2021

Peter Nguyen & Andrew Nguyen

N9698094 & N10408673

1/1/2021

Twitter Sentiment Radar

**

Replace image with one with some relevance to your application here

CAB432 Semester 2

Assignment 2: Cloud Project

*Contents*

[*Introduction 2*](#_Toc52357527)

[*Purpose & description 2*](#_Toc52357528)

[*Services used 2*](#_Toc52357529)

[*Twitter Standard Search API (v.1.1) 2*](#_Toc52357530)

[*API 2 2*](#_Toc52357531)

[*API n 2*](#_Toc52357532)

[*Use cases 2*](#_Toc52357533)

[*US 1 2*](#_Toc52357534)

[*US 2 2*](#_Toc52357535)

[*US n 3*](#_Toc52357536)

[*Technical breakdown 4*](#_Toc52357537)

[*Architecture 4*](#_Toc52357538)

[*Context diagram 4*](#_Toc52357539)

[*Sequence Diagram 4*](#_Toc52357540)

[*Process flow Diagram 5*](#_Toc52357541)

[*Network diagrams (Cloud specific) 5*](#_Toc52357542)

[*Client / server demarcation of responsibilities 5*](#_Toc52357543)

[*Response filtering / data object correlation 5*](#_Toc52357544)

[*Test plan 7*](#_Toc52357545)

[*Difficulties / Exclusions / unresolved & persistent errors 8*](#_Toc52357546)

[*Extensions (Optional) 8*](#_Toc52357547)

[*User guide 8*](#_Toc52357548)

[*References 8*](#_Toc52357549)

[*Appendices 8*](#_Toc52357550)

*This template is similar to the one provided for assignment 1. It is not compulsory to use it, but it will save a lot of effort if you do. You should assume that black text in italics is there as guidance and you should read it, follow the instructions and then delete it when you have entered your own text. Some examples are not italicized, but should obviously be replaced by your own material.*

*The report should be around 10-15 pages including screenshots, but this is a guide only – we will not be enforcing a page limit or marking you down for submitting something with 16 pages instead. But be sensible, we really don’t want something that is 25 pages or more. Please note there are examples of previous students’ work in this template & examples from Google searches. They are here to give you ideas on what you can do.* ***We do not guarantee that they are appropriate for your project.***  *You must do your own research and produce your own diagrams.*

# *Introduction*

## *Purpose & Description*

*Twitter Sentiment Radar (TSR) is a web application that allows users to see the emotional rating or sentiment of their desired twitter topic. This information will be displayed on a live graph that measures the sentiment value between 1 and -1. To elaborate, 1 represents that the tweet about the topic was presented positively (happy) and 0 is a negative sentiment value (sad/angry).*

## *APIs and Services used*

*The following APIs and Services were used within the NodeJs environment. All necessary packages were retrieved through their appointed versions with Node package manager.*

## *Twitter Standard Search API (v.1.1)*

*Returns a collection of relevant Tweets matching a specified query – may also be filtered based on popularity or geocoding [and whatever other obvious details we might decide to include]*

*Endpoint: https://api.twitter.com/1.1/search/tweets.json*

*Docs:* [*https://developer.twitter.com/en/docs/twitter-api/v1/tweets/search/api-reference/get-search-tweets*](https://developer.twitter.com/en/docs/twitter-api/v1/tweets/search/api-reference/get-search-tweets)

# Redis

Redis is an in-memory data structure storage that is used as a form of distributed cache. This is the first later within the Persistence hierarchy in Twitter Sentiment Radar.

## *AWS EC2*

*AWS EC2 is Amazon’s cloud service in which the application will be hosted on.*

## *AWS S3*

*Amazon S3 is an object storage service offered by Amazon Web Services. It acts as the second layer within the Persistence hierarchy in Twitter Sentiment Radar.*

# Natural

Natural is a natural language package for NodeJS. It contains functions that allow developers to tokenise, classify, stem, etc.

# Compromise

Compromise is a text parser tool that allows developers to conveniently modify, filter, and match select words and phrases within their string.

# *Use cases*

#### US 1

|  |  |
| --- | --- |
| *As a* | *Twitter user* |
| *I want* | *To see sentimental values of my desired tweet of topic* |
| *So that* | *I can understand how people are feeling about that topic* |

*INSERT SNIPPET OF GRAPH WITH DATA*

# Technical breakdown

*In this report – which covers the group components of the assignment - there should be some coverage of the architecture and the basic operation of the system. Some deeper analysis is now left to the individual report – please read that template and guide for details.*

## Architecture

The back end of this web application starts with the initial instantiation and declaration of all the variables, libraires, and API clients like S3 Bucket and Redis storage.

**SCREENSHOTS**

Promises are also declared early.

**SCREENSHOTS (Redis promise, bucket promise)**

Most of the back end is wrapped with Module.Exports to send values in and around the project directory to where the front end requires.

**SCREENSHOT**

**Andrew talk about get average, write json functions here**

**Screeenshots**

Since there will be frequent occurrences of uploading the Tweets to S3 and Redis, their methods were wrapped in a function.

**SCREENSHOTS**

The following screenshots presents the main block of the back end. It is responsible for handling the retrieval and storage of the tweets as well as the parsing and sentiment analysis. This is all done while adhering to the persistence structure of

**SCREENSHOTS**

*Your principal helper here will be one or more architecture diagrams – which we will consider in some more detail below. However, you may 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. If you have used particular libraries, then you should give us a brief overview of their use in this application.*

*A number of example architecture diagrams are provided below. Many students use diagram generators such as the tools at* [*https://cloudcraft.co/*](https://cloudcraft.co/)*. For assignment 2, this is the most important diagram used to document your approach. The ‘network diagrams’ below show some more complicated alternatives. Only the architecture diagram is compulsory. Please consider the others if they help you, and ignore them if they do not. Obviously you should delete all of the examples and include only diagrams which you have created to explain your application.*

#### Context diagram

|  |  |
| --- | --- |
| *Diagram  Description automatically generated* | *Diagram  Description automatically generated* |

#### Sequence Diagram

|  |  |
| --- | --- |
| *Diagram  Description automatically generated* | *C:\Users\denbi\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\5B3F0D8D.tmp* |

#### Process flow Diagram

|  |  |
| --- | --- |
| *Diagram  Description automatically generated* | *Diagram  Description automatically generated* |

#### Network diagrams (Cloud specific)

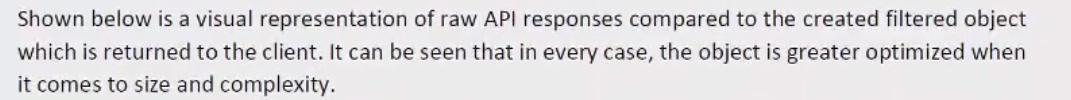
|  |  |
| --- | --- |
| *Diagram  Description automatically generated* | *Diagram  Description automatically generated* |

#### Client / server demarcation of responsibilities

*Explain to us what is doing what & where. Refer to the architecture diagram and any others that you find appropriate. This is particularly effective if you support your comments with well-chosen code fragments. These should be short and focused and you should give us any context that we need to work with them.*

#### Response filtering / data object correlation

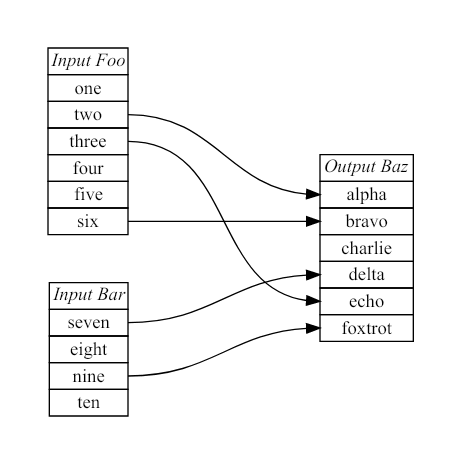
*Show us how you manipulated the data. The same comments apply about referring to the diagrams and supporting your work with code fragments as appropriate. We provide an example below of how to show this diagramatically. Please note that this example is quite specific to the system being explained. Yours might have an entirely different look, but do a similar job. This section will vary markedly according to the application and may not make much sense for some custom applications such as rendering.*

**

*Chart, treemap chart

Description automatically generated*

*You could instead use a data relationship diagram:*

**

### *Scaling and Performance*

*This is a crucial aspect of the report, and you should use this section to document the approach taken to scaling – the nature of the application load, how it was varied and how the scaling infrastructure responded. You should refer to the architectural diagram above or reproduce the relevant aspects here. You should include screenshots of CPU, network or queuing metrics as observed on the cloud services dashboard, together with screenshots of your settings and the scaling pool instance creation and destruction. We expect that your work here will demonstrate successful scale out and scale in as required in the assignment specification. The screen shots that you use here will also very likely be re-purposed as part of the slide deck for the demo.*

*An example scaling image is shown below, and we would normally expect to see this sort of image and some evidence of your group settings. Note the instance count on the left hand axis. As noted, many alternatives are possible.*

**

### *Test plan*

*Manual testing is fine and our expectations are in line with the example grid below. You can show the results through a screen shot and point us to these from the table.*

*Your tests should include*

* *Positive outcome cases*
* *Negative outcome cases (error scenarios)*
* *Edge cases*
* *Non-functional cases*

*Note that the grid below is unrelated to this application.*

*Table

Description automatically generated*

*As they are common in industry you could define your Acceptance Criteria as GWT statements. This is not compulsory, but see:* [*https://www.agilealliance.org/glossary/gwt/*](https://www.agilealliance.org/glossary/gwt/)*. And here is an example:*

*Graphical user interface, text, application, email

Description automatically generated*

*Difficulties / Exclusions / unresolved & persistent errors /*

*In this section, you should explain anything that caused you problems and how you overcame those problems. Tell us if there was any issue that prevented you completing the assignment to specification. Tell us about any assumptions or compromises that you have made. Those who worked with an API like Spotify, which presented particular concerns, should discuss the compromises here, and this is also where you can tell us about problems with API keys and responses.*

*More generally, you might consider:*

* *Your major roadblocks and how you resolved them.*
* *Any functionality you didn’t or couldn’t finish*
* *Are there any differences between your brief and what you delivered? If so, explain why.*
* *Are there any outstanding bugs?*

## *Extensions (Optional)*

*In this section, you can tell us if you wish to how you might extend your app and make it better. This is an opportunity to tell us about good ideas that you had that you didn’t have time to tell us about.*

## *User guide*

*Tell us how to use your application. You may re-use some of the screenshots from the use case descriptions, but this is more about how to use the app. As long as we can find what we need to do to use your application, this need not be all that long.*

*But either way, screenshots are your friend.*

## *References*

*Use a standard approach to referencing – see the guidance at* [*https://www.citewrite.qut.edu.au/cite/*](https://www.citewrite.qut.edu.au/cite/)*.*

## *Appendices*

*Stuff you want to include, but is too long or too complex to include in the main report text. The full Docker file, some longer excerpt from API docs. Whatever helps.*

*[Our thanks to those students who allowed us to use their work in the examples presented above.]*