SENTIMENT ANALYSIS FEEDBACK APP

Developmental Documents

UMUC – CMSC495

Professor: Nicholas Duchon

Document Version: 1.4

Revision History:

Date	Version	Author	Description
6/24/17	1.0	Qeturah J.	Initial compilation of comprehensive document
7/1/17	1.1	William D.	Edited text, formatting, added Phase I and Phase II reports
7/6/17	1.2	William D.	Editing formatting of all sections, made new table of contents, eliminated table of contents for previously compiled chapters
7/7/17	1.3	Tyler G.	Edited Text/ Changed formatting
7/15/17	1.4	William D.	Updated User Guide, Test Plan, added Final Report and Conclusions

Contents

SOFTWARE DEVELOPMENT PLAN

UMUC

Document Version: 1.3

Revision History:

Date	Version	Author	Description
5/30/17	1.0	Eric S.	Initial creation and writing of plan
7/1/17	1.1	William D.	Edited text, formatting
7/6/17	1.2	William D.	Edited formatting
7/7/17	1.3	Tyler G.	Changed the tense to present Corrected grammar, and spelling errors Reworded several portions to flow better

Sentiment Analysis App

Version 0.1

30 May 2017

Team Member	**Assigned Duty**
Eric Sabelhaus	Team Lead / Developer
Qeturah Jackson	Test Engineer
William Donabedian	Technical Writer
Justin Wheeler	Developer
Tyler Gibbs	Assistant Technical Writer / Developer

Overview

1.1 Project Summary

1.1.1 Purpose, Scope, and Objectives

- Purpose:
    - This project provides a web-based UI that allows a user to input one or multiple sentences into a text dialogue box. The user can then press a submit button and will be given a sentiment analysis of their provided text.
    -
- Scope:
    - The scope of this project incorporates two application components
    - The first component is a client web interface
    - The second component is a rest endpoint which exposes a POST route for submission of text for analysis
- Objectives
    - Define a user interface which follows industry standard accessibility guidelines
    - Define a REST endpoint which exposes a POST web interface for submitting text for sentiment analysis
    - Document functionality, design, testing, and user instruction of application
    - Deploy a working and functional website which exposes the application functionality defined in the REST endpoint

1.1.2 Assumptions and Constraints

- UI:
    - A user could input meaningless text, but if the AFINN / Emoji datasets can process it, they will produce an analysis
- API
    - This endpoint will kick off a delayed job to avoid binding the main thread of the server application
    - The input will accept new line characters, and individual sentiments will be broken up for the final analysis

1.1.3 Project Deliverables

- A fully functioning web application which can be accesses over HTTP from a server on the internet.
- Documentation on the page to describe what each element represents
- Comprehensive documentation of the process and implementation of the software, as well as attribution for any open source libraries leveraged to make the application

References

- IEEE Std 1058-1998, Software Project Management Plans

Definitions

- Agile – A lean development methodology used by the team to manage requirements on an iterative basis.
- Scrum – Agile based Software Development Lifecycle (SDLC) which incorporates a backlog of tasks to accomplish a requirement or multiple requirements
- TravisCI – Internet based continuous integration platform, free for use with public GitHub projects
- Taiga.io - Free scrum/kanban based project management application.
- Sprint – A measure of time the team uses to iterate features and produce deliverables which meet the acceptance criteria of the requirements
- User interface (UI) - the way through which the user interacts with the product
- Application program interface (API) - a set of routines, protocols, and tools for building software applications

Project Organization

4.1 External Interfaces

- Customer – Dr. Nicholas Duchon

4.2 Internal Structure

- The Sentiment Analysis development team consists of a team lead / senior developer, test engineer, technical writers, and junior developers. This team composition will create a structure allowing for direct development of the software requirements, design and modification of the user interface elements as agreed upon between the client and the software lead developer.

4.3 Roles and Responsibilities

- Team lead / senior developer - Performs the task of project oversight, requirements management, and schedule obligations for the team. Will deliver project-related integrated master schedules (IMS) as well as conduct schedule conflict resolution to mitigate any risks associated with the waterfall methodology. In addition, this position is also responsible for feedback from the customer base, and providing on-time deliverables at periodic milestones within the project lifecycle.
- Test engineer - uses the understanding and configuration of data sources and connectivity to persist data in the Sentiment Analysis tool. They will be required to validate and test requirements associated with saving data.
- Technical writer - Provides documentation for the project and oversight for all related documents. Will deliver the user guide.
- Junior developer - Collaboratively develops software to meet defined requirements with the guidance of the Sr. Developer.

Managerial Process Plans

5.1 Work Plan

-

5.1.1 Work Activities

-

1. Task Name	SDLC Task Category	SDLC Sub-Task
Form Teams	Analysis	Business Analysis
Initial Customer Engagement	Analysis	Business Analysis
Identify Customer Needs and High-Level Requirements	Analysis	Software Analysis
Develop Milestones	Analysis	Business Analysis
Develop Requirements	Design	Software Analysis
Deliver and Discuss Software Requirements Specification	Analysis	Business Analysis
Software Design Review	Design	Software Design
Create Software Design Document	Design	Software Design
Deliver and Discuss Software Design Document	Analysis	Software Design
Create Software Testing Specification	Design	Test Design
Deliver and Discuss Software Testing Specification	Analysis	Business Analysis
Code Background Task Worker	Code	Task runner development
Code Sentiment Analysis process (job for worker)	Code	Design Job code to
Write tests for Worker and Analysis code	Test	Unit testing
Initial User Acceptance Engagement (display worker functionality against sample test data)	Test	User Testing
Code POST endpoint to accept text	Code	Web API Development
Integrate POST endpoint with job submission to worker	Code	Core Functionality Development
Write tests for integration of POST endpoint	Code	Unit Testing
Deploy current code (0.1)	Deployment	Deploy beta functionality
Team review of current functionality	Test / Analysis	Review the capability of POSTing data to web endpoint and receiving expected

		response
Customer Engagement	Analysis	Business Analysis
Code Index page	Code	UI design and implementation
Code Client-side JavaScript for AJAX form submission to POST endpoint	Code	Client side script design
Deploy current code (0.2)	Deployment	Deploy release candidate
Customer Engagement	Analysis	Business Analysis
Produce Extended Functionality User Documentation	Code/Documentation	Documentation Development
Test client facing interface	Test	Ensure functionality with multiple types of data
Fix regressions existent	Review/Code	Fix any bugs which are present in the functionality of the app from the user perspective
Release 1.0	Deployment	Deploy working and deliverable project code
Final Customer Engagement	Analysis	Business Analysis
Final User Acceptance Engagement	Test	User Testing
Delivery of Product (Git repo containing all code and documentation)	Transfer of Code	Release of code to professor

5.1.2 Schedule Allocation

- The project schedule will be fluid, with weekly sprints being accomplished. Each sprint should encompass some major feature, and a retrospective session following that week to adjust to help meet the delivery schedule of the end product. As each sprint completes, there should be some code or documentation which can be displayed to the team and customer for feedback and review, as well as action planning should any regressions be noticed.

5.1.3 Resource Allocation

-

Resource	***Allocation***
Application Server	The server is housed in Digital Ocean, and runs against the Ubuntu 16.04 operating system
Docker	Linux container runtime, is used to allocate application resources on the server for proper application functionality

Table 2: Resource Allocation

5.4 Risk Management Plan

Risk Type	**Probability of Occurrence**	**Impact**	**Loss of time (Days)**	**Exposure (Days)**
Design flaw	moderate	minor	1-3	0
Internet outage	low	high	variable	variable
Regression	moderate	moderate	1-3	0

Table 3: Risk Management Matrix

Technical Process Plans

6.1 Process Model

The methodology used by the Sentiment Analysis tool development team is constructed in an agile fashion for the purpose of program management. Internally, the development team will organize and develop the assigned periods for deliverables in a pseudo-agile fashion, building epics, stories and tasks in conjunction with the overall planning strategy. This model is heavily influenced by the Department of Defense model for government acquisition planning with lean development phases.

The model helps to achieve larger scale planning for acquisition and management, and provides more flexibility to the development teams tasked with implementation.

The process in place accommodates for a start (kickoff) activity based on project assessment and business analysis with and without the customer present. This assists the development of both sub-processes and requirements to generate milestones over the course of the project, and to assign responsibility of development tasks throughout the team. The conclusion of the project, also listed as a milestone, serves as the final delivery after multiple cycles of customer engagement and user acceptance testing.

6.2 Methods, Tools, and Techniques

The Sentiment Analysis tool development team, following the agile methodology for program management in section 6.1, will also require a standard process by which development, testing, integration, and documentation is performed.

The following standards are implemented and adhered to throughout the lifecycle of the project.

- Each feature is assigned to a sprint, which commits to deliver that feature code, either as deployed software, or as a user facing interface to display.
- Software and documentation changes will be added to the git based software repository and pushed up to GitHub
- Every sprint will incorporate development, documentation, testing, and deployment
- The software repository will be integrated with TravisCI on every push of new code to GitHub (this includes documentation changes by design)
- Upon test success within Travis, the project's lead will deliver the designed code to the server for review and testing by the team prior to the next sprint planning

USER GUIDE

Document Version: 1.4

Revision History:

Date	Version	Author	Description
6/10/17	1.0	William D.	Initial Document Creation
7/1/17	1.1	William D.	Edited text
7/7/17	1.2	Tyler G.	Edited text / Grammar Reworded several portions to flow better
7/9/17	1.3	William D.	Added table for examples in 'Interpreting Results' section. Added 'For More Information' section.
7/15/17	1.4	William D.	Made minor revisions and added the app homepage address

Quick Description

The purpose of this application is to give you, the user, the sentiment of an input text as positive, negative, or neutral by providing you with an integer rating of each sentence and an overall rating. Negative texts correspond with a negative value, and positive texts correspond with a positive value. A neutral text returns a value of zero and may imply that words or phrases within the text submitted are not included in the data used as part of the analysis.

Getting Started

First, you need to navigate to the website homepage at https://sentiment-analysis.xyz. The application is compatible with most web browsers used today. At the top of the homepage you will see instructions explaining how to use the Sentiment Analysis Feedback App. To get started, enter the text you would like to analyze in the text dialogue box provided on the homepage.

The application does not limit the size of text that you would like to analyze. After you have entered to text, click the 'Analyze' button. The application will produce additional text at the bottom of the page providing you with an analysis of the text.

For inputs consisting of a single line, the application will return one numeric value indicating the overall sentiment of the text. For inputs consisting of multiple lines separated by a return, the application will return a numeric value indicating the sentiment of each sentence, as well as a numeric value indicating the overall sentiment of the text as a whole.

The application processes the data by parsing words and phrases and comparing them to a collection of words and phrases that have been assigned numeric values that represent sentiment.
If you would like to analyze additional text, you must enter the new text in the text field and click 'Analyze'. To view additional information about how the Sentiment Analyzer works, click the 'About' tab at the top of the homepage.

Interpreting Results

The Sentiment Analyzer application compares the user's text input to an existing list of words and phrases, which were assigned a numeric sentiment value. If words or phrases from the input are found in the list, they receive the associated numeric value from the list. If the words are not in the list, they do not receive a value. All of the values for each sentence are summed to determine a sentiment value for each line. For input of multiple lines separated by a return, the overall sentiment value is calculated by averaging the values of all words and phrases from the input that were found in the list.

Example	Input	Output	Interpretation
1	The cheers filled the room.	2	2 is a positive sentiment
2	Life is to be lived, not controlled; and humanity is won by continuing to play in face of certain defeat.	4	4 is a positive sentiment. It is more positive than the value of 2 seen in Example 1.
3	There was an accident last night.	-2	-2 is a negative sentiment
4	The investigators accidentally destroyed the evidence.	-5	-5 is a negative sentiment. It is more negative than the value of –2 seen in Example 3.
5	The investigators accidentally destroyed the evidence.	0	0 is a neutral sentiment.

For More Information

The list of words and phrases used as part of the Sentiment Analyzer is known as the AFINN list. The AFINN list is a collection of words, phrases, and emojis that have been rated with integer values ranging from negative five to positive five. It can be found here: AFINN.

Test Plan

Document Version: 1.9

Revision History:

Date	Version	Author	Description
6/07/17	1.0	William D.	Initial Document Creation
6/08/17	1.1	Qeturah J.	Added User Acceptance Tests
6/09/17	1.2	William D.	Document revision and formatting
6/10/17	1.3	Eric S.	Added Unit Test specification
6/11/17	1.4	Eric S.	Add Table of Contents Add introduction block
6/11/17	1.5	William D.	Added additional test cases
7/1/17	1.6	William D.	Edited test case text
7/6/17	1.7	William D.	Edited test case formatting and text
7/7/17	1.8	Tyler G	Edited Text / Reworded several portions
7/15/17	1.9	William D.	Corrected error in user acceptance test. Completed tests.

Introduction

1.1 Purpose

1.1.1 The purpose of this document is to provide clear and concise information on testing the Sentiment Analysis Feedback App. It will provide detailed instructions on how to test the application to ensure it's acceptable for use by an end user.

1.2 Background

1.2.1 This application will extend functionality of the Sentiment node module. The module is an implementation of word sentiment analysis based on the AFINN-165 specification and Emoji Sentiment Ranking

1.2.2 If the text entered by the user exists in the AFINN listing, the user will be provided with a numerical value representing the sentiment analysis

1.2.3 The primary purpose is to allow an end user to paste text into a single input form, press a single button, and be given feedback for every line of text, as well as a cumulative score.

1.2.4 The score will be an integer value between negative five and positive five, which represents the overall sentiment of a statement within those parameters.

1.2.5 An alternate function would be to allow multiple sources to extend the REST API, and programmatically interpret text with a simple numerical output.

1.2.6 The tests for user acceptance will ensure end users have the proper experience of the application as expected. The unit tests will assert the functionality is extensible and reliable for non-UI users.

1.3 Scope

1.3.1 This test plan provides a structured approach for manual testing, as well as specifications for writing application unit tests

1.4 References

1.4.1 AFINN-165 - http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010

1.4.2 Emoji Sentiment Ranking - http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0144296

1.5 Test Environment

1.5.1 Operating System: Linux - Ubuntu version 16.04

1.5.2 Software:

a. VirtualBox
b. NodeJS version 8.0
c. Redis.io Server

1.5.3 Hardware

a. Laptop/Desktop of team member

Definitions

Mocha: "simple, flexible, fun javascript test framework for node.js & the browser" - https://github.com/mochajs/mocha

Chai: "Chai is a BDD / TDD assertion library for node and the browser that can be delightfully paired with any javascript testing framework." - http://chaijs.com/

SuperTest: "Super-agent driven library for testing node.js HTTP servers using a fluent API" - https://github.com/visionmedia/supertest

Sentiment Analysis: The use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information.

User Acceptance Test Documentation

For all User Acceptance Test Cases 1-4, the following assumptions are made:

- The user should be able to successfully input text
- The user must be using the English language and character set
- A supported browser is being used

TUA_001: Analysis Success– Authenticate Successfully

Test Steps	Expected Result	Pass/Fail
1. Navigate to website homepage	Site should open	pass
2. In the 'input' field, enter the sentence(s) to be interpreted from Data Set 1.	Text can be entered	pass
3. Click the 'Analyze' button.	A page displaying the sentiment analysis value of 2 should load	pass

TUA_002: Analysis Success– Authenticate Successfully

Test Steps	Expected Result	Pass/Fail
1. Navigate to website homepage	Site should open	pass
2. In the 'input' field, enter the sentence(s) to be interpreted from Data Set 2.	Text can be entered	pass
3. Click the 'Analyze' button.	A page displaying the sentiment analysis value of –5 should load	pass

TUA_003: Analysis Success– Authenticate Successfully

Test Steps	Expected Result	Pass/Fail
1. Navigate to website homepage	Site should open	pass
2. In the 'input' field, enter the sentence(s) to be interpreted from Data Set 3.	Text can be entered	pass
3. Click the 'Analyze' button.	A page displaying the sentiment analysis of 4 should load	pass

TUA_004: Analysis Success– Authenticate Successfully

Test Steps	Expected Result	Pass/Fail
1. Navigate to website homepage	Site should open	pass
2. In the 'input' field, enter the sentence(s) to be interpreted from Data Set 4.	Text can be entered	pass
3. Click the 'Analyze' button.	A page displaying the sentiment analysis value of 0 should load	pass

f. **TUA_005**: Multiple Users

Test Steps	Expected Result	Pass/Fail
1. Two separate users access webpage from different devices	Site should open	pass
2. In the 'input' field, enter the sentence(s) to be interpreted	Text can be entered	pass

from Data Sets 6 and 7.		
3. Click the 'Analyze' button.	A page displaying the sentiment analysis value of 5 for Data Set 6 and value of –2 for Data Set 7 should load	pass

TUA_006: Checked Exception – Null Entry

Test Steps	Expected Result	Pass/Fail
1. Navigate to website homepage	Site should open	pass
2. In the 'input' field, do not enter any text	Null entry	pass
3. Click the 'Analyze' button.	A message will appear prompting the user to enter text	pass

TUA_007: Analysis Success– Authenticate Successfully

Test Steps	Expected Result	Pass/Fail
1. Navigate to website homepage	Site should open	pass
2. In the 'input' field, enter the sentence(s) to be interpreted from Data Set 8.	Text can be entered	pass
3. Click the 'Analyze' button.	A page displaying the sentiment analysis value of –2 should load	pass

TUA_008: Analysis Success– Evaluate Junk Text

Test Steps	Expected Result	Pass/Fail
1. Navigate to website homepage	Site should open	pass
2. In the 'input' field, enter the sentence(s) to be interpreted from Data Set 9.	Text can be entered	pass
3. Click the 'Analyze' button.	A page displaying the sentiment analysis value of 0 should load	pass

TUA_009: Analysis Success– Authenticate Large Input Successfully

Test Steps	Expected Result	Pass/Fail
1. Navigate to website homepage	Site should open	pass
2. In the 'input' field, enter the sentence(s) to be interpreted from Data Set 10.	Text can be entered	pass
3. Click the 'Analyze' button.	A page displaying the sentiment analysis value of Data Set 10 should load	pass

Unit Test Documentation

For all Unit Test Cases, the following strategy will be used:

- For every function which has an expected return, that function should have at least one associated unit test which asserts the code works in expected conditions.
- There may also be tests written for unexpected conditions to assert that error cases are being handled appropriately.
- These tests will be written using the Mocha unit testing framework for NodeJS, in conjunction with Chai to provide assertion based validation for each test
- Tests must be written in an isolated fashion.

TU_001: Testing of Analysis:

1. Description**:** This test will programmatically assert that the inputs to each function which makes up the analysis provides the expected output.
2. Precondition: N/A
3. Assumption: N/A

TU_002: Testing of POST REST Web Endpoint:

1. Description**:** This test will programmatically assert that the with inputs to each function which makes up the POST rest endpoint of the application provides the expected output.
2. Precondition: N/A
3. Assumption: N/A

TU_003: Testing of UI Server Web Endpoint:

1. Description**:** This test will programmatically assert that the inputs to each function which makes up the user facing web endpoint provides the expected output.
2. Precondition: N/A
3. Assumption: N/A

Test Plan Data Set

Data Set	Text	Expected Result
1	The cheers filled the room. It was hard to hear over the applause.	**Sentence 1: 2 Sentence 2: 1 Overall: 3**
2	The investigators accidentally destroyed the evidence.	**Sentence 1: -5 Overall: -5**
3	Life is to be lived, not controlled; and humanity is won by continuing to play in face of certain defeat.	**Sentence 1: 4 Overall: 4**
4	The only people for me are the mad ones, the ones who are mad to live, mad to talk, mad to be saved, desirous of everything at the same time, the ones who never yawn or say a commonplace thing, but burn, burn, burn like fabulous yellow roman candles exploding like spiders across the stars.	**Sentence 1: 0 Overall: 0**
5	We always enjoy a good game.	**Sentence 1: 5 Overall: 5**
6	The men faced one hardship after another. They did not believe they could go on.	**Sentence 1: -2 Sentence 2: 0 Overall: -2**
7	The group was accepting of its new member. She later admitted that she felt nervous.	**Sentence 1: 1 Sentence 2: -3 Overall: -2**
8	Ajjjababble numcler charltwor mumphensingen.	**Sentence 1: 0 Overall: 0**
9	[Text sample consisting of 5,000 words]	**Sentiment for any lines separated by return and overall sentiment are returned**

Application Design

Document Version: 1.6
Revision History

Date	Version	Author	Description
06/16/17	1.0	Eric S., William D., Justin W.	Defined functions Input / Output Interfaces Data Structures
6/17/17	1.1	Eric S.	Problem analysis Software definition Definitions section
6/18/17	1.2	Eric S.	Design Considerations Functional Design
6/18/17	1.3	William D., Tyler G.	Rewording Added detail to interfaces
6/18/17	1.4	Eric S.	Add table of contents
7/6/17	1.5	William D.	Edited formatting and text, removed table of contents
7/7/17	1.6	Tyler G.	Corrected errors / edited text

Problem Analysis

Computer science is a new topic of interest and focuses on data analysis which occurs through natural language processing. In natural language processing, there are multiple schools of focus. One of the primary techniques for processing and understanding big data is sentiment analysis. Sentiment analysis seeks to provide a programmatic meaning of text. Simply put, it can help provide a better understanding of whether text is positive or negative in nature. This process can be very helpful in correlating new understanding of words previously undefined as having positive or negative connotation, and as a result better understanding language itself.

Currently, numerous libraries exist to implement forms of sentiment analysis, but they were designed as software packages, which must be implemented within other software solutions to expose the functionality. None of them have been made easily available to the average person via the internet. This application seeks to provide such a solution to the average person, detailing the overall sentiment of a given body of text through the use of a simple web interface.

Definitions

Synchronous – When there are two pieces of functionality which must happen in a serial fashion, and they are in sync as a result. This usually occurs when one function requires the output of another function, and they must happen serially.

Asynchronous – When there are two pieces of functionality which can occur independently of one another, and as a result, both functions may happen at the same or different time. The second may depend on the first, but they can occur at various times, and not impact the overall functionality of an application. This is usually associated with single threaded software languages, as they often cannot allow the response of a function to block the primary application thread, else the application could think the server it is running on offline

Background Job – A secondary process, usually spawned from a parent process. These jobs run as a separate entity from an application, which does not impede the functionality of the primary process, allowing the primary process to function uninterrupted, and more seamlessly from the end user's perspective. This technology is often associated with single threaded software languages, as it allows a primary application to avoid becoming CPU bound and locking up.

CRUD – acronym to define create, read, update, and delete. This represents the primary methods of REST used to implement an HTTP server. The mapping is as follows, create (POST), read (GET), update (PUT), delete (DELETE). These methods are used to define a rest endpoint which can be leveraged by other servers over HTTP, to perform various functions on data programmatically.

DOM – With respect to a web browser, this is the Document Object Model, which is used to define visual elements within a web browser.

Design Considerations

Time Constraint

- Given the short implementation time available for the class, it is prudent that the team utilize a software solution which can be successfully crafted and deployed in that short time.

Familiarity

- The team lead has prior experience designing and implementing software using the various open sources solutions which will be utilized on this application.
- As such, the developers and technical writers can glean experience of enterprise application development and deployment due to that familiarity in a mentoring fashion.

Simplicity

- Related to Time Constraints, the simplicity with which the application will be designed will help everyone involved better grasp the design and development concepts used to implement this software solution

Software Choices

JavaScript / NodeJS

- The application will be written using JavaScript in the web server development framework NodeJS
- Given the single threaded nature of JavaScript, the application will be written in a functional and asynchronous format
- Any synchronous functions will be designed as background jobs to avoid blocking the main execution thread of the application

ExpressJS

- Within npmjs.com, there is a well-defined HTTP web server framework provided in the form of an open source module
- This can be found here: https://www.npmjs.com/package/express
- This plugin provides an extensible API for creating consistent web server architecture that utilizes the CRUD restful web server designs.

Sentiment plugin

- Within npmjs.com, there is a well-defined implementation of sentiment analysis provided in the form of an open source module
- This can be found here: https://www.npmjs.com/package/sentiment
- The plugin is extensible, and easily implemented within an application

Nginx

- Nginx is a widely used, and industry trusted web server
- It is often used as a reverse HTTP proxy, and allows a simple application to run on a local HTTP port, and be securely served up over SSL, as opposed to implementing SSL at the entry points of an application
- This design strategy has been used by millions of websites to securely serve up their content over TLS with little effort, and great success

Functional Design

Function:

- Analyze – Provides:
    - Analysis– This will process the incoming JSON object from the client web interface, and determine whether there has been an error on submission. It will process each line of sentiment provided.
        - Some function to interact with sentiment
        - Some function to iterate over array of strings

        - Some function to wrap those and return data
    - Result of analysis – This will format and return the data which was created as a result of the completion of a sentiment analysis. It will be returned in an asynchronous manner to avoid overloading the server, and binding the CPU to singular events for long periods of time.
- Router – Provides:
    - GET
        - This implements the endpoint to serve up the index page of the application
        - It will be the only graphical user interface endpoint of the application
        - This will be what is considered a single page app, as a result
    - POST
        - This implements the endpoint to accept input, and create a background job to process the web request for sentiment analysis
        - Once the job has completed, an asynchronous JavaScript function within the client space will receive the data response and inject it into the DOM accordingly
- Logger – Provides:
    - Used as a consistent log message bus, and will be implemented such that every function of the application provides various levels of logging for levels defined as:
    - Debug – helpful for application developers and testers alike
    - Info – extensive information, good for some debugging by those who implement the web server in a production space to get more helpful information
    - Error – designed to use for logging when deployed in production as a means of observing generic operations within the application, and not to flood the log with unnecessary information
- Client Side – Provides:
    - User interface leveraging HTML5 and CSS3 standards to design a simple user input form
    - JavaScript functionality to pre-process the text provided by a user into an array of strings
    - JavaScript AJAX request to the POST endpoint using JSON formatted array. This function will reformat the page on response from the server

Interfaces

Restful web interface

- Allows requesting systems to access and manipulate textual representations of Web resources using a predefined set of stateless operations

Graphical User Interface (GUI)

- Visual way of interacting with a computer using items such as windows, icons, and menus
- Implemented using HTML5 and CSS3 standards

Data structures

Array – Arrays of strings will be utilized in the transport of data to and from the server and the client
String – Unicode encoded strings will be input and output for the application
Token – Strings will be tokenized by sentiment plugin into individual sentiments to be analyzed and measured.

Input formats

Array of Unicode String(s)

- The client side JavaScript, on submit, will validate and chunk the text provided into Unicode encoded lines, and that will be what is submitted to the POST endpoint for analysis
- There is a potential that users may input emoji icons
- Should emoji exist in the input, the expectation is that the emoji will also be processed as a sentiment if it exists within the Emoji Sentiment Ranking data implemented by the sentiment plugin.

HTTP Web Requests:

- All input to the application from external sources will be made over POST and GET http method requests.
- The POST web request will be required to be in the following format:

Output formats

JSON Response to be returned from POST endpoint:

```
{
 "overall": 0,
  “processed_sentiments”: [
    {
      “score”: 0,
      “sentence”: “this is a sentence”
    },
    {
      “score”: #,
      “sentence”: “line”
  ]
}
```

This response will be consumed by an asynchronous JavaScript function within the web browser of a user at the website, and will reformat the page to display the results according to the response data.

Overall Design

The design choices made for this application allow it to run on a low resource server. This will allow someone to later implement this as a clustered swarm of servers capable of providing sentiment analysis against very large sums of data as a part of big data analysis. The additional purpose behind much of this design is to practice integration of existing software in a way that resembles modern development practices, and allows for development of modern RESTful web interfaces.

Phase I Report

Phase 1 Deliverable

- Source Code: https://github.com/esabelhaus/cmsc_495_sentiment_analysis/tree/phase_1

Work Breakdown

- Eric Sabelhaus
  - Coordinated developer team
  - Provided guidance to achieve basic functionality and Unit Test acceptance
- Qeturah Jackson
  - Worked on Unit Tests
  - Compiled previous documents into new format
- Will Donabedian
  - Wrote Phase I Report
  - Produced mock-up of web app GUI
- Tyler Gibbs
  - Developed code to achieve basic functionality
  - Worked on Unit Tests
- Justin Wheeler
  - Developed code to achieve basic functionality
  - Worked on Unit Tests

Original Milestones for Phase 1

- Implement basic functionality
  - Implement code for logger, queue, and router
  - Implement code for scheduler
  - Get app to the point that it can accept Unit Tests

Schedule Status

- We are on schedule

Special Problems

- Two members of our team are located on different continents. This presented logistical issues with coordinating meetings, etc.
- The software language for this project is new to everyone except for one team member.

Reevaluation of Previous Decisions

- We have not had to reevaluate or change any of our previous decisions.

Changes to previous documents

- All documents were compiled into one master document
- No other changes were made this week

Phase II Report

Phase 2 Deliverable

- Source Code: https://github.com/esabelhaus/cmsc_495_sentiment_analysis/tree/phase_2

Work Breakdown

- Eric Sabelhaus
  - Coordinated developer team
  - Provided guidance to enhance functionality of Phase II code base
  - Updated the method signature of all queue functions to remove the need to specify a job
  - Debugged code
  - Updated Tests
- Qeturah Jackson
  - Worked on Unit Tests
- Will Donabedian
  - Wrote Phase II Report
  - Enhanced mock-up of web app GUI
- Tyler Gibbs
  - Worked on sentiment analysis job within queue library
  - Worked on worker function within queue library
  - Worked on router library
- Justin Wheeler
  - Worked on sentiment analysis job within queue library
  - Worked on worker function within queue library
  - Worked on router library

Original Milestones for Phase 1

- Implement enhanced functionality
  - Add additional functions in queue library
  - Update router library
  - Use logging for debugging

Schedule Status

- We are on schedule

Special Problems

- Two members of our team are located on different continents. This continues to present logistical issues with coordinating meetings, etc.
- The software language for this project is new to everyone except for one team member.

Reevaluation of Previous Decisions

- We have not had to reevaluate or change any of our major decisions. Minor adjustments have been made to several previously submitted documents.

Changes to previous documents

- Modifications were made to the User Guide and Test Plan
- Some formatting changes were made to the development documents (the compilation of everything)

Phase III Report

Phase 3 Deliverable

- Source Code: https://github.com/esabelhaus/cmsc_495_sentiment_analysis/tree/phase_3

Work Breakdown

- Eric Sabelhaus
  - Coordinated developer team
  - Provided guidance to enhance functionality of Phase III code base
  - Updated the code to include the requirements for functioning UI
  - Debugged code
  - Updated Tests
- Qeturah Jackson
  - Worked on Unit Tests
- Will Donabedian
  - Enhanced mock-up of web app GUI
  - Assisted with Phase III Report
- Tyler Gibbs
  - Wrote Phase III Report
  - Updated documentation based on Peer Review Data
- Justin Wheeler
  - Worked on a functioning UI

Original Milestones for Phase 3

- Implement enhanced functionality
  - Creating a functioning UI

Schedule Status

- We are on schedule

Special Problems

- Two members of our team are located in different time zones, and work different times of the day. This continues to present logistical issues with coordinating meetings, etc.
- The software language for this project is new to everyone except for one team member.

Reevaluation of Previous Decisions

- We've decided to prevent the application from running background as a background jobs. We found that due to a circular JavaScript design issue with HTTP requests and delayed job processing, it was not possible to use background jobs in the intended design. As a result, we've redesigned the application to utilize the internal JavaScript asynchronous callback loop to perform non-blocking operations when performing sentiment analysis.

- Documents were reformatted to fit into the comprehensive developmental document.

Changes to previous documents

- Modifications were made to the Development Plan and User Guide
- Most of the modifications were for grammar and to make the structure flow better

Final Report

Phase 3 Deliverable

- Final Deliverable Website: http://sentiment-analysis.xyz/
- Source Code: https://github.com/esabelhaus/cmsc_495_sentiment_analysis/tree/master

Work Breakdown

- Eric Sabelhaus
  - Coordinated developer team
  - Provided guidance to enhance functionality of Phase III code base
  - Updated the code to include the connection between functioning UI and backend code
  - Debugged code
- Qeturah Jackson
  - Worked on testing
- Will Donabedian
  - Modified information in the GUI
  - Completed user acceptance testing
  - Updated comprehensive document
  - Produced final report and conclusions
- Tyler Gibbs
  - Updated comprehensive document
  - Worked on testing
  - Updated the code to include the connection between functioning UI and backend code
  - Debugged code
- Justin Wheeler
  - Updated the code to include the connection between functioning UI and backend code
  - Debugged code

Original Milestones for Final

- Implement final deliverable functionality
  - Create a functioning UI that is connected to backend code and generates real results based on real data

Schedule Status

- We are on schedule and delivered the final product

Special Problems

- Two members of our team are located in different time zones, and work different times of the day. This continues to present logistical issues with coordinating meetings, etc.
- The software language for this project is new to everyone except for one team member.

Reevaluation of Previous Decisions

- As in our last phase, we decided to prevent the application from running background as a background jobs. We found that due to a circular JavaScript design issue with HTTP requests and delayed job processing, it was not possible to use background jobs in the intended design. As a result, we've redesigned the application to utilize the internal JavaScript asynchronous callback loop to perform non-blocking operations when performing sentiment analysis.
- We changed the way that the overall sentiment of the input is calculated. Previously, it was calculated by averaging the values of lines that are separated by returns. We decided that it would be a better representation to average the value of all tokens in the overall text.

Changes to previous documents

- Modifications were made to the Test Plan to correct errors.
- The User Guide was updated to include the actual app homepage.
- Comprehensive report was edited again for grammar and to make the structure flow better.

Conclusions

Lessons Learned

Completing this group project taught us many lessons. This was the largest project most of us have ever worked on, so the planning and delegation of tasks was especially important. Most of us did not have previous experience with project management and breaking down a large project into manageable chunks. This was a very educational experience and illustrated the importance of a strong foundation in the project plan and test plan phases.

Working in a team on a project of this scale was also a first for most of us. We had to overcome obstacles like having team members in different parts of the world, widely varied schedules and personal commitments. The use of tools like Slack, Github, Google Hangouts, and OneDrive was indispensable. We were able to coordinate seamlessly across time zones at different times of the day. Most of us were a little worried going into this project but found that everyone was able to contribute and work toward delivering the final product.

Design Strengths

- Allows user to insert text of unlimited size
- Provides analysis text and emojis
- The sentiment analysis is based on open source, widely available data
- The sentiment analysis allows for the updating of sentiment values for words, phrases, and emojis
- Is designed in an extensible way so that other applications could easily extend this functionality through web requests
- Written with unit and integration tests to allow for easier future upgrades to dependencies
- Designed with modularity in mind for easier design changes in the future
- Leverages existing functionality from open source software to flexibly design and implement more robust features in the given time

Limitations

- If the user wishes to receive an analysis of each sentence of a text, they must put each sentence on its own line by separating them with a return
- The analysis of the text is based on limited lists of words, phrases, and emojis. These lists could be further expanded and refined to produce a more accurate assessment of the sentiment of an input text.
- The software language used was new to a large majority of the team, but it provided a great opportunity to learn and build new skills

Suggestions for Future Improvement

- Change the parsing of the text to provide the sentiment of each sentence separated by a comma if desired by the user
- The AFINN list could be updated to include a greater number of words, phrases, and emojis.

- Possibly, using word elimination, the application could begin performing heuristic analysis based off the sentiment score of surrounding words within a line. This could allow for it to provide a tailored dictionary to the AFINN creators to define a larger subset of words with tokenization. It could also provide an additional dictionary to source analysis from.
- Design a feature for Optical Character Recognition on submission of a picture or PDF to extract text for analysis. This could allow someone to analyze a book from scanned pages potentially, or submit a book PDF file to be processed to raw text and then submitted for analysis
- It may also be neat to store sentiments as they're processed for future analysis and possible applications of machine learning. Since it's unstructured data, it could easily be stored in ElasticSearch or MongoDB with little to no overhead added to a web request