below will assume the user has made a track search.

1. Client sends request to server with Spotify Access Token & the user's search terms
2. Server sends request to Spotify Endpoint passing the Access Token & search terms
3. Spotify responds to the server with the result
4. A. Server formats data & sends it back to the client
   B. Server saves a subset of the received data in Google Firebase
5. Client-side Google Analytics records the search event
6. Analytics records the event in Bigquery



There are some instances where the client will request data directly from the Spotify endpoint. This is mostly done in instances where the requested data does not need to be to be concatenated or formated. For example, the data requests on the "Analytics" page of SPADE are all Client → Spotify endpoints (see diagram below).

1. Client sends request to the serSpotify Endpoint with Spotify Access Token & filter parameters
2. Spotify responds to the Client with the result



SPADE Application Report

APIs:

Spotify APIs:

1. Spotify Web API

   The Spotify Web API endpoints return JSON data about music artists, albums, and tracks, directly from Spotify's Databanks. Listed below are the endpoints and functionality utilized by our application.

   a. User Profile API
      i. Get data for current user's profile
      ii. Get data for another user's profile.
   b. Search API
      i. Get Spotify Data about tracks, artists, albums based off user's search terms
   c. Playlists API
      i. Create a new playlist in the user's Spotify Account
      ii. Add songs to a playlist in the user's Spotify Account
   d. Personalization API
      i. Get user's top rated tracks & artists as calculated by Spotify

2. Spotify Web Player SDK

https://developer.spotify.com/documentation/web-playback-sdk/

As stated from the link above

*"**Important:** The Web Playback SDK is currently in Beta. The content and functionality may change without warning in future versions."*

Due to the Web Player SDK limited functionality & beta status, it is not the main focus of our application but acts more as a small piece of extended functionality our team wanted to implement.

The Spotify Web Player SDK allows users to create a new Spotify Player & stream their music directly through their browser. Our implementation includes basic controls, making it easy for user's to control their listening experience. Controls include:

- Play/Pause
- Increase/Decrease Volume

SPADE Application Report

Google Cloud Services:

1. Cloud Logging API

https://cloud.google.com/logging/docs/reference/v2/rpc

The Cloud Logging API writes log entries for our web application that can be viewed in the log viewer on the google cloud platform console. This was used to ensure the correct data was being retrieved to and from multiple endpoints in our application & to ensure certain functionality was working as expected.

2. Cloud Pub/Sub API

https://cloud.google.com/pubsub/docs/

Cloud Pub/Sub is an asynchronous messaging service. In our project, it was used as part of Firebase Analytics for event ingestion and delivery for streaming analytics pipelines.

3. Cloud Translation API

https://cloud.google.com/translate

Cloud Translation API allows for dynamically translating between languages using Google machine learning. SPADE utilizes this to translate crucial information to the user's preferred language, increasing the accessibility of SPADE to users around the globe.

4. App Engine Admin API

https://cloud.google.com/appengine/docs/admin-api

Google Engine Admin API allows developers to build highly scalable applications on a fully managed serverless platform. This allowed our team to continue developing code for the application without needing to work about managing underlying infrastructure. Once a newer stable version of the application was complete, we could simply re-deploy to the cloud.

This API was also used as part of authentication when accessing other Google Cloud Services.

5. Cloud Firebase

https://firebase.google.com/

Cloud FIrebase acted as a NoSQL Database that allowed us to store, sync and query data on any instance of our application. The database was realtime, meaning we could easily test & deploy the database with ease. Our team utilized the database to store a number of elements relating to the user, including their login count, search history & basic profile facts, which would later be used for data visualisation.

6. Google Analytics

https://developers.google.com/analytics

Google Analytics measures how the user interacts with your application. Through default & custom events, our team has access to a vast pool of data retained to our user base & our application. We can see where users are from (country wise), what pages they are visiting (dashboard, activity, web player) & what actions they are performing (searching, saving).

7. Google Bigquery

https://cloud.google.com/bigquery

Bigquery is a SQL Cloud Data warehouse. Our team linked out Google Analytic events to Bigquery, allowing all events triggered by users to be stored in Bigquery in daily reports. As a result, Bigquery stored information pertaining to activity on the application as a whole, rather than information pertaining to individual users.

8. Google Charts

https://developers.google.com/chart

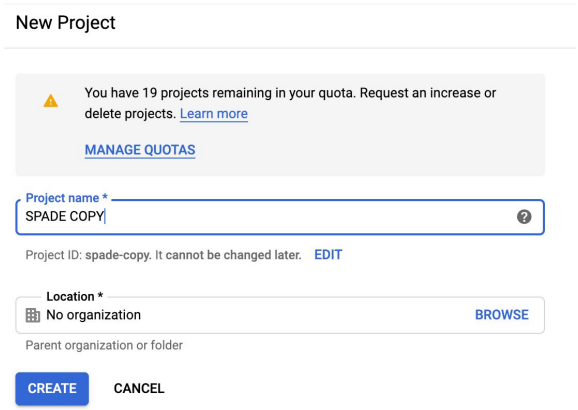
Google Charts was our data visualisation API of choice. The data displayed in the interactive graphs was pulled from our realtime Firebase, allowing it to stay scalable & up to date. The charts on SPADE's activity page allows the user to view their activity on SPADE, including their most searched & saved songs.

SPADE Application Report

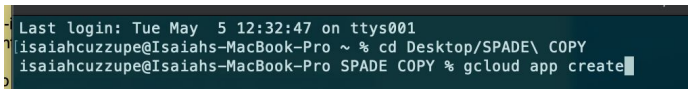
Implementation (Developer Manual)

The following provides the general steps to downloading & replicating the SPADE Application

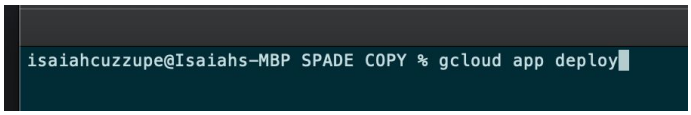
1. Create a new project on GCP



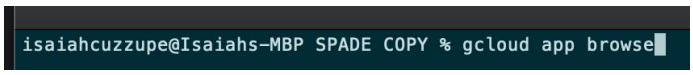
2. Ensure billing is enabled for the project
   https://console.cloud.google.com/billing
3. Download the project code on Github
   https://github.com/etricity/CCAssignment2.git
4. Download & install Google Cloud Console tool & ensure it is configured for your project (account & project id)
   https://cloud.google.com/sdk
5. Navigate to the project directory in terminal and run commands
   a. gcloud app create



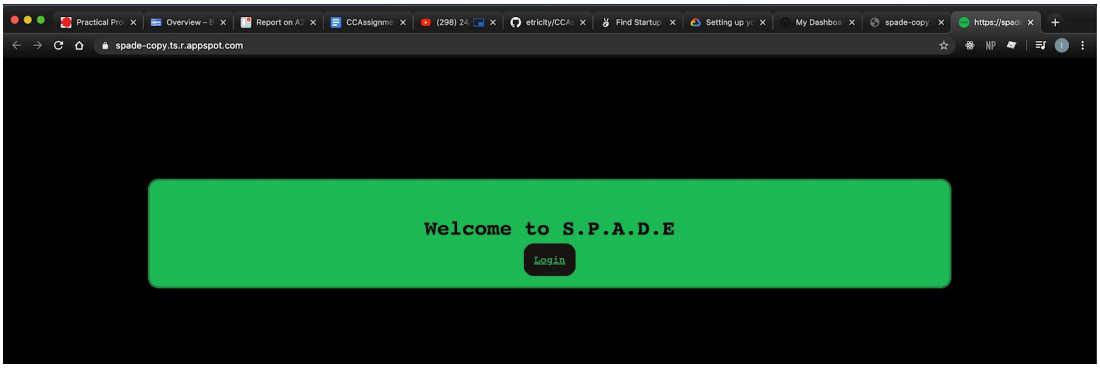
   b. gcloud app deploy



   c. gcloud app browse



SPADE Application Report

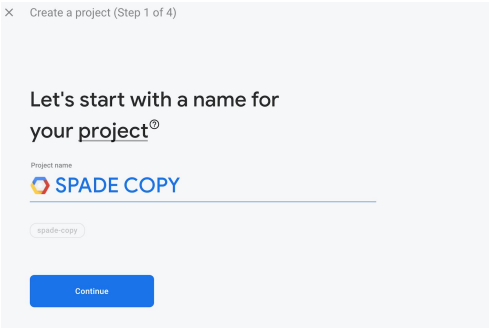


6. Head on over to https://developer.spotify.com/ to register a client id with Spotify API (allows you to access Spotify API)
7. Set up Spotify API Callbacks on your Spotify Developer Dashboard.
   Navigate to: Dashboard → edit settings
   a. Add both:
      i. http://localhost:8080/spotify/callback
      ii. https://[PROJECT-URL]/spotify/callback

8. Refactor the Spotify authentication variables in ".backendJS/spotify.js" with your own ids & urls

```
18  var client_id = '[CLIENT-ID]';
19  var client_secret = '[CLIENT-SECRET]';
20  var redirect_uri = '[REDIRECT-URL]';
21  var stateKey = 'spotify_auth_state';
```

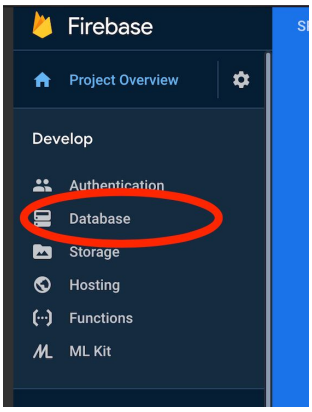
9. Add your own firebase to the project
   Go to https://console.firebase.google.com & create a new firebase project using your existing Google Cloud Project



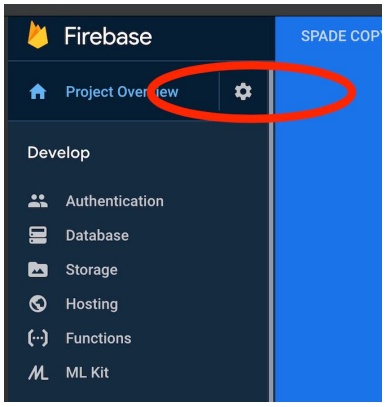
SPADE Application Report

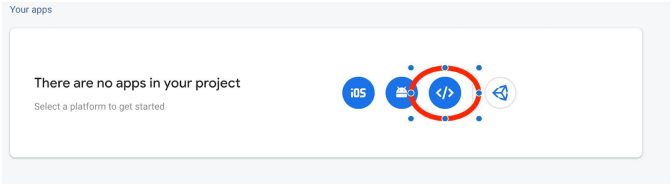
10. On your Firebase Console following the setups below in order:

a. Go to firebase → database → create real time database

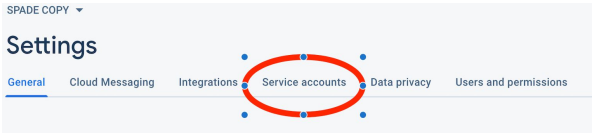


b. Go to firebase → settings → add application



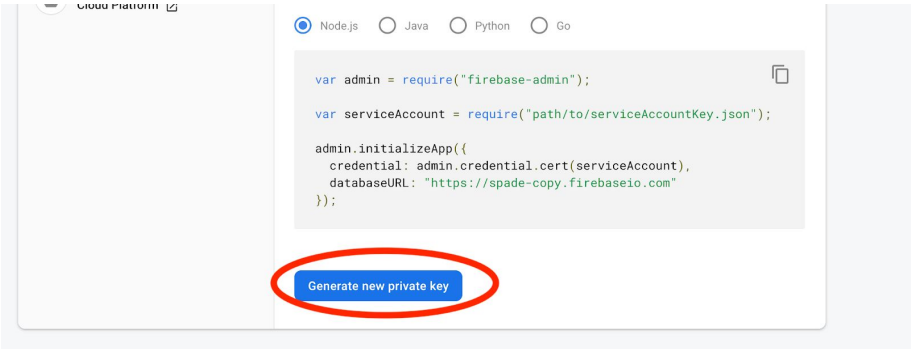


c. Go to firebase → settings → service account



d. Create a service account for firebase, download the key and replace it with the current keys in /key folder in the project files

e. Replace the "Admin SDK configuration snippet" in firebase.js with the one found in your firebase settings

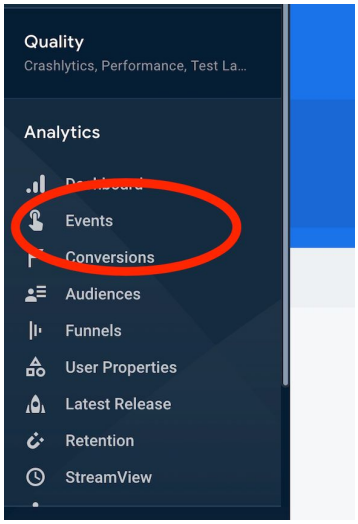


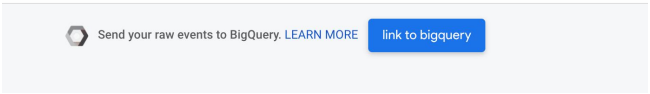
SPADE Application Report

f. Go to Firebase → settings → general &
Replace the code in analyticsConfig.js with the “Firebase Configuration Object”

```
const firebaseConfig = {
  apiKey: "AIzaSyADVhN1iQfBh8MJiaUFEwFEMhC3RoycXoc",
  authDomain: "spade-copy.firebaseapp.com",
  databaseURL: "https://spade-copy.firebaseio.com",
  projectId: "spade-copy",
  storageBucket: "spade-copy.appspot.com",
  messagingSenderId: "701223864729",
  appId: "1:701223864729:web:290b79cb9c0aa53e442b08",
  measurementId: "G-XE0BFNSJ1X"
};
```

g. Go to Firebase → events & click link to big query





11. Enable APIs

APIs should enable themselves as needed. Ensure these APIs are enabled.

Name
Cloud Logging API
Cloud Functions API
Cloud Pub/Sub API
Cloud Translation API
App Engine Admin API
Firebase Installations API
Firebase Extensions API
Firebase Mods API
BigQuery API

SPADE Application Report

12. Deploy to Google Cloud Platform

NOTE: ensure the “http://localhost:8080/spotify/callback” is commented out before deploying to gcloud (this is for local testing only).

Run Commands in directory of project:

a. gcloud app deploy
b. gcloud app browse

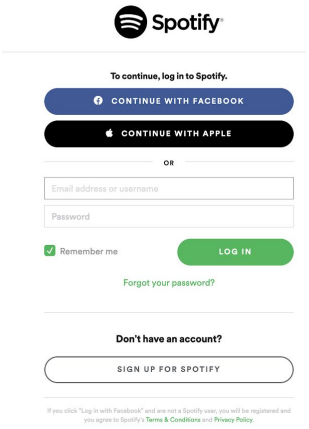
SPADE Application Report

User Manual

The following provides the general steps to using the application functionality.

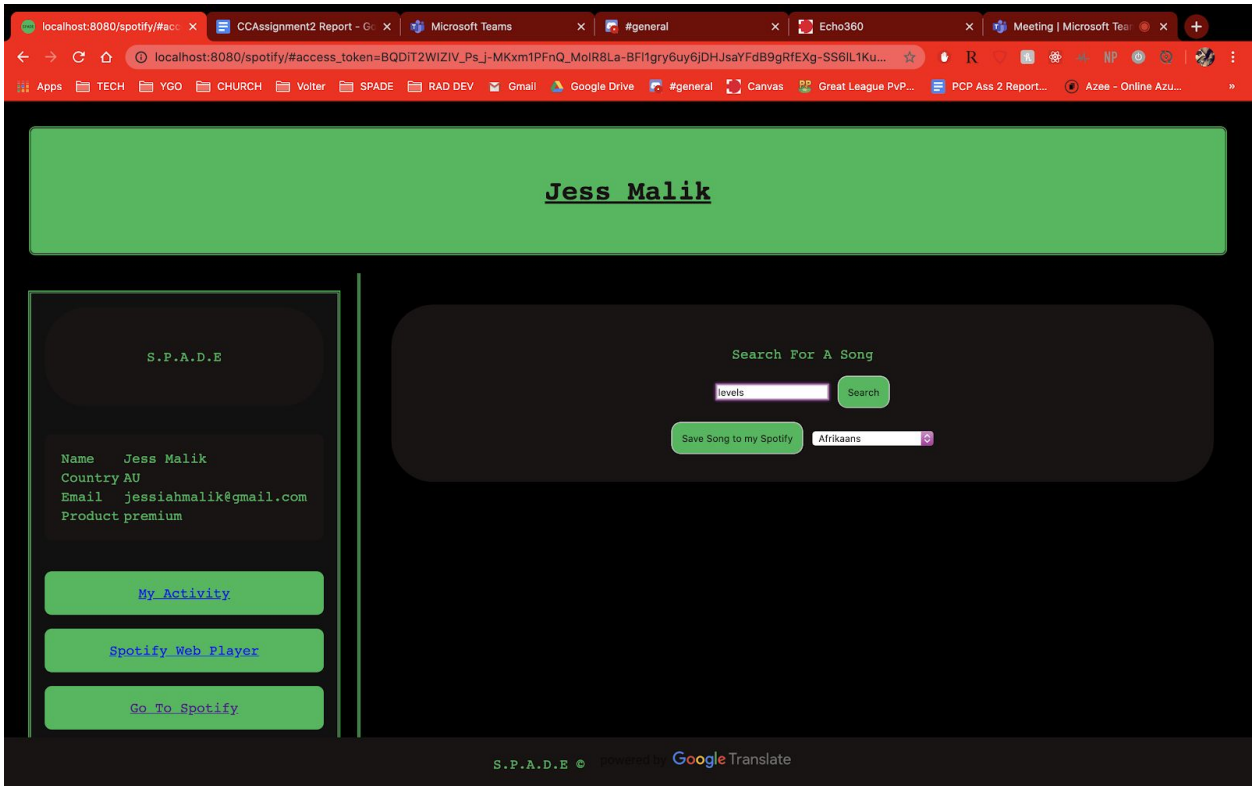
1. Login From The Root Page

2. Success Will Redirect you to the spotify Login Harness To Either Provide login credentials or Regsiter



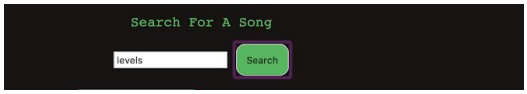
SPADE Application Report

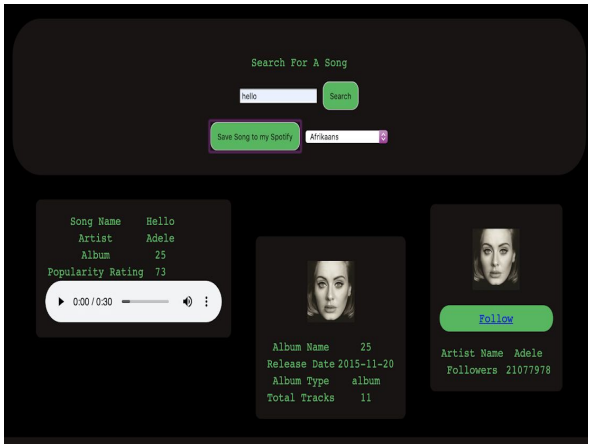
3 Upon Sucess You Will be redirected to the Dashboard



4. Dashboard interaction:

-Search: An action That will Provide Results to a Queried Song Title ,
the results of which provides data about the song, album and artist most relevant to the queried String



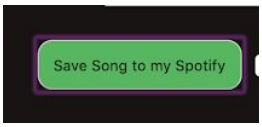


SPADE Application Report

-Translate: An action that translate key text on the page to specified

languages on selection.

-Add Song To Spotify: Creates a Playlist named SPADE if not existent then adds song to the playlist:



5. Side Nav:

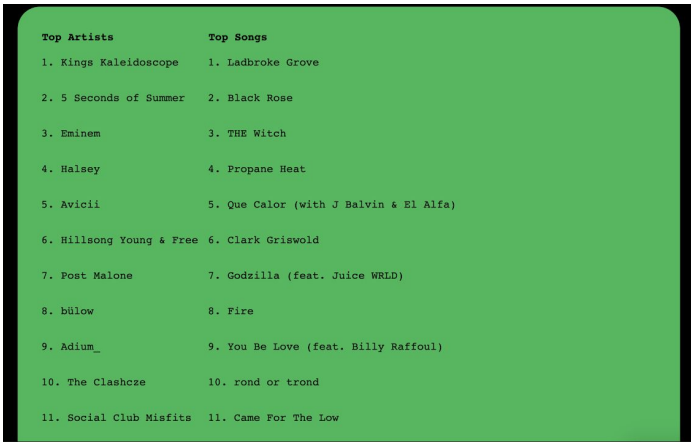
-Dashboard: Redirects to the Dashboard
-My Activity: Redirects to the activity Page
-Spotify Web Player (not Shown): redirects to the Spotify Web player.

SPADE Application Report

6. My Activity

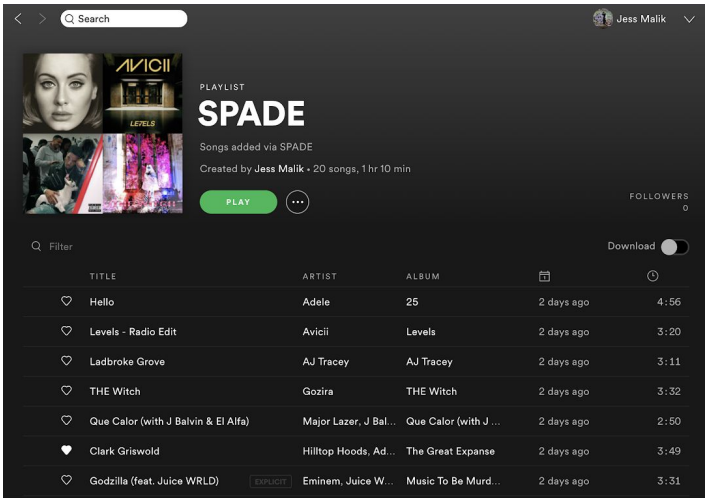
This Page Shows Visualisation of your searched Music and Added songs through the application, Displaying aslo the Count of how many times you have logged in

The Page also includes a summary of your top 20 Songs and artists over the last 6 months, this data from the spotify database, not the application interaction.



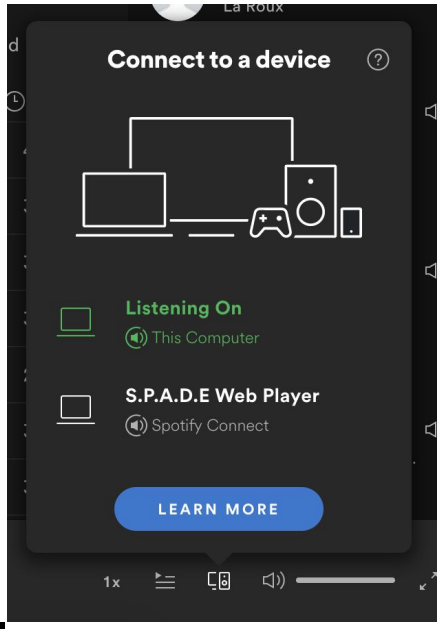
Top Artists	Top Songs
1. Kings Kaleidoscope	1. Ladbroke Grove
2. 5 Seconds of Summer	2. Black Rose
3. Eminem	3. THE Witch
4. Halsey	4. Propane Heat
5. Avicii	5. Que Calor (with J Balvin & El Alfa)
6. Hillsong Young & Free	6. Clark Griswold
7. Post Malone	7. Godzilla (feat. Juice WRLD)
8. bülow	8. Fire
9. Adium_	9. You Be Love (feat. Billy Raffoul)
10. The Clashcze	10. rond or trond
11. Social Club Misfits	11. Came For The Low

The final Functionality of the page is the button that adds your top 20 songs automatically to the SPADE Spotify Playlist.

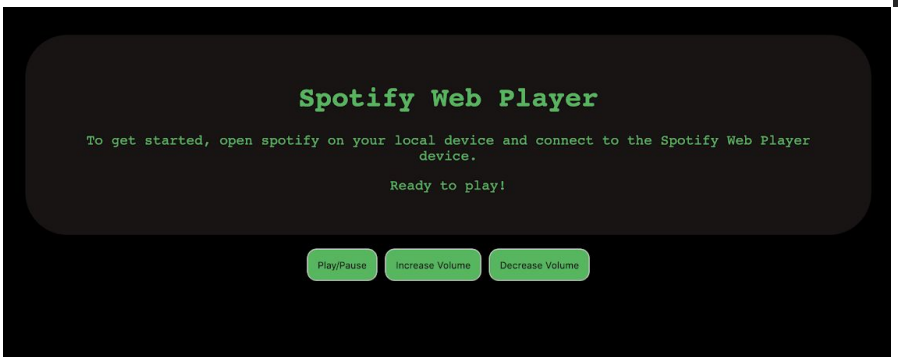


SPADE Application Report

7. The Spotify Web Player: The Apllications Active External interaction is the way in which you can control Music Playing throught the spotify web player.



- Load the Web Player Page and wait for “Ready To Play! ”Message.
- Open Spotify App
- Open “Connect To A Device” Setting Found in the bottom left hand side of the Spotify App
- Select S.P.A.D.E Web Player, it will be highlighted green.
- Return to the S.P.A.D.E Web app, You should now have full control over music playing and its volume using the respective buttons



8. App Logout Can Be Performed from the respective link in the side nave.
   - Logging out will redirect User to the Log In page

SPADE Application Report

References

Many of these references were used as learning material by our team to gain the necessary knowledge to develop our application.

[1]V. Ofoegbu, "The only NodeJs introduction you'll ever need.", *codeburst.io*, 2018. [Online]. Available: https://codeburst.io/the-only-nodejs-introduction-youll-ever-need-d969a47ef219. [Accessed: 12- Apr- 2020].

[2]"Run Express.js on Google App Engine Flexible Environment", *Google Cloud*, 2020. [Online]. Available: https://cloud.google.com/community/tutorials/run-expressjs-on-google-app-engine. [Accessed: 13- Apr- 2020].

[3]P. Guo, *Frontend setup and basic Ajax (Node.js+Express Part 4)*. 2018.

[4]Academind, *Node.js Basics*. 2020.

[5]Y. Agrawal, "Methods for defining functions in JavaScript - LogRocket Blog", *LogRocket Blog*, 2019. [Online]. Available: https://blog.logrocket.com/defining-functions-in-javascript/. [Accessed: 21- Apr- 2020].

[6]"Google Cloud Platform | Documentation", *Google Cloud*, 2020. [Online]. Available: https://cloud.google.com/docs. [Accessed: 05- Apr- 2020].

SPADE Application Report

Video Demonstration

Developer Demo:
https://www.youtube.com/watch?v=bBBGnOtcYQY

User Demo:
https://www.youtube.com/watch?v=iVhzt4fzI34