

lyriqs.io

CAB432 Assignment 1

Matthias Eder


11378093

17 September 2022

## Contents

Introduction.....	2
Mashup Purpose & description.....	2
Services used .....	3
Musixmatch API (v.1.1) .....	3
text-processing.com API (v.1.0).....	3
Quickchart Word Cloud API (v.1.4.3).....	3
AWS S3 .....	4
Mashup Use Cases and Services.....	4
UC1: Carpool karaoke .....	4
UC2: Compilation of a radio show playlist.....	5
UC3: Lyrics critic .....	6
Technical breakdown .....	7
Architecture and Data Flow.....	7
Deployment and the Use of Docker .....	9
Test plan .....	10
Difficulties / Exclusions / Reflexion.....	12
Extensions.....	12
User guide .....	13
Analysis.....	13
Question 1: Vendor Lock-in .....	13
Question 2: Container Deployment .....	14
Appendices .....	16
Appendix A.....	16
Appendix B .....	18

### Mashup Purpose & description


lyriqs.io

Counter: 58

Song search

why does it always rain on me

×

Search

Title	Artist	Album
Why Does It Always Rain On Me	Mariachi Rock-o	Sonidos de Jalisco, Vol. 3
Why Does It Always Rain On Me	Paul Young	Rock Swings
Why Does It Always Rain on Me	Alexander Klaws	Here I Am
<b>Why Does It Always Rain On Me?</b>	<b>Travis</b>	<b>The Man Who</b>
Why Does It Always Rain On Me? - Live At Glastonbury Festival / 1999	Travis	Live At Glastonbury '99
Why Does It Always Rain On Me (Karaoke Version) (Originally Performed By Travis)	Karaoke Diamonds	The Best for Rock Musicians, Vol. 17 (Karaoke Version) [Sing the Songs of Your Stars]
Why Does It Always Rain On Me?	2 Play	Sax At Midnight
Why does it always rain on me?	Janet Gabriel	The Fire Within
Why Does It Always Rain on Me? (live)	Travis	Driftwood

Go ahead and choose a song from the table. This will fetch the lyrics and analyse their content for you.

Why Does It Always Rain On Me?

Travis

I can't sleep tonight  
Everybody's saying everything is alright  
Still I can't close my eyes  
I'm seeing a tunnel at the end of all these lights

Sunny days  
Where have you gone?  
I get the strangest feeling you belong

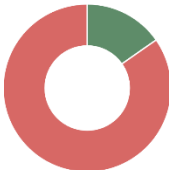
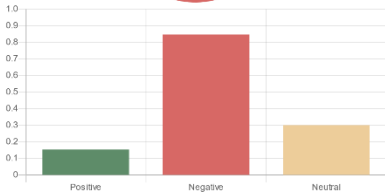
Why Does It Always Rain on Me?  
Is it because I lied when I was seventeen?  
Why Does It Always Rain on Me?  
Even when the sun is shining  
I can't avoid the lightning

I can't stand myself  
I'm being held up by invisible men  
Still life on a shelf when  
I got my mind on something else

Sentiment Charts

Positive

Negative

Positive and negative sentiments will add up to 1, while neutral is standalone. If neutral is greater than 0.5 then the label will be neutral. Otherwise, the label will be positive or negative, whichever has the greater probability.

Lyrics Word Cloud




Figure 1: Screenshot of song search and display of lyrics, sentiment analysis results, and word cloud



## Services used

### *Musixmatch API (v.1.1)*

API to search and retrieve song information and lyrics. Authentication happens via an API key retrieved after setting up an account and 2k calls per day are free. Only 30% of the lyrics are accessible with a free plan, but for the sake of the assignment this limitation was deemed acceptable (there were no objects from the tutor after sending the proposal). The API works fine and really just needs the API key to work. The information provided per song is extensive. I would have liked to additionally to the sentiment analysis display Musixmatch's own mood analysis via the endpoint `track.lyrics.mood.get`, but the endpoint was only accessible with a paid plan. The mood would have been divided up into a list of fitting keywords which I would have liked to display alongside the sentiment charts.

Endpoint: GET to `http://api.musixmatch.com/ws/1.1/track.search?`

Returns: A JSON object containing a list of up to 10 fitting tracks with regard to the user input search term.

Docs: <https://developer.musixmatch.com/documentation/api-reference/track-search>

Endpoint: GET to `http://api.musixmatch.com/ws/1.1/track.lyrics.get?`

Returns: A JSON object containing the lyrics content and additional metadata.

Docs: <https://developer.musixmatch.com/documentation/api-reference/track-lyrics-get>

### *text-processing.com API (v.1.0)*

Simple JSON over HTTP web service for text mining and natural language processing similar to the natural language toolkit (NLTK) known from Python. It is currently free and open for public use without authentication, but limited to 1k calls per IP per day. The API is used to analyse the sentiment of song lyrics and return the positive/negative/neutral shares of the lyrics content. The documentation is limited for this API. It took me some time to figure out how exactly the lyrics content should be passed to the endpoint as the value of the `text` parameter in the body is expected to be form-encoded. In the end, it was necessary to set the headers `Content-Type` to `multipart/form-data` and `Content-Length` to the length of the supplied lyrics string.

Endpoint: POST to `http://text-processing.com/api/sentiment/`

Returns: A JSON object containing the confidence values for the sentiments positive/negative/neutral and the predominant sentiment label.

Docs: <http://text-processing.com/docs/sentiment.html>

### *Quickchart Word Cloud API (v.1.4.3)*

Open-source web service creating charts from Chart.js on a backend and returning them as an image. It also offers an endpoint to create a wordcloud which is used for the mashup. The lyrics are supplied as a parameter to this call and the result is a full HTML `<svg>` tag with the word cloud as an SVG image ready to be displayed on a web page. It is very simple to use and

the wordcloud is cutomizable to a great extent. The API is limited to 1 chart per second and 10k charts per month per user for a free plan which is more than enough. No authentication is necessary.

Endpoint: GET to <https://quickchart.io/wordcloud/>?

Returns: The SVG image of the generated word cloud inside a HTML `<svg>` tag as text.

Docs: <https://quickchart.io/documentation/word-cloud-api/>

### AWS S3

A bucket on AWS S3 contains a JSON file *page\_counter.json* with a counter entry that is updated on each visit to the site. Each reload of the client sends a request to the server on the endpoint */counter*. If the bucket does not exist yet it is created using the provided AWS credentials. If the counter object does not exist on startup it is also created with the first request and initialized with the counter set to 0. The default name used for the page counter bucket is *cab432-n11378093-lyriqs*.

Docs: <https://aws.amazon.com/s3/>

## Mashup Use Cases and Services

### UC1: Carpool karaoke

As a	music enthusiast
I want	to find the lyrics of my favourite song
So that	my friends and I can sing along to it in the car.

The user interface provides a text input field [1] where a music enthusiast can enter an arbitrary search term (e.g. "american pie"). Pressing the "Search" button [2] next to the text input triggers the song search. The input is sent as a query parameter from the client to the endpoint */songs/search?song=american pie* on the server which then sends a request to the */track.search* endpoint of the Musixmatch API. This returns a list of up to 10 search results which are sent back to the client and displayed in a table [3]. The user can then choose the matching song by clicking on the respective entry in the table [4]. The Musixmatch *track\_id* of the chosen song is then used as an input to the mashup endpoint */songs/lyrics?id=12345* on the server. The server takes this ID and uses it to send a request to the */track.lyrics.get* endpoint from Musixmatch which retrieves the lyrics. The content of the lyrics is then cleaned on the server-side and returned to the client along with the other mashup data which are not relevant for this use case. The (preview of the) lyrics are then displayed next to the table [5]. These lyrics can then be sung out loud in any car. Please refer to Figure 2 to follow the process described here.

lyriqs.io

Counter: 58

Song search:  
american pie

×

Search

Title	Artist	Album
American Pie	Don McLean	Don McLean: Live In Manchester
American Pie	Madonna	American Pie - Single
American Pie	Os Barões da Pisadinha	As Melhores 2018
American Pie	The Deep Dixie Boys	Only a Country Heart
American Pie	Cast of Zoey's Extraordinary Playlist	Zoey's Extraordinary Playlist: Season 1, Episode 12 (Music From the Original TV Series) - EP
<b>American Pie (Full Length Version)</b>	<b>Don McLean</b>	<b>American Pie</b>
The Saga Begins (Lyrical Adaption of "American Pie")	"Weird Al" Yankovic	Running With Scissors
American Pie (Single Version)	Don McLean	Legendary Songs Of Don McLean
Bye Bye Miss American Pie	Billy Cassidy	Bye Bye Miss American Pie
Be Like That (American Pie Edit)	3 Doors Down	American Pie 2

Go ahead and choose a song from the table. This will fetch the lyrics and analyse their content for you.

**American Pie (Full Length Version)**
Negative sentiment

Don McLean

A long, long time ago  
I can still remember how that music  
Used to make me smile  
And I knew if I had my chance  
That I could make those people dance  
And maybe they'd be happy for a while  
  
But February made me shiver  
With every paper I'd deliver  
Bad news on the doorstep  
I couldn't take one more step  
I can't remember if I cried  
When I read about his widowed bride  
Something touched me deep inside  
The day the music died  
  
So, bye-bye, Miss American Pie  
Drove my Chevy to the levee, but the levee was dry  
And them good ol' boys were drinkin' whiskey and rye  
Singin', 'This'll be the day that I die  
This'll be the day that I die'  
  
Did you write the book of love  
And do you have faith in God above  
If the Bible tells you so?  
Now, do you believe in rock 'n' roll  
Can music save your mortal soul  
And can you teach me how to dance real slow?

Figure 2: Screenshot for UC1

#### UC2: Compilation of a radio show playlist

As a	radio DJ
I want	to find out the mood of a particular song
So that	I can put together a playlist of songs that go well together for my radio show.

The same process as for UC1 is followed here. The radio DJ can search for a song like "recovery" which retrieves a list of songs the DJ can choose from. They can then choose one of them (e.g. "Recovery" by Frank Turner) which again triggers a request to the server on endpoint `/songs/lyrics?id=45678` corresponding to the track ID of the chosen song. The cleaned lyrics retrieved from Musixmatch are additionally used as input to the sentiment analysis of the `text-processing.com` API and the endpoint `/sentiment`. The lyrics are supplied as form-encoded data in the body of the POST request. The resulting confidence levels for positive/negative/neutral sentiments and the predominant label are returned as part of the mashup response to the client. The label is displayed next to the song title [1] and the mood values are shown in two charts: a donut chart displaying positive/negative shares [2] and a bar chart additionally displaying the neutral share [3]. Also, a short info text is shown to explain the values [4]. The DJ can then decide if the song fits the rest of the playlist. Please refer to Figure 3 to follow the process described here.

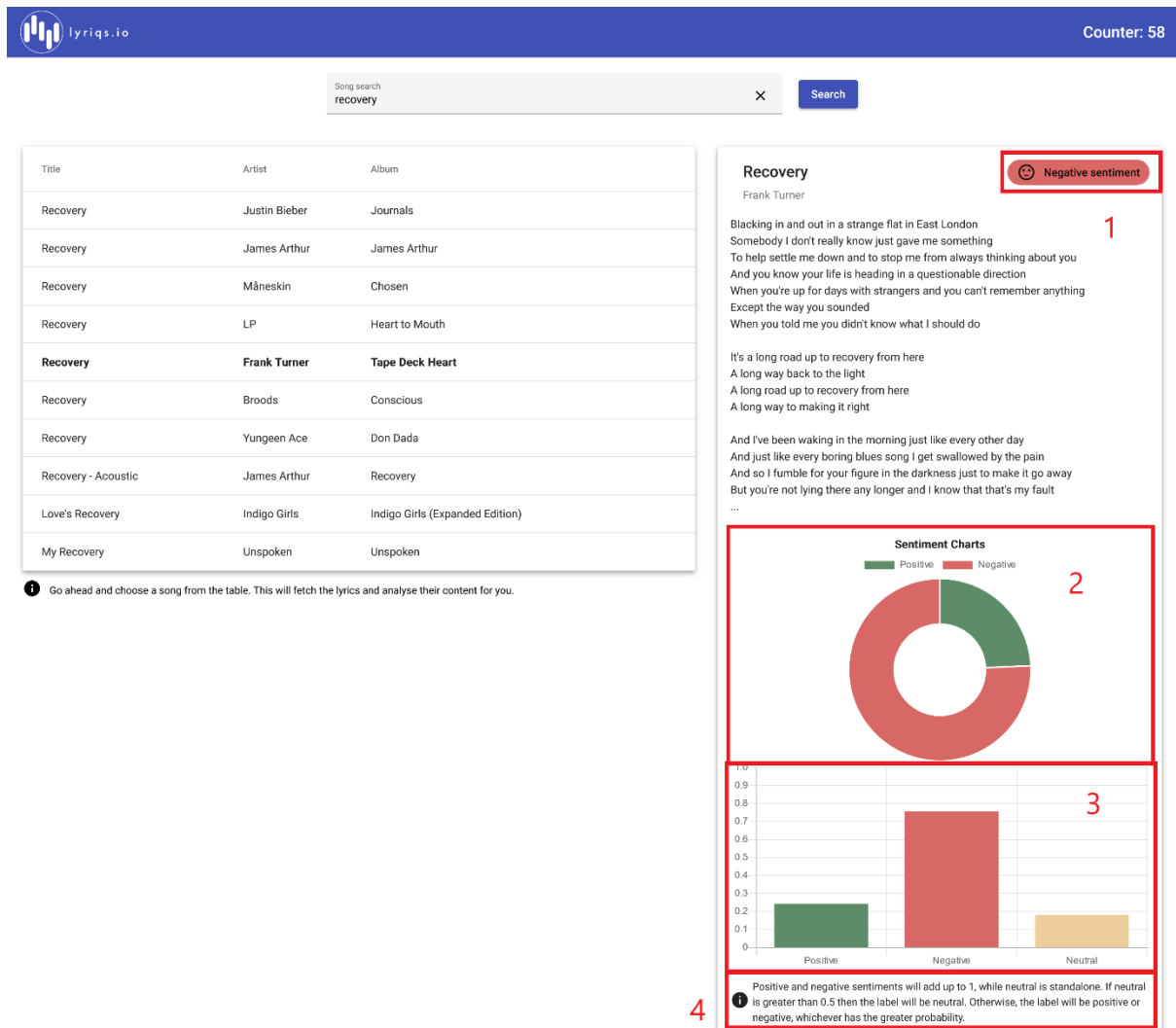


Figure 3: Screenshot for UC2

### UC3: Lyrics critic

As a	critic of musical lyrics
I want	to find out the most common words in the lyrics of a particular song
So that	I can assess the quality of the language used in the song.

The final part of the mashup response (apart from the cleaned lyrics content covered in UC1, the sentiment analysis results covered in UC2) is the word cloud image. The process again follows the same flow: the music critic can input a search term like "seventeen going under", choose a particular entry from the table of search results (e.g. "Seventeen Going Under" by Sam Fender). This then sends out a request to the mashup endpoint `/songs/lyrics?id=13579` which gets the lyrics from Musixmatch, uses them as input to the sentiment analysis from `text-processing.com` and finally sends a request with the lyrics to endpoint `/wordcloud` of the Quickchart Word Cloud API. The resulting `<svg>` HTML tag containing the vector graphic. This is added to the mashup response and returned to the client where it is displayed below the sentiment charts [1]. The music critic can then easily make out the most frequent words used

in the lyrics and judge the quality of the language used in the song. Please refer to Figure 4 to follow the process described here.

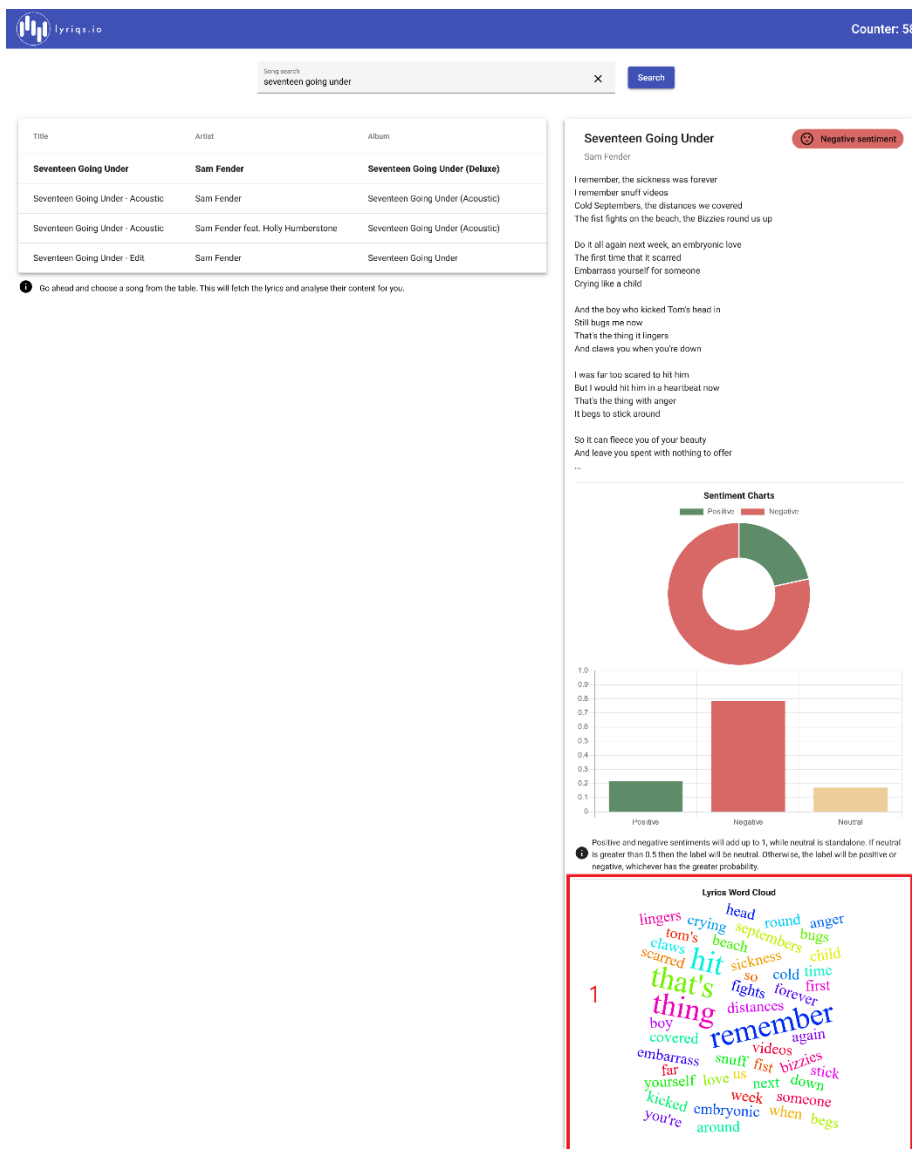


Figure 4: Screenshot for UC3

## Technical breakdown

## Architecture and Data Flow

Explain how your system operates, making it clear how data flows around the system through requests and responses. You are describing the overall architecture of your application at a source code level. The description above tells us something of the application's use. Now we want to see how that maps to the code organization – show us how your code is organized and tell us how you have split the responsibilities. We should get some sense of how the application works and how the data and control flows around. You may also find it helpful to show us screen grabs of code if that makes your points clearer. Tell us anything you think we need to know about how you have structured the application and made it work, but there also a section below to describe problems.



Please note that the architecture description is largely taken from the README files part of the GitHub repository for this project.

I decided to follow a more real-life approach and divided frontend (*lyriqs-client*) and backend (*lyriqs-server*) into separate sub-projects which can be dockerized independently. This also accounts for separation of concerns, encapsulation, abstraction and several other principles of software architecture. I chose Angular as a frontend framework as I already had experience with this tech stack and wanted to take the opportunity of the assignment to freshen up my knowledge with respect to the dockerization of a SPA. Dockerization happens in principle with *docker compose* to minimize the effort required to run the project from scratch.

### lyriqs-client

The client-side project is an Angular single-page web application served on a NGINX web server in production mode responsible for proxying requests to the server. The client provides a user interface to search for songs, displays the results and allows users to choose a song. The UI displays (a preview of) the content, a mood label as well as the result of the underlying sentiment analysis, and a wordcloud of the lyrics. The client is a SPA created with the frontend framework Angular version 14.1.3. The client is responsible for providing an entrypoint for users, process input and allow interaction with results. The user input is sent to the corresponding endpoints of the server API. Each reload of the application will send a GET request to increment and return the page counter value at endpoint */counter*. All other API requests are triggered by user action.

### lyriqs-server

The server provides an API to search for songs, fetch their lyrics and perform an analysis of the lyrics content (sentiment, word cloud). The server is written using [TypeScript](#) and consists of a basic [Express](#) backend served on a [Node.js](#) server. The API of the server exposes three endpoints: */counter*, */songs/search*, and */songs/lyrics*.

#### 1. */counter*

A GET call to this endpoint increments and returns the new value of the number of page views with regard to a page counter hosted on AWS S3. The raw number of the page views is returned as text.

#### 2. */songs/search*

A GET call to this endpoint returns a list of search results for a specific search term supplied as a query parameter *song*. A full request would look like */songs/search?song=Yesterday*. The server sends a request to the song search endpoint of the Musixmatch API to retrieve the results and then returns the track list as a JSON. The list contains objects of type track from Musixmatch (details can be found at the [API documentation](#)).

#### 3. */songs/lyrics*

This is the actual mashup endpoint. A GET call with query parameter *id*, which contains the *track\_id* of a Musixmatch song entry, will perform several API calls from the server and compile

a mashup response object that is returned once all internal calls have concluded. First, the lyrics for the track ID are fetched from Musixmatch. The lyrics are then cleaned and used as the input to calls to the text-processing.com and Quickchart Word Cloud APIs which perform the sentiment analysis and wordcloud generation, respectively. The mashup response is an object of type *models/lyrics.interface.ts*. A full request to the mashup endpoint would look like */songs/lyrics?id=15953433*.

The server route code of the mashup endpoint is shown in Appendix A 1 for reference. Log statements have been removed for clarity.

### Deployment and the Use of Docker

The application can be dockerized using *docker compose up* from the project root. The *docker-compose.yml* file is shown in Appendix A 2. This creates the images and containers for the server and client, connects and starts them. It is necessary to set AWS and API keys as environment variables and source them from the same shell as the docker compose command is executed before actually running the project. The following variables need to be set:

- MUSIXMATCH\_API\_KEY: API key for the Musixmatch service
- AWS\_ACCESS\_KEY\_ID: key ID for AWS
- AWS\_SECRET\_ACCESS\_KEY: access key for AWS
- AWS\_SESSION\_TOKEN: session token for AWS

If the images have already been created and you want to run the containers, it is recommended to execute the following commands to facilitate communication between frontend and backend, as well as set environment variables explicitly:

```
docker network create net
docker run --network net --name server
    -p 3000:3000
    -e MUSIXMATCH_API_KEY="XYZ"
    -e AWS_ACCESS_KEY_ID="XYZ"
    -e AWS_SECRET_ACCESS_KEY="XYZ"
    -e AWS_SESSION_TOKEN="XYZ"
    -t your-registry/lyriqs-image-name:server
docker run --network net --name client
    -p 4200:80
    -t your-registry/lyriqs-image-name:client
```

Docker compose executes the Dockerfiles for both subprojects *lyriqs-client* and *lyriqs-server*. The Dockerfile for the client is shown in Appendix A 3. The Dockerfile of the client uses a multi-stage build. The first stage builds the Angular application and copies the artifacts. The second stage serves the app on a lightweight [NGINX](#) web server. This server also acts as a reverse proxy to redirect requests between client and server in any environment. This setup follows best-practise guidelines for serving Angular applications in a production environment and reduces the size of the Docker image heavily to around 25 MB (serving the app on the development server would have resulted in >2 GB). The configuration of the server was a major challenge. The Dockerfile of the server is trivial and therefore not shown in the Appendix.

Please note that the images are distinguished only by tag server or client. This is due to a limitation on DockerHub allowing only a single private repository. It was necessary to keep the project private for the submission of the assignment. Otherwise the two images would have been pushed to separate repositories.

## Test plan

Task	Expected Outcome	Result	Screenshot (Appendix B)
Search for song	loading bar shows during request; table with search results is shown	PASS	1
Select a song from the result table	selected song is highlighted in the table; loading bar shows during request; lyrics + sentiment charts + word cloud are shown	PASS	2
Clear search input	search input field is empty again; clear button not visible	PASS	3
Select different song from same result table	only loading bar shows during request; lyrics + sentiment charts + word cloud change once request returns	PASS	4, 5
Search for new song with result table already present	loading bar and logo graphic show during request; table changes once request returns; lyrics + sentiment charts + word cloud are not shown as selection is cleared (see task 1)	PASS	6
Display exact sentiment values by hovering over chart parts (Donut chart)	exact confidence values are shown for both positive and negative shares	PASS	hovering behaviour not visible in screenshots
Display exact sentiment values by hovering over chart parts (Bar chart)	exact confidence values are shown for positive, negative, and neutral values	PASS	hovering behaviour not visible in screenshots
Empty input disables search button	search button cannot be clicked and search cannot be triggered	PASS	3
Searching for nonsense term	request without search results shows large logo graphic and info text that no search results are available	PASS	7
Page counter	page counter value is visible	PASS	8
Page counter update	after a reload the page counter is incremented	PASS	9
Initial state of page	logo graphic and info text that no search results are available are shown; focus is in search input	PASS	8, 9

Please refer to the images provided in Appendix B to confirm the passing of the test cases. Some are hard to prove with a simple screenshot. I will be happy to show that these work in the demo.

Apart from the general struggles with CSS ("how do I center a div again?"), there were no real issues during the development of the application regarding programming/scripting. The real issues occurred when I tried to connect the Angular app with the Express server with docker compose. The dockerization of the server was very straightforward and worked from scratch. I then read into the dockerization of an Angular application which is not as easy. The development server created with the *ng serve* directive is very bulky and should not be used in a production environment. Much rather, the lightweight web server NGINX should be used to serve a single-page application. In principal, this can easily be achieved in a two-stage Docker build where the second stage takes the build artifacts of the first stage and serves them on the NGINX server. However, this requires the server being set up as a reverse proxy so that the requests between frontend and backend are sent to the correct address. The configuration of this proxy turned out to be difficult.

I chose a tech stack that is largely out of scope of the general specification of assignment 1 and therefore knew that I could not count on guidance from the tutors for this problem and was on my own. I easily spent just as much time with the dockerization, web server proxy set up and configuration and communication linking between the containers as with the actual development of the project. In the end, I was able to figure out how to configure NGINX to serve the SPA and redirect the requests to the server endpoints correctly. I definitely feel like I learned a lot, especially about Docker and the advantages and potential pitfalls of containerization.

The only differences between my proposal and final project was the chart API. I decided to use Chart.js instead of D3.js as I felt more comfortable with it. I also spent a few hours on designing and generating the logo graphics for the project and styling of UI components, which definitely was not necessary for the assignment (but it was fun).

I would also like to improve the error handling in further steps. Mainly, I would like to display the errors as Snackbars from Material Design instead of showing the same graphic with the information reading that there are no search results available. This would greatly improve initial debugging and the user experience in general. I had no issues with the API usages apart from the necessary headers and encoding of the body of the POST request to the *text-processing.com* NLP API. The rest worked as expected.

### Extensions

I would have liked to additionally to the sentiment analysis display Musixmatch's own mood analysis via the endpoint *track.lyrics.mood.get*, but the endpoint was only accessible with a paid plan. The mood would have been divided up into a list of fitting keywords which I would have liked to display alongside the sentiment charts.

Additionally, the limitation of the Musixmatch API to display only 30% of the lyrics on a free plan was OK for an assignment showcasing an API mashup, but it would not be viable for a production-ready application. I would have to either change to a paid plan or choose a different API to search for songs and retrieve the lyrics. Genius.com has a decent API but required OAuth which is why I did not choose it for the assignment. It also provides song



interpretations which could also extend the analysis component of *lyriqs.io*. However, most of these analysis features are hidden behind a paywall anyway, so investing to a certain degree seems unavoidable.

A redesign of the UI might also be a good step with the addition of new features. Using a translation API for the lyrics might be interesting. Additional natural language processing steps could also be worth adding.

### User guide

As already explained, the UI offers a single search input field which can be used to search for songs. Enter a song name and press "Enter" or the "Search" button next to the input. You can then choose a song from the search result table which will load its lyrics, display the sentiment analysis label and charts, and the word cloud. You can also see the page counter in the top right corner in the toolbar. Hover over the sentiment charts to examine the confidence values.

For visual guidance, please refer to Figure 2 where the UI components are explained. For a guide on how to run the Docker containers, please refer to the README file(s) in the repository or section "Architecture and Data Flow".

### Analysis

#### Question 1: Vendor Lock-in

*Looking at your mashup as it stands, how dependent are you on the people who provide the services that you use? In a commercial context, the APIs, the data and the cloud services all matter to you. How hard would it be to change?*

- *How hard would it be to replace the APIs that you are working with? Could you easily replace them if they were shut down suddenly or their terms of service changed? Consider each of them in turn and explore the domain of the API and tell us about obvious competitors or their absence.*

(4 marks)

Every API in use for *lyriqs.io* is easily replaceable – to a certain degree. The mashup can be divided into several tasks which can be carried out by numerous services: song search, lyrics retrieval, sentiment analysis, and word cloud generation. From a technological point of view, the application does not heavily rely on a single service, as the calls have been modularized, responses have been abstracted and the input used to the various requests is kept to a minimum (only inputs from APIs used are the song ID and the lyrics).

The only entangling issue that currently exists is that the tasks song search and lyrics retrieval are carried out by the same API. The proprietary ID from Musixmatch for the chosen song obtained during the song search is used as the input to the lyrics retrieval from an endpoint of the same API. These services could be still be replaced, but it would require adaptations to the expected input of the lyrics retrieval. For instance, the [Genius API](#) could be used for both tasks, but the internal IDs used are different of course. Also, the API is not as simple to use as it would require OAuth2. This technical limitation means that it almost seems necessary to handle both the song search and lyrics retrieval task with the same API. It is necessary to use a song ID of some sort as the input for the lyrics retrieval as a song title as input would by no

means be distinct enough. Other APIs like [powerlyrics](#) are easy to use but do not fit the current setup of the application where the song search and lyrics display are separated.

The major issue for a change in service would be the willingness to pay for an API. This already poses an issue for the current state of the application as the Musixmatch API has limitations with regard to the results obtainable from various endpoints (e.g. only 10 tracks for a song search, only 30% of the lyrics, some endpoints are disabled completely). There are good alternatives for the sentiment analysis like [Twinword](#), [AssemblyAI](#) or [Google NLP API](#) but all of these require at least a paid plan setup. The same can be said for the word cloud API. Both the word cloud generation and the sentiment analysis only require the lyrics as input. The origin of the lyrics does not matter proving that there is no dependency between these tasks.

- *How hard would it be to change the persistence service you have used to one supplied by another vendor? Identify equivalent alternatives and discuss briefly how that might affect your approach.*

(2 marks)

The architecture is currently semi-dependent on AWS S3 as a persistence service. The client is completely abstracted and only expects to retrieve the number of page views as text so there are no changes necessary if it were to be replaced. However, the server side is set up specifically to authenticate to AWS, create or access a bucket and set and retrieve the number from it. A significant amount of code would therefore need to be changed. In general, it would not be a problem. Viable alternatives would be a simple MySQL, MongoDB or DynamoDB server or a different kind of database in form of a Docker container. My database of choice would be a MongoDB instance which would turn the architecture into a full-on MEAN stack (MongoDB + Express + Angular + Node). This is an industry-proven standard and easy to use. The actual change would require some changes in the dockerization and the server code, but the client would be unaffected.

#### Question 2: Container Deployment

*Scenario: more substantial application with external services, user management, security, persistence services, etc. What are the advantages and disadvantages of the container-based deployment?*

- *Assuming that we have access to a service to manage container deployment and communication, are there any disadvantages to a container-based deployment for this application? Would we consider deployment of the application directly to a virtual machine i.e. one instance of the application for each instance of the EC2 or Azure Linux VM?*

(2 marks)

I personally do not see any striking disadvantages of a container-based deployment for the application presented here.

A major advantage is the abstraction of services and encapsulation. It is possible to create a specific container for every process in an application which allows for easy exchange of services, rapid deployment and simple interoperability. Due to this separation, security is also

increased as the containers run in isolation and do not interfere with each other. Most processes are also easy to dockerize. The Dockerfile of an Express app, for instance, is only a few lines long and does not require special configuration in most cases. This holds also for scale as a much larger server app would still not require more adaptations. Additionally, Docker containers can be integrated into a continuous integration (CI) pipeline. Deployment is easy with the tagging function. Consistency allows for the container to work on any system regardless of the configuration which is also important for large-scale applications.

However, not *all* processes are easy to dockerize. One example is the serving of an Angular app. While the configuration of the NGINX reverse proxy is not related to Docker, the multi-stage build still was not easy to achieve. I imagine this being the case for several different architectures. Also, backup and recovery in case of breakdown might become an issue for a larger application with some sort of user management service and additional persistence layers. The major downside of Docker that I have experienced myself is the paradigm-shift its adoption entails and the learning curve associated with it.

- *Drawing upon the discussions of cloud architectures in the early lectures and the material on persistence from week 5 onwards, what persistence options would you consider if the application were to be deployed at scale using a collection of software containers? You may consider more than one level.*

*(2 marks)*

A caching layer using Redis could make sense here to a certain degree. The most frequently searched (i.e. most popular) songs and lyrics could be cached to allow for faster retrieval. Also, additional persistence services providing redundancy are important especially for a system handling user data. If the application were to be supplied at scale, some form of load balancing would be required to redirect requests to the server instances.

## Appendices

### Appendix A

```
router.get('/lyrics', (req, res) => {
  const id = req.query['id']!;
  const options = createLyricsOptions(id);
  const url = `http://${musixmatch.host}${options.path}`;

  axios.get(url)
    .then(rsp => {
      const lyricsRes = rsp.data.message.body.lyrics;
      const lyrics = cleanLyrics(lyricsRes.lyrics_body);
      const cleanedLyrics = cleanupLineBreaks(lyrics);

      const sentimentCall = axios(setupSentimentCall(cleanedLyrics));
      const wordcloudCall = axios(setupWordcloudCall(cleanedLyrics));

      axios.all([sentimentCall, wordcloudCall])
        .then(axios.spread((...responses) => {
          const sentimentRes = responses[0];
          const wordcloudRes = responses[1];
          const sentiment: Sentiment = {
            label: sentimentRes.data.label,
            negative: sentimentRes.data.probability.neg,
            neutral: sentimentRes.data.probability.neutral,
            positive: sentimentRes.data.probability.pos
          };
          const mashupResponse: Lyrics = {
            lyrics_id: lyricsRes.lyrics_id,
            track_id: +id,
            content: lyrics,
            language: lyricsRes.lyrics_language,
            sentiment: sentiment,
            wordcloud: wordcloudRes.data
          };
          res.json(mashupResponse);
        })))
        .catch(error => {
          ...
        });
    })
    .catch(error => {
      ...
    });
});
```

Appendix A 1: Server route for the mashup call at endpoint /songs/lyrics

```

version: '2.2'
services:
  server:
    build: ./lyriqs-server
    container_name: server
    image: meder96/lyriqs:server
    environment:
      - AWS_ACCESS_KEY_ID
      - AWS_SECRET_ACCESS_KEY
      - AWS_SESSION_TOKEN
      - MUSIXMATCH_API_KEY
    ports:
      - "3000:3000"
  client:
    build: ./lyriqs-client
    container_name: client
    image: meder96/lyriqs:client
    ports:
      - "4200:80"

```

*Appendix A 2: docker-compose.yml for the root project*

```

# Stage 1: Build
FROM node:16-alpine as build
WORKDIR /usr/src/app
COPY package.json package-lock.json ./
RUN npm install
COPY . .
RUN npm run build --prod

# Stage 2: Run
FROM nginx:stable-alpine
# Remove default nginx website
RUN rm -rf /usr/share/nginx/html/*
# Copy dist folder from build stage to nginx public folder
COPY --from=build /usr/src/app/dist/lyriqs-client /usr/share/nginx/html
# Copy nginx config file
COPY nginx.conf /etc/nginx/conf.d/default.conf
# Expose port
EXPOSE 80

```

*Appendix A 3: Dockerfile for lyriqs-client*



## Appendix B



lyriqs.io

Counter: 58

Song search  
seventeen going under

Search

Title	Artist	Album
Seventeen Going Under	Sam Fender	Seventeen Going Under (Deluxe)
Seventeen Going Under - Acoustic	Sam Fender	Seventeen Going Under (Acoustic)
Seventeen Going Under - Acoustic	Sam Fender feat. Holly Humberstone	Seventeen Going Under (Acoustic)
Seventeen Going Under - Edit	Sam Fender	Seventeen Going Under

1 Go ahead and choose a song from the table. This will fetch the lyrics and analyse their content for you.

### Appendix B 1: Screenshot for test task 1 – search for a song without selection



lyriqs.io

Counter: 58

Song search  
seventeen going under

Search

Title	Artist	Album
Seventeen Going Under	Sam Fender	Seventeen Going Under (Deluxe)
Seventeen Going Under - Acoustic	Sam Fender	Seventeen Going Under (Acoustic)
Seventeen Going Under - Acoustic	Sam Fender feat. Holly Humberstone	Seventeen Going Under (Acoustic)
Seventeen Going Under - Edit	Sam Fender	Seventeen Going Under

1 Go ahead and choose a song from the table. This will fetch the lyrics and analyse their content for you.

#### Seventeen Going Under

Sam Fender

Negative sentiment

I remember the sickness was forever  
I remember snuff videos  
Cold Septembers, the distances we covered  
The fist fights on the beach, the Bizzies round us up

Do it all again next week, an embryonic love  
The first time that it scared  
Embarrass yourself for someone  
Crying like a child

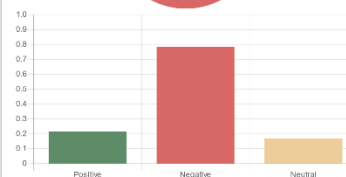
And the boy who kicked Tom's head in  
Still bugs me now  
That's the thing it lingers  
And claws you when you're down

I was for too scared to hit him  
But I would hit him in a heartbeat now  
That's the thing with anger  
It begs to stick around

So it can fleece you of your beauty  
And leave you spent with nothing to offer  
...

#### Sentiment Charts

Positive Negative



1 Positive and negative sentiments will add up to 1, while neutral is standalone. If neutral is greater than 0.5 then the label will be neutral. Otherwise, the label will be positive or negative, whichever has the greater probability.

#### Lyrics Word Cloud



### Appendix B 2: Screenshot for test task 2 – selection of song from table

lyriqs.io

Counter: 58

Song search

Search

Title	Artist	Album
Seventeen Going Under	Sam Fender	Seventeen Going Under (Deluxe)
Seventeen Going Under - Acoustic	Sam Fender	Seventeen Going Under (Acoustic)
Seventeen Going Under - Acoustic	Sam Fender feat. Holly Humberstone	Seventeen Going Under (Acoustic)
Seventeen Going Under - Edit	Sam Fender	Seventeen Going Under

Go ahead and choose a song from the table. This will fetch the lyrics and analyse their content for you.

### Seventeen Going Under

Sam Fender

Negative sentiment

I remember, the sickness was forever  
I remember snuff videos  
Cold Septembers, the distances we covered  
The fist fights on the beach, the Bizzies round us up

Do it all again next week, an embryonic love  
The first time that it scared  
Embarrass yourself for someone  
Crying like a child

And the boy who kicked Tom's head in  
Still bugs me now  
That's the thing it lingers  
And claws you when you're down

I was far too scared to hit him  
But I would hit him in a heartbeat now  
That's the thing with anger  
It begs to stick around

So it can fleece you of your beauty  
And leave you spent with nothing to offer

...

Sentiment Charts

Positive Negative

Appendix B 3: Screenshot for test task 3 and 8 – clearing search input and disabling search button with empty input

lyriqs.io

Counter: 58

Song search

why

X

Search

Title	Artist	Album
Why'd You Only Call Me When You're High?	Arctic Monkeys	AM
Why Would I Lack? (feat. SPMB Bills)	22Gz feat. SPMB Bills	Why Would I Lack? (feat. SPMB Bills) - Single
the remedy for a broken heart (why am i so in love)	XXXTENTACION	?
Back To You - From 13 Reasons Why - Season 2 Soundtrack	Selena Gomez	Back To You (From 13 Reasons Why - Season 2 Soundtrack)
Let Me Know (I Wonder Why Freestyle)	Juice WRLD	Let Me Know (I Wonder Why Freestyle)
Don't Know Why	Norah Jones	Come Away With Me (Remastered 2022)
unliss (Why Don't You Stay) [From KinnPorsche The Series]	Jeff Saur	unliss (Why Don't You Stay) [From KinnPorsche The Series]
That's Why (You Go Away)	Michael Learns To Rock	Played On Pepper
Tell Me Why (Re-Recorded / Remastered)	Berlin	Greatest Hits (Re-Recorded / Remastered)
Why Can't It Be Christmastime All Year?	Rosie Thomas	A Very Rosie Christmas! (Expanded Edition)

Go ahead and choose a song from the table. This will fetch the lyrics and analyse their content for you.

### Why'd You Only Call Me When You're High?

Arctic Monkeys

Negative sentiment

The mirror's image, it tells me it's home time  
But I'm not finished, 'cause you're not by my side  
And as I arrived I thought I saw you leavin', carryin' your shoes  
Decided that once again I was just dreamin' of bumpin' into you

Now it's three in the mornin' and I'm tryin' to change your mind  
Left you multiple missed calls and to my message, you reply  
"Why'd you only call me when you're high?"  
"Hi, why'd you only call me when you're high?"

Somewhere darker, talkin' the same shite

...

Sentiment Charts

Positive Negative

Positive and negative sentiments will add up to 1, while neutral is standalone. If neutral is greater than 0.5 then the label will be neutral. Otherwise, the label will be positive or negative, whichever has the greater probability.

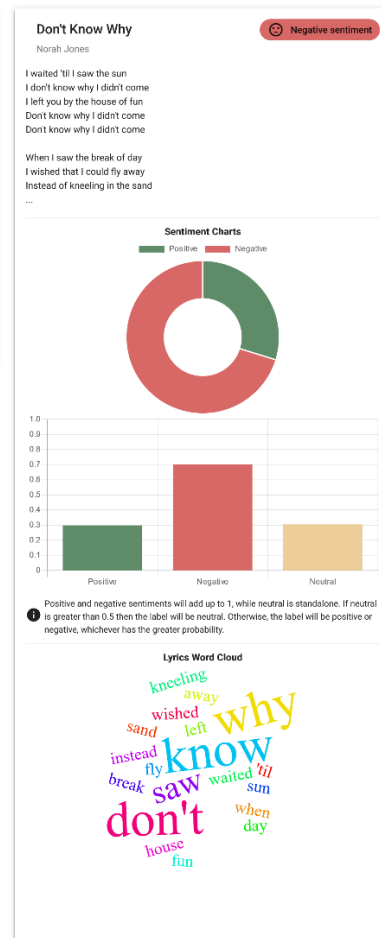
Lyrics Word Cloud

Appendix B 4: Screenshot for test task 4 – selection of song from result table

Song search why X Search

Title	Artist	Album
Why'd You Only Call Me When You're High?	Arctic Monkeys	AM
Why Would I Lack? (feat. SPMB Bills)	22Gz feat. SPMB Bills	Why Would I Lack? (feat. SPMB Bills) - Single
the remedy for a broken heart (why am i so in love)	XXXTENTACION	?
Back To You - From 13 Reasons Why - Season 2 Soundtrack	Selena Gomez	Back To You (From 13 Reasons Why - Season 2 Soundtrack)
Let Me Know (I Wonder Why Freestyle)	Juice WRLD	Let Me Know (I Wonder Why Freestyle)
<b>Don't Know Why</b>	<b>Norah Jones</b>	<b>Come Away With Me (Remastered 2022)</b>
unisa (Why Don't You Stay) [From KinnPorsche The Series]	Jeff Satur	unisa (Why Don't You Stay) [From KinnPorsche The Series]
That's Why (You Go Away)	Michael Learns To Rock	Played On Pepper
Tell Me Why (Re-Recorded / Remastered)	Berlin	Greatest Hits (Re-Recorded / Remastered)
Why Can't It Be Christmastime All Year?	Rosie Thomas	A Very Rosie Christmas! (Expanded Edition)

Go ahead and choose a song from the table. This will fetch the lyrics and analyse their content for you.



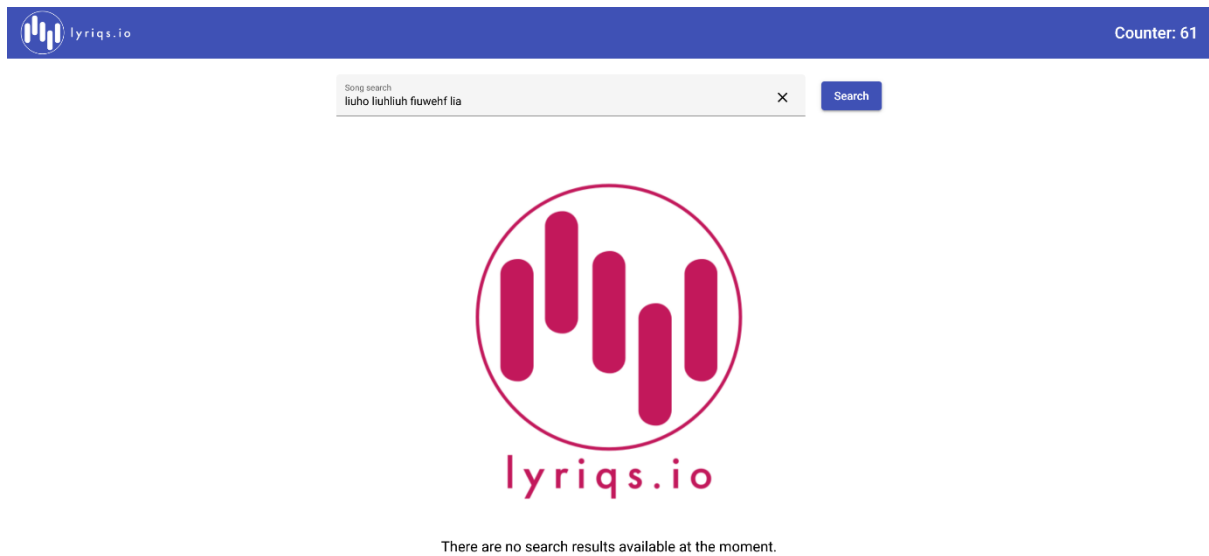
Appendix B 5: Screenshot for test task 4 – selection of different song from same result table

Song search

	Title	Artist	Album
	Peggy Sang the Blues	Frank Turner	England Keep My Bones
	Peggy Sang the Blues	Frank Turner & The Sleeping Souls	2012-09-30: Webster Hall, New York, NY, USA
	Peggy Sang The Blues - Acoustic	Frank Turner	England Keep My Bones (Deluxe Edition)
	Peggy Sang the Blues (Acoustic)	Frank Turner	England Keep My Bones (Deluxe Edition)

**i** Go ahead and choose a song from the table. This will fetch the lyrics and analyse their content for you.

Appendix B 6: Screenshot for test task 5 – search for new song after initial search



*Appendix B 7: Screenshot for test task 9 – search for nonsense term leading to empty result shows logo and info text*



*Appendix B 8: Screenshot for test task 10 – page counter visible in top right corner*



*Appendix B 9: Screenshot for test task 11 – update of page counter after reload*