**SENTIMENT ANALYSIS FEEDBACK APP**

Developmental Documents

UMUC – CMSC495

Professor: Nicholas Duchon

|  |  |
| --- | --- |
| **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 |

**SOFTWARE DEVELOPMENT PLAN**

**Sentiment Analysis App**

**Version 0.1**

**30 May 2017**

**Change History**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

1. Overview2

1.1 Project Summary2

1.1.1 Purpose, Scope, and Objectives2

1.1.2 Assumptions and Constraints2

1.1.3 Project Deliverables3

2. References3

3. Definitions4

4. Project Organization4

4.1 External Interfaces4

4.2 Internal Structure4

4.3 Roles and Responsibilities6

5. Managerial Process Plans6

5.2 Work Plan6

5.2.1 Work Activities6

5.2.2 Schedule Allocation8

5.2.3 Resource Allocation9

5.4 Risk Management Plan10

6. Technical Process Plans10

6.1 Process Model10

6.2 Methods, Tools, and Techniques11

1. Overview
2. Project Summary
3. Purpose, Scope, and Objectives

* Purpose:
* This project will provide a web UI which allows any arbitrary person to input one or multiple sentences, and upon pressing a submit button, will be provided with the discernable sentiment analysis of the provided text.
* Scope:
* The scope of this project will incorporate two application components
* The first component is going to be a client web interface
* The second component will be a rest endpoint which exposes a POST route for submission of text for analysis
* Objectives
* The primary objectives will be:
* 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. Assumptions and Constraints

* UI:
* A user could input garbage text, but if the AFINN / Emoji datasets can process it, they will
* API
* This endpoint should also kick off a delayed job to avoid binding the main thread of the server application
* The expectation, is that the input will accept new line characters, and that is how individual sentiments will be broken up for the overall score

1. Project Deliverables

* At the end of this project, the team should have a fully functioning web application which is served up over HTTP from a server on the internet.
* There should also be clear documentation on the page to describe what each element represents
* Lastly there should be comprehensive documentation of the process and implementation of the software, as well as attribution for any open source libraries leveraged to make the application what it is.

1. References

* IEEE Std 1058-1998, Software Project Management Plans

1. 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 over which the team will iterate on features and be expected to 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

1. Project Organization
2. External Interfaces

* Customer – Dr. Nicholas Duchon

1. Internal Structure

* The Sentiment Analysis development team will consist of a team lead / senior developer, test engineer, technical writer, and junior developer. 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.

1. 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.

5. 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 will be housed in Digital Ocean, and run against the Ubuntu 16.04 operating system |
| *Docker* | Linux container runtime, will be used to allocate application resources on the server for proper application functionality |
|  |  |
|  |  |

**Table 2: Resource Allocation**

1. 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**

6.Technical Process Plans

6.1 Process Model

The methodology to be used by the Sentiment Analysis tool development team shall be constructed in a agile fashion for the purpose of program management. Internally, the development team may organize and develop within the assigned periods for deliverables in a pseudo-agile fashion, building epics, stories and tasks in conjunction with the overall planning strategy for the team. This model is heavily influenced by the Department of Defense model for government acquisition planning with lean development phases. This model helps to achieve larger scale planning for acquisition and management, but provide 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 provides for the development of both sub-processes and requirements which are used 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 of the project 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 to be implemented and adhered to throughout the lifecycle of the project.

* As each feature is assigned to a sprint, that is a commitment 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 lead will deliver the designed code to the server for review and testing by the team prior to the next sprint planning

**USER GUIDE**

**Change History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

1. Quick Description
2. Getting Started
3. Interpreting Results

1. Quick Description

The purpose of this application is to give you, the user, the overall sentiment of an input text as positive, negative, or neutral by providing you with an integer value.  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 within the text submitted are not captured in the data used as part of the analysis.

1. Getting Started

When you navigate to the website homepage, the site should load.  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 input field.  The application does not limit the size of text that you would like to analyze.  If you attempt to input text greater than this size, you will receive an error message. Also, if you input no text at all or completely illegible text, you will be presented an error message.  When you are ready to analyze the text, click the 'Submit' button.  After clicking 'Submit' a new page will load providing you with an analysis of the text.  For inputs consisting of a single sentence,  the application will return one numeric value indicating the overall sentiment of the text.  For inputs consisting of multiple sentences, the application will return a numeric value indicating the sentiment of each sentence separated by a period, as well as a numeric value indicating the overall sentiment of the text as a whole. The program arrives at this conclusion by parsing and comparing the input to a collection of words and phrases that have been manually assigned numerical values that represent sentiment.

If you would like to analyze additional text, you must return to the homepage.  To return to the homepage, click the 'Return to Homepage' button.  Once there, you can follow the same process outlined above to analyze additional text.

1. Interpreting Results

The App will compare the text input to the AFINN list. The AFINN list is a collection of words that have been rated with integer values from minus five to positive five. Each word in the user input will be assigned a number value, and then the App will average the results to a single integer value output. The following examples will provide you with a better idea of how to interpret the results you get from the Sentiment Analysis Feedback App.

*Example (1)*

You enter the following text in the text field and press 'Submit': "The cheers filled the room."

This text will return a numeric value of 2, which is a positive sentiment.

*Example (2)*

You enter the following text in the text field and press 'Submit':  "Life is to be lived, not controlled; and humanity is won by continuing to play in face of certain defeat."

This text will return a numeric value of 4, which is a positive sentiment.  The value of 4 is considered more positive than the value of 2 seen in Example 1.

*Example (3)*

You enter the following text in the text field and press 'Submit': "There was an accident last night."

This text will return a numeric value of –2, which is a negative sentiment.

*Example (4)*

You enter the following text in the text field and press 'Submit': "The investigators accidentally destroyed the evidence."

This text will return a numeric value of –5, which is a negative sentiment.  The value of –5 is considered more negative than the value of –2 seen in Example 3.

*Example (5)*

You enter the following text in the text field and press 'Submit': "The crowd accidentally applauded at the show."

This text will return a numeric value of 0, which is considered a neutral sentiment.  Neutral sentiments are neither positive nor negative.

**Test Plan**

**Document Version: 1.5**

**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 |

**Table of Contents**

1. Introduction
   1. Purpose
   2. Background
   3. Scope
   4. References
   5. Test Environment
2. Definitions
3. User Acceptance Test Documentation
4. Unit Test Documentation
5. Test Plan Data Set

1. **Introduction**
2. **Purpose**
3. The purpose of this document is to provide clear and informative information on how to adequately test the Sentiment Analysis Feedback App. It will provide detailed instructions on how to test the application such that it would be acceptable for use by an end user.
4. **Background**
5. 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
6. At the core functionality, the expectation is that a numerical representation of how positive or negative a sentiment is will be given, so long as a word within the sentence provided by an end user exists in these two data sets
7. 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.
8. The score, as defined by the plugin, is between negative five, and positive five, providing the overall sentiment of a statement within those parameters.
9. An alternate purpose of such an application, would be to allow multiple sources to extend the REST API, and programmatically interpret text with a simple numerical output.
10. The tests for user acceptance will ensure that end users have the proper experience of the application as expected, and the unit tests will assert that the functionality will be extensible and reliable for non-UI users.
11. **Scope**
12. This test plan will provide a structured approach for manual testing, as well as specifications for writing application unit tests
13. **References**
14. AFINN-165 - <http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010>
15. Emoji Sentiment Ranking - <http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0144296>
16. **Test Environment**
17. Operating System: Linux - Ubuntu version 16.04
18. Software:
19. VirtualBox
20. NodeJS version 8.0
21. Redis.io Server
22. Hardware
23. Laptop/Desktop of team member

1. **Definitions**
2. **Mocha: "**simple, flexible, fun javascript test framework for node.js & the browser" - <https://github.com/mochajs/mocha>
3. **Chai: "**Chai is a BDD / TDD assertion library for [node](http://nodejs.org/) and the browser that can be delightfully paired with any javascript testing framework." - <http://chaijs.com/>
4. **SuperTest: "**Super-agent driven library for testing node.js HTTP servers using a fluent API" - <https://github.com/visionmedia/supertest>
5. **Redis.io: "**Redis is an open source (BSD licensed), in-memory data structure store**" -**<https://redis.io/>
6. **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.
7. **User Acceptance Test Documentation**
8. **TUA\_001**: Analysis Success– Authenticate Successfully
9. Description**:**A user should be able to successfully input text
10. Precondition: the user must already be using the English language and character set.
11. Assumption: a supported browser is being used.

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

1. **TUA\_002**: Analysis Success– Authenticate Successfully
2. Description**:**A user should be able to successfully input text
3. Precondition: the user must already be using the English language and character set.
4. Assumption: a supported browser is being used.

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

1. **TUA\_003**: Analysis Success– Authenticate Successfully
2. Description**:**A user should be able to successfully input text
3. Precondition: the user must already be using the English language and character set.
4. Assumption: a supported browser is being used.

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

1. **TUA\_004**: Analysis Success– Authenticate Successfully
2. Description**:**A user should be able to successfully input text
3. Precondition: the user must already be using the English language and character set.
4. Assumption: a supported browser is being used.

|  |  |  |
| --- | --- | --- |
| **Test Steps** | **Expected Result** | **Pass/Fail** |
| 1. **Navigate to website homepage** | Site should open |  |
| 1. **In the ’input’ field, enter the sentence(s) to be interpreted from Data Set 4.** | Text can be entered |  |
| 1. **Click the ‘Submit’ button.** | A page displaying the  sentiment analysis value of –3 should load |  |

1. **TUA\_005**:  Analysis Fail –Character Overload
2. Description**:**A user should be able to receive error message when inputting more than 1MB worth of text
3. Precondition: the user must already be using the English language and character set.
4. Assumption: a supported browser is being used.

|  |  |  |
| --- | --- | --- |
| **Test Steps** | **Expected Result** | **Pass/Fail** |
| 1. **Navigate to website homepage** | Site should open |  |
| 1. **In the ’input’ field, enter more than 5,000 words worth of text from Data Set 5.** | Text can be entered |  |
| 1. **Click the ‘Submit’ button.** | A page displaying the error message of the provided text should load |  |

1. **TUA\_006**: Multiple Users
2. Description**:**Multiple users should be able to successfully access the website and input text from different servers
3. Precondition: the user must have an internet connection.
4. Assumption: a supported browser is being used.

|  |  |  |
| --- | --- | --- |
| **Test Steps** | **Expected Result** | **Pass/Fail** |
| 1. **Two separate users access webpage from different devices** | Site should open |  |
| 1. **In the ’input’ field, enter the sentence(s) to be interpreted from Data Sets 6 and 7.** | Text can be entered |  |
| 1. **Click the ‘Submit’ button.** | A page displaying the  sentiment analysis value of 5 for Data Set 6 and value of –2 for Data Set 7 should load |  |

1. **TUA\_007:**Checked Exception – Null Entry
2. Description**:**A user inputs nothing and submits
3. Precondition: the user must already be using the English language and character set.
4. Assumption: a supported browser is being used.

|  |  |  |
| --- | --- | --- |
| **Test Steps** | **Expected Result** | **Pass/Fail** |
| 1. **Navigate to website homepage** | Site should open |  |
| 1. **In the ’input’ field, do not enter any text** | Null entry |  |
| 1. **Click the ‘Submit’ button.** | A message will appear prompting the user to enter text |  |

1. **TUA\_008:**Analysis Success– Authenticate Successfully
2. Description**:**A user should be able to successfully input text
3. Precondition: the user must already be using the English language and character set.
4. Assumption: a supported browser is being used.

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

1. **TUA\_009:**Analysis Success– Evaluate Junk Text
2. Description**:**A user should be able to successfully input text
3. Precondition: the user must already be using the English language and character set.
4. Assumption: a supported browser is being used.

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

1. **TUA\_010:**Analysis Success– Authenticate Large Input Successfully
2. Description**:**A user should be able to successfully input text
3. Precondition: the user must already be using the English language and character set.
4. Assumption: a supported browser is being used.

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

**Unit Test Documentation**

1. **TU\_001:** Testing of Queue:
2. Description**:** This test will programmatically assert that the inputs to each function which makes up the queue for background jobs provides the expected output.
3. Precondition:
4. Redis.io server must be running for queue to functionally accept new tasks
5. Assumption: N/A
6. Test Strategy:
7. 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.
8. There may also be tests written for unexpected conditions to assert that error cases are being handled appropriately.
9. 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
10. Tests must be written in an isolated fashion.

1. **TU\_002:** Testing of Worker:
2. Description**:** This test will programmatically assert that the inputs to each function which makes up the worker code (job which is sent to queue) provides the expected output.
3. Precondition: N/A
4. Assumption:
5. Inputs are given in the form of the text within the Test Plan Data Set, not in a randomized fashion
6. Test Strategy:
7. 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.
8. There may also be tests written for unexpected conditions to assert that error cases are being handled appropriately.
9. 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
10. Tests must be written in an isolated fashion.
11. **TU\_003:** Testing of POST REST Web Endpoint:
12. 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.
13. Precondition: N/A
14. Assumption: N/A
15. Test Strategy:
16. 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.
17. There may also be tests written for unexpected conditions to assert that error cases are being handled appropriately.
18. These tests will be written using the Mocha unit testing framework for NodeJS, in conjunction with  SuperTest and Chai to provide assertion based validation for each test.
19. Tests must be written in an isolated fashion.

1. **TU\_004:** Testing of UI Server Web Endpoint:
2. Description**:** This test will programmatically assert that the inputs to each function which makes up the user facing web endpoint provides the expected output.
3. Precondition: N/A
4. Assumption: N/A
5. Test Strategy:
6. 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.
7. There may also be tests written for unexpected conditions to assert that error cases are being handled appropriately.
8. These tests will be written using the Mocha unit testing framework for NodeJS, in conjunction with Supertest and Chai to provide assertion based validation for each test.
9. Tests must be written in an isolated fashion.
10. **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: -3**  **Overall: -3** |
| **5** | [Text sample consisting of 5,001 words] | **Error** |
| **6** | We always enjoy a good game. | **Sentence 1: 5**  **Overall: 5** |
| **7** | The men faced one hardship after another.  They did not believe they could go on. | **Sentence 1: -2**  **Sentence 2: 0**  **Overall: -2** |
| **8** | The group was accepting of its new member.  She later admitted that she felt nervous. | **Sentence 1: 1**  **Sentence 2: -3**  **Overall: -2** |
| **9** | Ajjjababble numcler charltwor mumphensingen. | **Sentence 1: 0**  **Overall: 0** |
| **10** | [Text sample consisting of 5,000 words] | **Sentiment for each sentence**  **and overall sentiment are returned** |

**Application Design**

**Document Version: 1.2**

**Change History**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| 06/16/17 | 0.1 | Defined functions  Input / Output  Interfaces  Data Structures | Eric Sabelhaus, William Donabedian, Justin Wheeler |
| 06/17/17 | 0.2 | Problem analysis  Software definition  Definitions section | Eric Sabelhaus |
| 06/18/17 | 1.0 | Design Considerations  Functional Design | Eric Sabelhaus |
| 06/18/17 | 1.1 | Rewording  Added detail to interfaces | William Donabedian  Tyler Gibbs |
| 06/18/17 | 1.2 | Add table of contents | Eric Sabelhaus |
|  |  |  |  |

**Table of Contents**

1. Problem Analysis
2. Definitions
3. Design Considerations
4. Software Choices
5. Functional Design
6. Interfaces
7. Data Structures
8. Input formats
9. Output formats
10. Overall Design

1. Problem Analysis:
2. In computer science, a new topic of interest which focuses on data analysis with respect to big data, is the topic of natural language processing.
3. With natural language processing, there are multiple schools of focus, one of the primary techniques for processing and understanding big data is sentiment analysis.
4. Sentiment analysis seeks to provide a programmatic meaning of text, such that, it could help provide a better understanding of whether text is positive or negative in nature.
5. 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.
6. As of now, numerous existing libraries have been written to implement forms of sentiment analysis, but they’ve been designed as software packages which must be implemented within other software solutions to expose the functionality, and none have been made easily available to the average person via the internet.
7. This application seeks to provide such a solution to the average person, detailing the overall sentiment of a given body of text.
8. Definitions:
9. 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.
10. 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
11. Background Job – A secondary process, usually spawned from a parent process, or 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.
12. 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 aa rest endpoint which can be leveraged by other servers over HTTP, to perform various functions on data programmatically.
13. DOM – With respect to a web browser, this is the Document Object Model, which is used to define visual elements within a web browser.
14. Design Considerations:
15. Time Constraint:
16. 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.
17. Familiarity:
18. The team lead has prior experience designing and implementing software using the various open sources solutions which will be utilized on this application.
19. As such, the developers and technical writers can glean experience of enterprise application development and deployment due to that familiarity in a mentoring fashion.
20. Simplicity:
21. 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
22. Software Choices:
23. JavaScript / NodeJS:
24. The application will be written using JavaScript in the web server development framework NodeJS
25. Given the single threaded nature of JavaScript, the application will be written in a functional and asynchronous format
26. Any synchronous functions will be designed as background jobs to avoid blocking the main execution thread of the application
27. ExpressJS:
28. Within npmjs.com, there is a well-defined HTTP web server framework provided in the form of an open source module
29. This can be found here: <https://www.npmjs.com/package/express>
30. This plugin provides an extensible API for creating consistent web server architecture that utilizes the CRUD restful web server designs.
31. Sentiment plugin:
32. Within npmjs.com, there is a well-defined implementation of sentiment analysis provided in the form of an open source module
33. This can be found here: <https://www.npmjs.com/package/sentiment>
34. The plugin is extensible, and will be implemented as a background job to be run within node-resque
35. node-resque plugin:
36. Within npmjs.com, there is a well-defined implementation of background job queue provided in the form of an open source module
37. This can be found here: <https://www.npmjs.com/package/node-resque>
38. The plugin is extensible, and will provide a consistent background job system for implementing the worker which will perform the actual sentiment analysis
39. Redis server:
40. Redis is a well trusted object key storage server. It is used by millions of high throughput services which many of us use on a daily basis.
41. The Resque process utilizes Redis server as a back end for pop and shift functionality as jobs are added and worked from the queue, and since the sentiment analysis feedback app requires no persistent data storage, it is a great implementation to use redis, as it takes up very little memory to store a queue
42. Nginx:
43. Nginx is a widely used, and industry trusted web server
44. 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
45. This design strategy has been used by millions of websites to securely serve up their content over TLS with little effort, and great success
46. Functional Design
47. Functions:
48. Worker – Provides:
49. Background job – This will be the initialization of numerous function within the worker that comprehensively gather the sentiment analysis of all strings within an input
50. Some function to interact with sentiment
51. Some function to iterate over array of strings
52. Some function to wrap those and return data
53. Result of job – This will format and return the data which was created as a result of the completion of a background job.
54. Queue – Provides:
55. Push – Add a new job to the queue
56. Pop – Take a job off the queue to be worked
57. Router – Provides:
58. GET
59. This implements the endpoint to serve up the index page of the application
60. It will be the only graphical user interface endpoint of the application
61. This will be what is considered a single page app, as a result
62. POST
63. This implements the endpoint to accept input, and create a background job to process the web request for sentiment analysis
64. 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
65. Logger – Provides:
66. 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:
67. Debug – helpful for application developers and testers alike
68. Info – extensive information, good for some debugging by those who implement the web server in a production space to get more helpful information
69. 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
70. Client Side – Provides:
71. User interface leveraging HTML5 and CSS3 standards to design a simple user input form
72. JavaScript functionality to pre-process the text provided by a user into an array of strings
73. JavaScript AJAX request to the POST endpoint using JSON formatted array. This function will reformat the page on response from the server
74. Interfaces
75. Restful web interface
76. Allows requesting systems to access and manipulate textual representations of Web resources using a predefined set of stateless operations
77. Graphical User Interface (GUI)
78. Visual way of interacting with a computer using items such as windows, icons, and menus
79. Implemented using HTML5 and CSS3 standards
80. Data structures
81. Queue – The queue will be used by node-resque to push and pop jobs off the queue as resources become available to the application
82. Array – Arrays of strings will be utilized in the transport of data to and from the server and the client
83. String – Unicode encoded strings will be input and output for the application
84. Token – Strings will be tokenized by sentiment plugin into individual sentiments to be analyzed and measured.
85. Input formats
86. Array of Unicode String(s)
87. 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
88. There is a potential that users may input emoji icons
89. 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.
90. HTTP Web Requests:
91. All input to the application from external sources will be made over POST and GET http method requests.
92. The POST web request will be required to be in the following format:

1. Output formats
2. JSON Response to be returned from POST endpoint:

{

  “data”: [

    {

      “score”: 0,

      “line”: “this is a sentence”

    },

    {

      “score”: #,

      “line”: “line”

  ]

}

1. 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.
2. Overall Design
3. The design choices made for this application are intended to allow it to be run on a low resource server.
4. The ability to run this on a low resource server could enable 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.
5. 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.