2021

Live Video Search

CAB432 Assignment 1

Matthew Weir

N10509020

9/17/2021

Contents

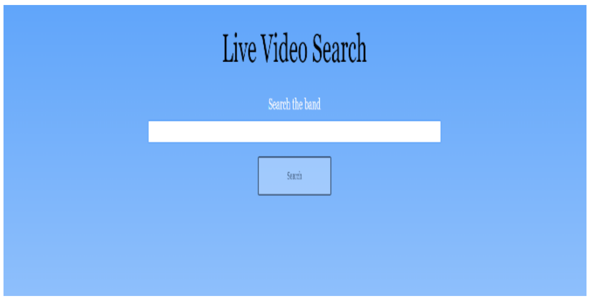
This report shows an overarching idea of the application and progressively get more in depth about the technicalities of how the application is designed and organised. The document will also contain a user guide and a testing suite.

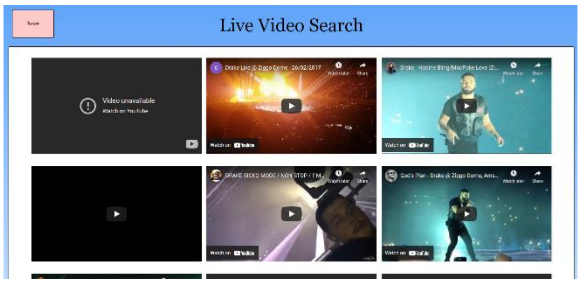
Mashup Purpose & description

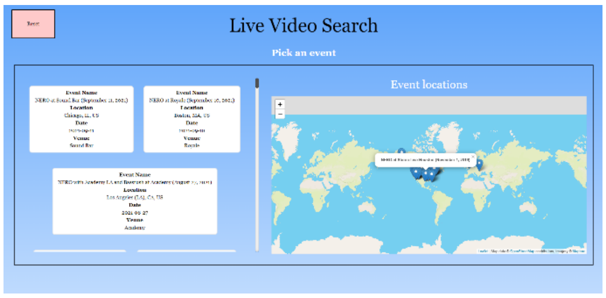
Live Video Search

This application is a single page lightweight application that combines multiple API services to bring a rich user experience.

The application allows a user to search for band/artist and receive information about their past events, see where those events are located on an interactive map, and from there they can choose an event to receive live videos from YouTube of the event.







Services used

Songkick Artist Search

Returns a collection of Artist ID's matching a given query

Options: totalEntries, perPage, page

Endpoint:
https://api.songkick.com/api/3.0/search/artists.json?apikey={your_api_key}&query={artist_name}

Docs: https://www.songkick.com/developer/artist-search

Songkick Artists past events (gigography)

Returns a collection container details of a given artists past events.

Options: ArtistID, min_date, max_date, page, per_page, order

Endpoint:
https://api.songkick.com/api/3.0/artists/{artist_id}/gigography.json?apikey={your_api_key}

Docs: https://www.songkick.com/developer/past-events-for-artist

LeafletJS (mapbox)

Retrieves the tiles to display on the leaftletJs interactive map

Options: username, style_id, zxy (zoom, column, row), tilesize, @2x (render speed).

Docs: *https://docs.mapbox.com/api/maps/static-tiles/*

Endpoint: https://api.mapbox.com/styles/v1/{id}/tiles/{z}/{x}/{y}?access_token={accessToken}

Google API (YouTube Search)

Retrieves an response object containing each video that matches the queries, (Video name, and channel id)

Options: Check docs, there are expansive options. (I used part, q (query), maxResults)

Docs: https://developers.google.com/youtube/v3/docs/search/list

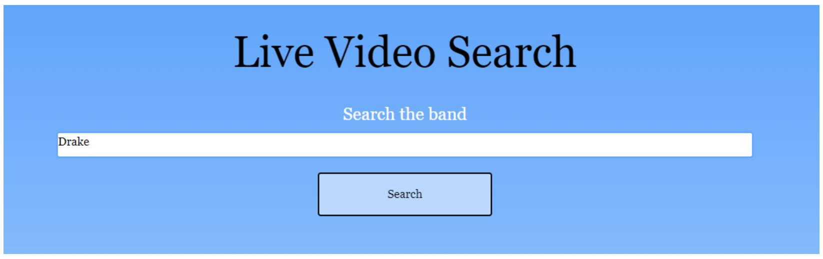
Endpoint: https://developers.google.com/youtube/v3/docs/search/list

Mashup Use Cases and Services

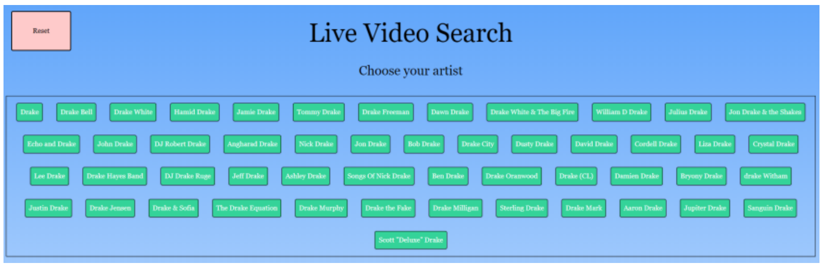
See Live Videos

As a	User
I want	To search for a band/artist to see their old events
So that	I can click on one of the events and see live videos

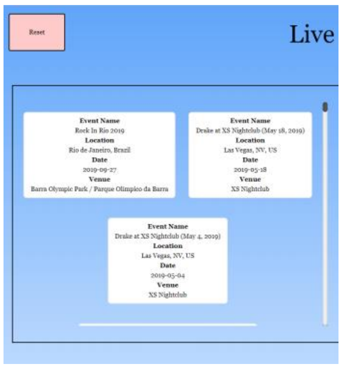
1. Once a user has entered their desired band/artist,
2. that query is passed to the Songkick artist search API which returns a list of artists,
3. The user is then able to choose an artist that better matches their query by clicking on that tile,
4. Upon clicking on that tile, the artist id is passed to the Songkick past events API,
5. this returns a scrollable list of events the user can click one,
6. which sends a query to the YouTube API to return a page of imbedded YouTube videos of that event.



Step 1 Users input



Step 2,3,4 (Query = Drake, That is sent to the Songkick Artist Search API, Returns artist Id's and names)

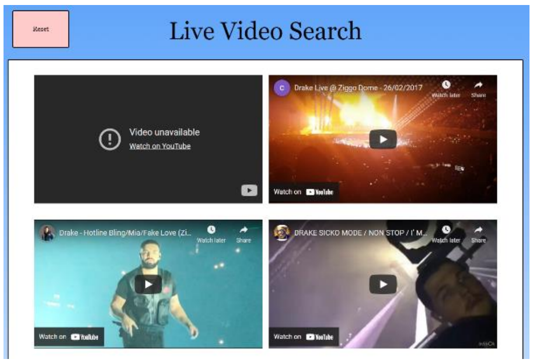


Event Name
Rock In Rio 2019
Location
Rio de Janeiro, Brazil
Date
2019-09-27
Venue
Barra Olympic Park / Parque Olímpico da Barra

Event Name
Drake at XS Nightclub (May 18, 2019)
Location
Las Vegas, NV, US
Date
2019-05-18
Venue
XS Nightclub

Event Name
Drake at XS Nightclub (May 4, 2019)
Location
Las Vegas, NV, US
Date
2019-05-04
Venue
XS Nightclub

Step 5, (Query = Drake's Artist ID, returns a set of his past events)

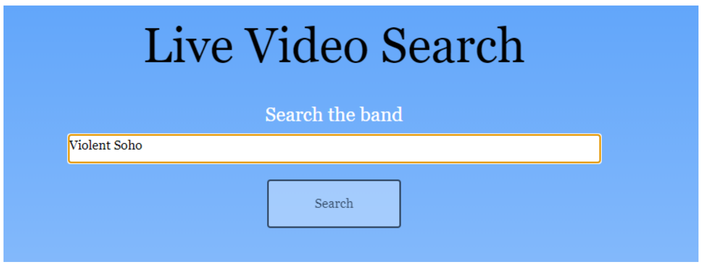

Reset
Live Video Search
Video unavailable
Watch on YouTube
Drake Live @ Ziggo Dome - 26/02/2017
Drake - Hotline Bling/Mia/Fake Love (Zi...
DRAKE SICKO MODE / NON STOP / I' M...
Watch later
Share
Watch on YouTube

Step 6, (Query = The past event that was clicked on, sent to YouTube API to return a set of video identifiers which are imbedded into the page)

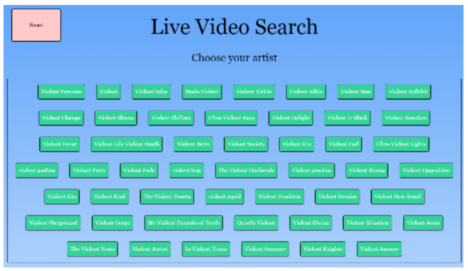
See events location

As a	User
I want	to enter a bands name into the system, and it to display map showing the locations of the event
So that	I can better choose which events live videos I want to see

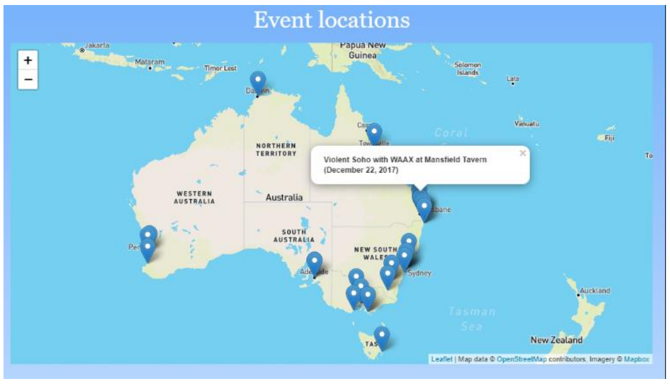
1. Once a user has entered their desired band/artist,
2. that query is passed to the Songkick artist search API which returns a list of artists,
3. The user is then able to choose an artist that better matches their query by clicking on that tile,
4. Upon clicking on that tile, the artist id is passed to the Songkick past events API,
5. Upon receiving the past events those events lat/long and event description is used to create markers on the LeafletJS map,



Users input



Step 2,3,4 (Query = Violent Soho, That is sent to the Songkick Artist Search API, Returns artist Id's and names)



Step 5 (Query = Violent Soho's Artist ID) Retrieves the events lat long and description and plots the markers on the map

Technical breakdown

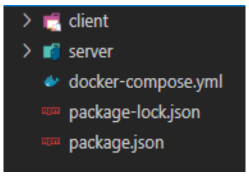
This application is built on top of a very popular technical stack, React JS for the front end, and Node.js Express for the back end. These two are then converted into a docker container and in turn hosted on AWS.

I chose React JS as a I have experience in using the framework and find it easy to pass the data around between each component. The website is a single page application that works in stages as the user passes in new queries and receives new results.

This data is all received by sending fetch requests to the Express server on the back end. The express server is fetching data from two private API's requiring keys, (SongKick & Google).

Architecture and Data Flow

The applications file structure at the top directory is simple, there is a file each for the client (Front end) & server (Back end).

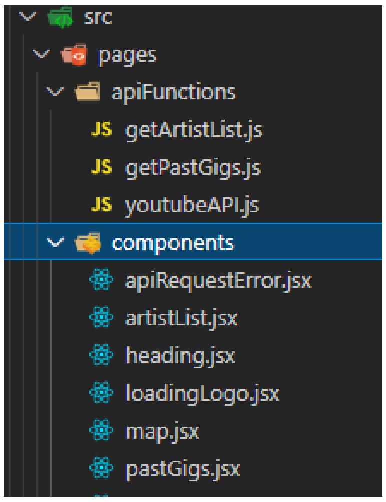


```
> client
> server
  docker-compose.yml
  package-lock.json
  package.json
```

Overall file structure

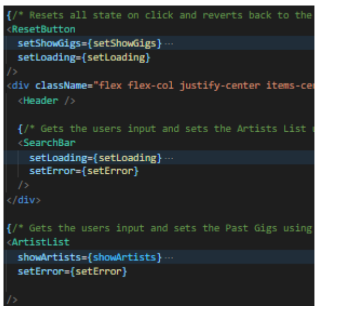
Digging deeper into the Client file, this is where the React Application has been generated and altered. The application is organised well into a lot of components that are shown throughout the different stages of the application. These components are all called from within the home.js base file.

Above the components is an API Functions folder containing three JavaScript files that I use to fetch data from different API endpoints on the backend, I have split them up for reusability and code quality.



```
v src
  v pages
    v apiFunctions
        getArtistList.js
        getPastGigs.js
        youtubeAPI.js
    v components
        apiRequestError.jsx
        artistList.jsx
        heading.jsx
        loadingLogo.jsx
        map.jsx
        pastGigs.jsx
```

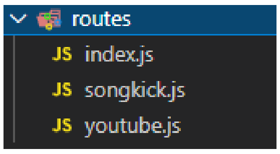
React file structure



```
{/* Resets all state on click and reverts back to the
<ResetButton
  setShowGigs={setShowGigs} ···
  setLoading={setLoading}
/>
<div className="flex flex-col justify-center items-ce
  <Header />

  {/* Gets the users input and sets the Artists List
  <SearchBar
    setLoading={setLoading} ···
    setError={setError}
  />
</div>

{/* Gets the users input and sets the Past Gigs using
<ArtistList
  showArtists={showArtists} ···
  setError={setError}

/>
```

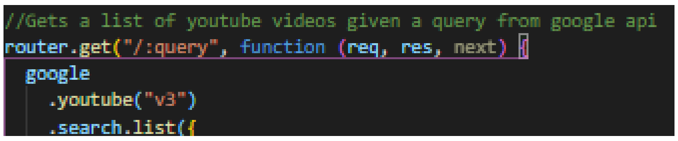
Home.js

These API endpoints are configured from within Express's routes folder, this folder contains two JavaScript files that contain the get request functions that are needed to serve the front end.



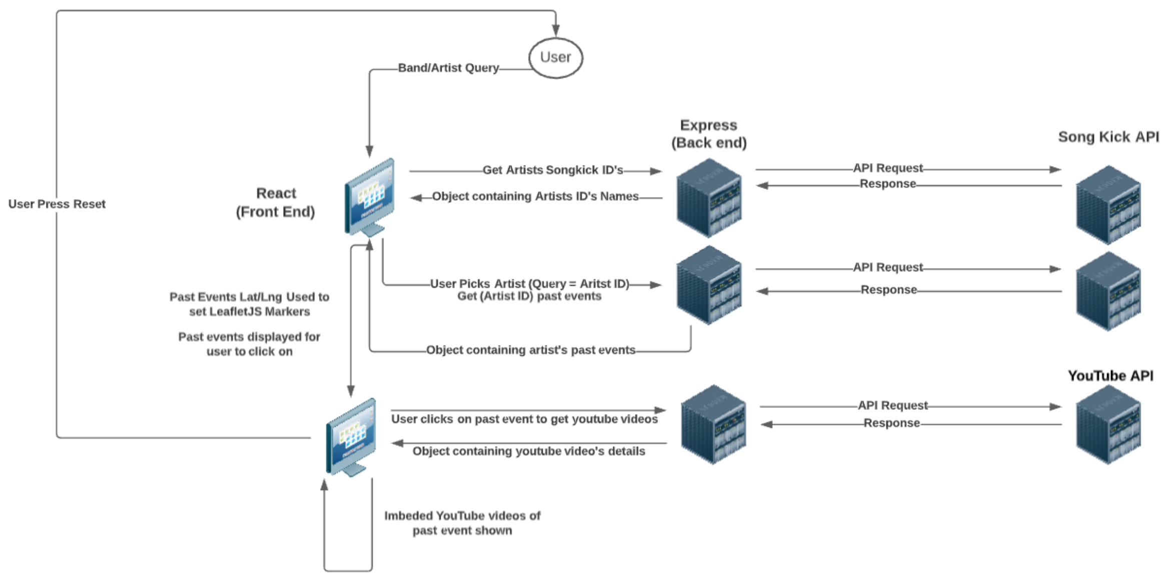
```
v routes
    index.js
    songkick.js
    youtube.js
```

Express Routes



```
//Gets a list of youtube videos given a query from google api
router.get("/:query", function (req, res, next) {
  google
    .youtube("v3")
    .search.list({
```

Express Get Function


User
Band/Artist Query
User Press Reset
React
(Front End)
Express
(Back end)
Song Kick API
Get Artists Songkick ID's
Object containing Artists ID's Names
API Request
Response
User Picks Artist (Query = Aritst ID)
Get (Artist ID) past events
Past Events Lat/Lng Used to set LeafletJS Markers
Past events displayed for user to click on
Object containing artist's past events
YouTube API
User clicks on past event to get youtube videos
Object containing youtube video's details
Imbeded YouTube videos of past event shown

Live Video Search Data Flow Diagram

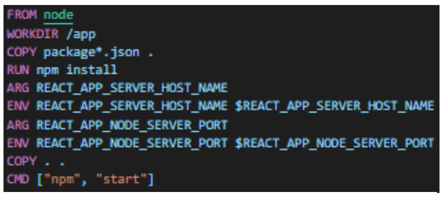
Deployment and the Use of Docker

The application has been able to be "dockerised" by putting both the Client and Server in their own Docker containers.

The Server dockerfile and Client dockerfile at very similar in that they both set there working directory, copy over their package.json files & then NPM install. This is done first before copying over the rest of the application as docker takes advantage of caching in layers. The also both declare environment variables so that when the containers are run the user can set run time specific variables such as port number and hostname.

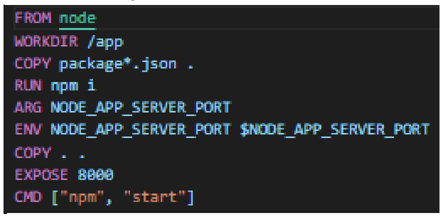
For simplicity of building and testing the containers during development there is also a docker compose file at the root directory that can build and run both containers in one command (docker compose up/docker compose build).

However, when running the containers on AWS or anywhere else you will need to use (docker run) and set the environment variables or they will be set to default values, Client (REACT_APP_SERVER_HOST_NAME = localhost, REACT_APP_NODE_SERVER_PORT = 8000) & on the server (NODE_APP_SERVER_PORT= 8000).



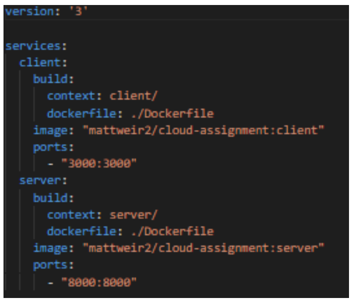
```
FROM node
WORKDIR /app
COPY package*.json .
RUN npm install
ARG REACT_APP_SERVER_HOST_NAME
ENV REACT_APP_SERVER_HOST_NAME $REACT_APP_SERVER_HOST_NAME
ARG REACT_APP_NODE_SERVER_PORT
ENV REACT_APP_NODE_SERVER_PORT $REACT_APP_NODE_SERVER_PORT
COPY . .
CMD ["npm", "start"]
```

Client Dockerfile



```
FROM node
WORKDIR /app
COPY package*.json .
RUN npm i
ARG NODE_APP_SERVER_PORT
ENV NODE_APP_SERVER_PORT $NODE_APP_SERVER_PORT
COPY . .
EXPOSE 8000
CMD ["npm", "start"]
```

Server Dockerfile



```
version: '3'

services:
  client:
    build:
      context: client/
      dockerfile: ./Dockerfile
    image: "mattweir2/cloud-assignment:client"
    ports:
      - "3000:3000"
  server:
    build:
      context: server/
      dockerfile: ./Dockerfile
    image: "mattweir2/cloud-assignment:server"
    ports:
      - "8000:8000"
```

Application Docker Compose File

Test plan

Task	Expected Outcome	Result	Screenshot (Appendix)
Search Artist	Return array of buttons of artist names	PASS	1
Search Artist that doesn't exist	Show error message, Application continues	PASS	2
Search Artist but can't connect to back-end server	Show error message, application continues	PASS	3
Click on artist	Move application to next stage, Show past events	PASS	4
Map markers show description	Click on map markers, description shows	PASS	5
Clicking on past event	Shows imbedded YouTube videos	PASS	6
YouTube API Limit reached	Shows error message, application continues	PASS	-
Click on Artist but can't connect to back end	Shows error message, application continues	PASS	7
Click on past gig but can't connect to back end	Shows error message and application continues	PASS	8
Click on artist	Move application to next stage, Shows map with markers	PASS	9

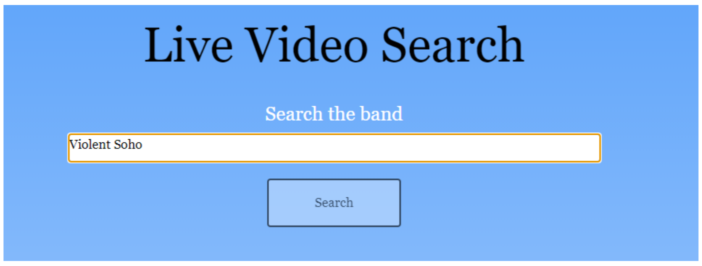
Difficulties / Exclusions / unresolved & persistent errors

This application was a big learning curve for me, when it came to using docker and AWS, as never working with the technologies before.

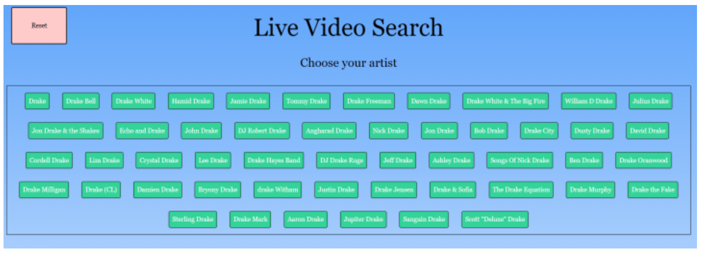
My major roadblock was trying to combine the two containers into one, I was able to follow the guide on how to serve react from express however I could not get docker to combine them both into the same container, every attempt I made the server was the only file within the container. I settled for just having a separate container for each of them and it worked fine.

User guide

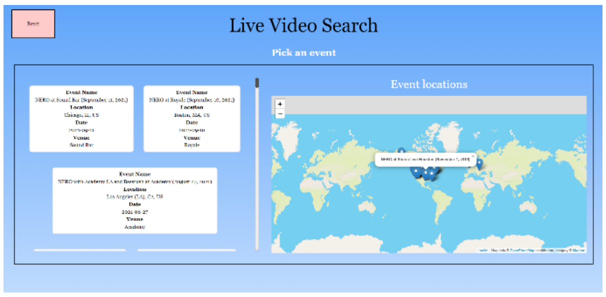
The application is straight forward and easy to use, the user begins by entering a query into the search bar on the main page.



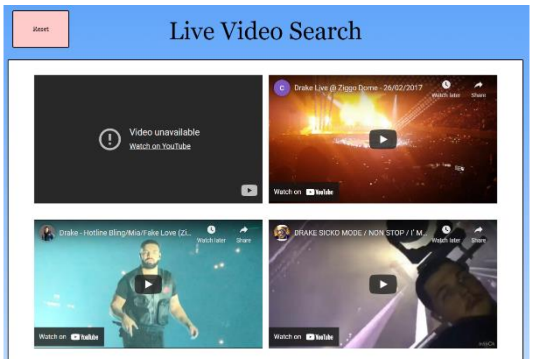
After the user continues, the are met with a list of artists names that they can pick from.



Upon clicking on their desired artist,

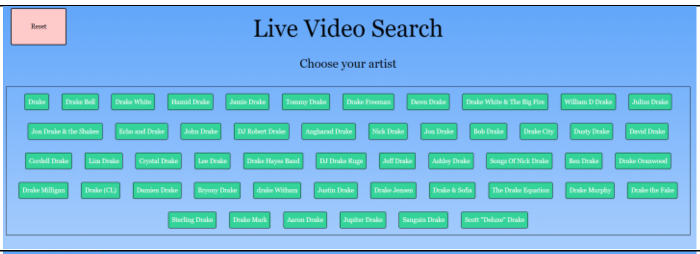
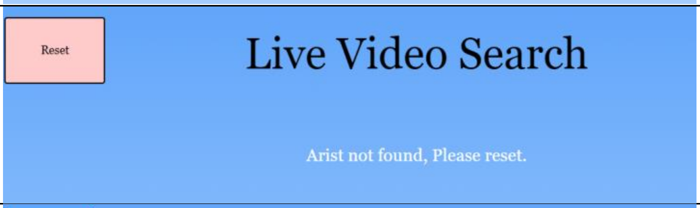
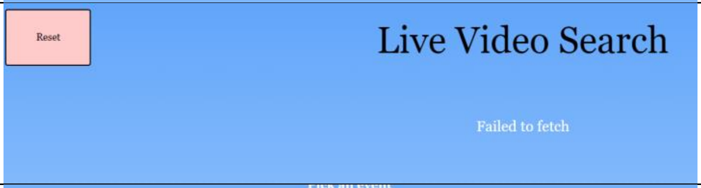
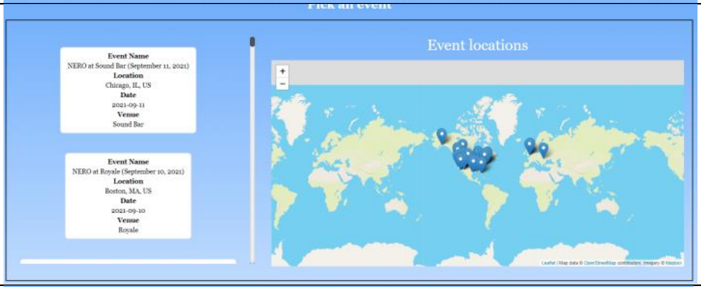
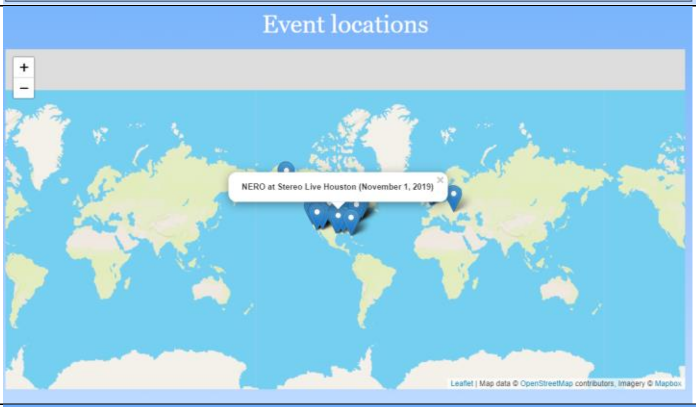
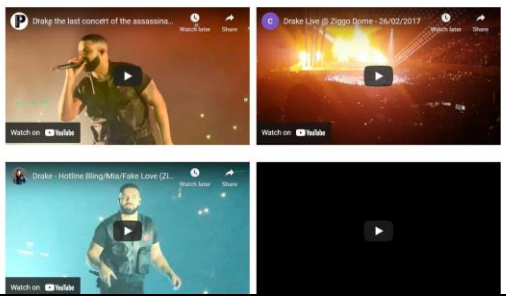


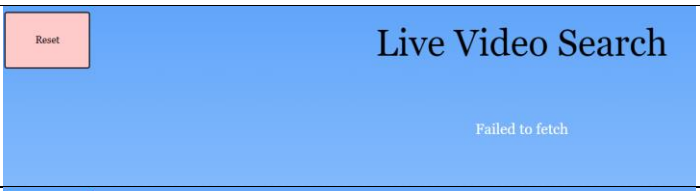
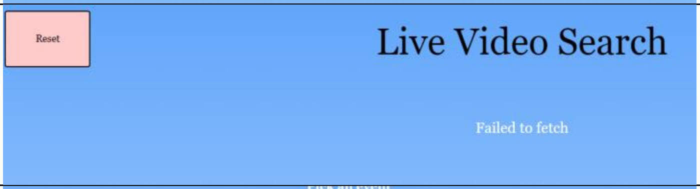
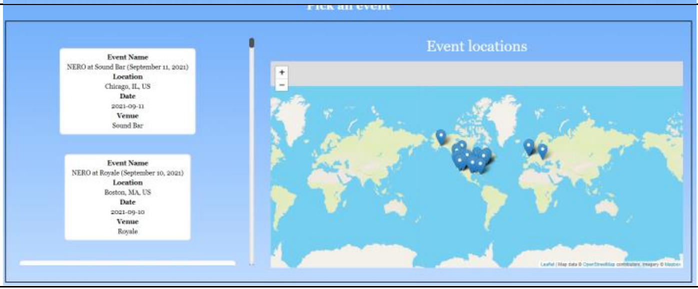
They can click on markers and receive a description of that event or click on an event to see videos from that event,


Reset
Live Video Search
Video unavailable
Watch on YouTube
Drake Live @ Ziggo Dome - 26/02/2017
Drake - Hotline Bling/Mia/Fake Love (Zi...
DRAKE SICKO MODE / NON STOP / I'M...
Watch later
Share
Watch on YouTube

If at any time the user changes, there mind on a query or an error occurs they can use the red reset button to return to the main page and reset everything.

Appendices

ID	Screenshot
1	 Reset Live Video Search Choose your artist Drake · Drake Bell · Drake White · Hamid Drake · Jamie Drake · Tommy Drake · Drake Freeman · Dawn Drake · Drake White & The Big Fire · William D Drake · Julius Drake · Jon Drake & the Shakes · Echo and Drake · John Drake · DJ Robert Drake · Angharad Drake · Nick Drake · Jon Drake · Bob Drake · Drake City · Dusty Drake · David Drake · Cordell Drake · Liza Drake · Crystal Drake · Lee Drake · Drake Hayes Band · DJ Drake Ruge · Jeff Drake · Ashley Drake · Songs Of Nick Drake · Ben Drake · Drake Oranwood · Drake Milligan · Drake (CL) · Damien Drake · Bryony Drake · drake Witham · Justin Drake · Drake Jensen · Drake & Sofia · The Drake Equation · Drake Murphy · Drake the Fake · Sterling Drake · Drake Mark · Aaron Drake · Jupiter Drake · Sanguin Drake · Scott "Deluxe" Drake
2	 Reset Live Video Search Arist not found, Please reset.
3	 Reset Live Video Search Failed to fetch
4	 **Event Name** NERO at Sound Bar (September 11, 2021) **Location** Chicago, IL, US **Date** 2021-09-11 **Venue** Sound Bar **Event Name** NERO at Royale (September 10, 2021) **Location** Boston, MA, US **Date** 2021-09-10 **Venue** Royale
5	 NERO at Stereo Live Houston (November 1, 2019) Leaflet | Map data © OpenStreetMap contributors, Imagery © Mapbox
6	 Drake the last concert of the assassina... Drake Live @ Ziggo Dome - 26/02/2017 Drake - Hotline Bling/Mia/Fake Love (Zl... Watch later Share Watch on YouTube

7	 Reset Live Video Search Failed to fetch
8	 Reset Live Video Search Failed to fetch
9	 Pick an event **Event Name** NERO at Sound Bar (September 11, 2021) **Location** Chicago, IL, US **Date** 2021-09-11 **Venue** Sound Bar **Event Name** NERO at Royale (September 10, 2021) **Location** Boston, MA, US **Date** 2021-09-10 **Venue** Royale Event locations Leaflet | Map data © OpenStreetMap contributors, Imagery © Mapbox
10	