*Greyden Scott*

*N9935924*

*CAB432 – Cloud Computing*

Assignment 01 – Mashup Project Report

Table of Contents

[Introduction 3](#_Toc524076759)

[News API 3](#_Toc524076760)

[IBM Watson Tone Analyzer 3](#_Toc524076761)

[Google Charts 3](#_Toc524076762)

[Use Cases 4](#_Toc524076763)

[Use Case A 4](#_Toc524076764)

[Use Case B 4](#_Toc524076765)

[Use Case C 4](#_Toc524076766)

[Technical Breakdown 5](#_Toc524076767)

[Client 5](#_Toc524076768)

[Server 5](#_Toc524076769)

[Difficulties 6](#_Toc524076770)

[Filtering Responses 6](#_Toc524076771)

[Extensions 7](#_Toc524076772)

[Cloud Vision 7](#_Toc524076773)

[More News Sources 7](#_Toc524076774)

[Advance Search 7](#_Toc524076775)

[Testing 8](#_Toc524076776)

[Appendix A – User Guide 9](#_Toc524076777)

[Appendix B – Test Case Screenshots 10](#_Toc524076778)

# Introduction

BuzzTone aims to provide users with a visual representation of the descriptions of news articles from the Buzzfeed Website. Utilising the News API, IBM Watson Tone Analyzer and Google Charts, users can search for Buzzfeed news articles on a topic of their choosing and be presented with graphed information relating to the emotional tone of the article.

## News API

<https://newsapi.org>

News API is a well document API that returns JSON metadata for news headlines and articles from all over the web. News API has access to over 30,000 news sources and blogs. However, for the purposes of BuzzTone, information is only obtained from the Buzzfeed news website.

BuzzTone utilises only one API endpoint provided by News API; *newsapi.v2.everything,* which is used to search for news headlines, a URL linking to the original article, an image URL related to the article and the description of the article.

## IBM Watson Tone Analyzer

<https://www.ibm.com/watson/services/tone-analyzer/>

The IBM Watson Tone Analyzer API is one of IBM Watsons machine learning API’s which can be used to analyse emotions and tones in written text. Tone Analyzer breaks down the text inputted, providing the tones it detects and a score for each tone.

BuzzTone uses Tone Analyzer to analyse the Buzzfeed news article descriptions for tones to predict the emotional disposition of a news article.

## Google Charts

<https://developers.google.com/chart/>

Google Charts is a JavaScript API that allows structured data to be displayed in a visual diagram. The API is well documented and easily allows for the creation of interactive diagram charts.

The Google Charts library is included within the client-side index and is obtain via CDN. The diagrams are initialised and displayed when the web page has completed loading.

# Use Cases

## Use Case A

As a Buzzfeed reader I want to search for articles on a subject and see the overall emotional tone so that I can see a visual representation of all article tones.

## Use Case B

As a Buzzfeed reader I want to search for articles on a subject and see the tone for each article so that I can see a visual representation of the specific article tones.

## Use Case C

As a Buzzfeed reader I want to search for articles within a particular date range so that I can see articles from within a particular time frame.

# Technical Breakdown

# Client

The client initiates two types of request, initially a GET request which is sent to the server, the server then in turn serves an index page back to the client. The section request is a POST request which is initiated by the user when the complete a search on the initially returned index page. The POST request is sent to the server in the body of the request message.

The server then handles the information sent from the POST request, the requested information is processed by the server and displayed, providing a Pie Chart of the overall emotional tone of all the articles retrieved, and individual cards displaying information about the articles and a Bar Graph displaying the results for the specific article.

## Server

The Node Server deals with routing incoming client requests and outgoing responses, when a GET request is received from the client, the server deals with the request by calling two external API’s. The first being the News API, which returns a JSON response to the server with Article objects. The server filters these objects and stores the filtered information temporarily in an array of newly created objects with only the relevant information present. This information is then passed to the IBM Watson Tone Analyzer API where the tones for each object are appended to the previously created objects as new object parameters. This information is then forwarded to the renderer to be displayed.

When a POST request is made similar server functionality is performed, the key difference being that the information sent to the server via the POST request is added to the News API to query the API with a specific subject matter that the user has specified. The response from the News API is still processed and passed to the IBM Watson Tone Analyzer API and then forwarded to the rendered to be displayed.

## Docker

Docker has been used to encapsulate the BuzzTone application and include the node server, JavaScript libraries and front-end pug templates.

The build has been derived from the official Docker Node image, using the Node 8 image, carbon.

FROM node:carbon

WORKDIR /App

ADD . /App

EXPOSE 3000

WORKDIR /App/BuzzTone

ENV NODE\_ENV production

RUN npm install

CMD npm start

The first line of the docker file sets the node:carbon as the parent image as previously mentioned, this image contains the necessary dependencies to run the BuzzTone application. The next line sets the working directory to /app, from there we add the /App directory, then 3000 is exposed for the Node server. We then set the working directory to inside the BuzzTone folder, this then allows us to change the environment to production. Then npm is run to ensure dependent node packages are installed and that the packets are in the docker image. The last line then starts the application.

## PUG

PUG has been used to manage the frontend templates for BuzzTone, it is used on the server-side to render the index page. Three pug files are utilised, the first is *layout.pug* which holds the header that contains links to the CSS, JavaScript for some additional CSS functionality and gstatic charts loader. The second file is the index which renders the main page content. When the server renders the index page it attaches the *index.pug* to the *layout.pug*. The third file is another extension of the *layout.pug* called *error.pug* which is used to render potential errors that may occur. Again, this is called through the layout.pug, which ensure that the user can re-run searches in the event they fail.

# Difficulties

## Filtering Responses

Initially BuzzTone was to be a service that presented news headlines from numerous news sources, this proved to be a significant challenged for the IBM Watson Tone Analyzer to process. IBM Watson Tone Analyzer API would not necessarily return a tone for every item sent to its API. News headlines tended to be too short, lacking proper grammar to have analysis performed on them or just down right trivial and un-interpretable.

To combat this, I first attempted to limit the sources of news headlines. However, this had very limited effect, for every 20 headlines sent to IBM Watson Tone Analyzer, only 5 would come back with proper Tone Analysis. Instead I pivoted the design to instead look at the News Article Descriptions.

This presented further challenges, as either too much information was being sent to the API or too many API calls were being made (nearing me towards maxing my monthly API call limit). The solution to this problem was to filter and clean the information before it was being sent to the IBM Watson Tone Analyzer. Buzzfeed article descriptions were all appended with a “View entire article” text, which was unnecessary to process the tone of. Some descriptions also contained special characters which also confused the Tone Analyzer.

By filtering the data being sent to the IBM Watson Tone Analyzer the volume of descriptions being analysed incorrectly drastically decreased.

# Extensions

## Cloud Vision

It was consideration to utilised either Google Cloud vision or IBM Visual Recognition to run analysis on the images returned by the news sources related to the individual articles. However, it was excluded so that more time could be focused towards the Tone Analysis functionality. A form of image recognition would allow users to get further information on the overall tone of the article as accompany imager can also be used to reflect the disposition and emotion in an article.

## More News Sources

The initial design proposed using a wider range of news sources, allowing the user to see a broader selection of news articles on the particular subject searched for from a variety of sources. This functionality was abandoned due to time constraints, as filtering the responses from News API would require a significantly more complex approach to ensure the information provided to the IBM Watson Tone Analyzer produced consistent and accurate results. Including this extension would provide the user with a wider variety of news sources to compare the tones. This could then be extended further to show the difference in tone from one news outlet to another.

## Advance Search

It was initially proposed that users would be able to perform a more complex search, allowing the user to choose how the data would be sorted (relevance or popularity), chose which news sources to obtain articles from and how the data should be represented (different types of graphs). However, this functionality was not implemented due to time constraints. Allowing this functionality would provide the user greater control of the information they search for and how it is displayed.

# Testing

Errors are produced in 2 possible places, either by the News API failing, or by IBM Watson Tone Analyzer failing.

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Expected Outcome | Result | Screenshot/s (Appendix B) |
| Retrieve Top Articles | Results displayed, graphs Populated | PASS |  |
| Search Articles | Results displayed, graphs populated | PASS |  |
| Click Read More button | Redirects to appropriate article source | PASS |  |
| Handle News API response error | Display error, application continues | PASS |  |
| Handle Tone Analyzer Error | Displayer error, application continues | PASS |  |

# Appendix A – User Guide

# Appendix B – Test Case Screenshots