Tag Cloud - cloudfive

Concept of Operations

COP4331C-001 Fall 2010

Modification history:

Version	Date	Who	Comment
v0.0	08/15/00	G. H. Walton	Template
v0.1	9/15/10	Pierre LaBorde	Template draft
v0.1.1	9/16/10	Pierre LaBorde	Typos corrected
v1.0	9/20/10	Pierre LaBorde	Template finalized

Team Name: cloudfive

Team Members:

- Hector Colon hectorhector@gmail.com http://hec.to/r/tagcloud.html
- Pierre LaBorde pierrelaborde@gmail.com http://eustis.eecs.ucf.edu/~pi135483
- Robert Bieber robby@bieberphoto.com http://www.bieberphoto.com/groupfive/bieber/
- Chris Orchard corchard@cfl.rr.com http://eustis.eecs.ucf.edu/~ch525234/
- Jason Hochreiter jasonhochreiter@gmail.com http://eustis.eecs.ucf.edu/~ja612957/
- Charles Allen chip@knights.ucf.edu http://allencharles.weebly.com/

Contents of this Document

The Current System

There is no current system.

The Proposed System: Needs

The proposed system needs to be a script that is written in such a way as to be able to be integrated into an existing web server with

little to no modification necessary to start the service. The output of this webservice should be displayed the same HTML page that has our server-side script embedded in it. Additionally, this output should be in an HTML format that makes use of no non-standard or proprietary packages or libraries. Due to the previous condition the HTML output should be able to be downloaded and/or copied by the user into his/her own webpage and embedded in his/her website.

The Proposed System: Users and Modes of Operation

Anyone with unstructured information would be able to organize it using our cloud tag.

Bloggers would find this organization feature particularly useful.

Websites that host blogs could provide this as a service to their users.

Generating the tag cloud would be a mode of operation in which the user's information is read in and the algorithm is applied that will then output a selection of the most frequently occurring words.

Displaying this data would be another mode of operation because the text size with which a particular word is displayed should vary based on the frequency.

Providing downloadable output would be another mode of operation. In this case, what is displayed on the screen as HTML should be provided to the user in the same format so that it may be directly embedded into the user's web page. If there is enough time for us to make our way down the list to the want to have section then we would be able to provide this downloadable output in Adobe PDF and/or Microsoft Word format.

The Proposed System: Operational Scenarios

The routine operational scenario of the system would require our script to be installed on a webserver and then displayed using HTML. Then the user would have to input text into a text box by either copy/paste or typing directly. When the user presses the submit button our script will take the unstructured information entered by the user and output a set of relevant words ranked by frequency. This ranking will be in the form of text size changes and perhaps even color, style, or orientation changes as well. This output will be available for the user to display on their own website in the form of HTML that could be downloaded and/or copy/pasted. If time permits, we may add other formats in which to download the information, i.e. Adobe PDF and/or Microsoft Word.

The singularly possible atypical scenario for our script would be an attempt by the user to input executable code into our input textbox. We can handle this by ensuring that the script does not accept any more data than it can handle perhaps with a character limit on the input textbox itself or another solution that is not visible to the user. Additionally, the executable code could be handled as plaintext and we would have successfully thwarted the malicious user's potential attack. However, this particular user may not be happy with his tag cloud.

The Proposed System: Operational Features

Must Have:

- Text box where user can paste input
- Return tag cloud
- HTML embed of tag cloud
- Change text size based on frequency
- Change colors based on frequency
- User-specified minimum word frequency

Minimum number of words to display in tag cloud

Would Like to Have:

Adobe Acrobat PDF export of tag cloud

Word Document export

User-specified maximum font size

User-specified font size algorithm

The Proposed System: Expected Impacts

User navigation of previously unstructured data would be greatly improved. This could lead to increased readership of a blog or other website implementing our proposed system. This is due to the increased ability (following improved navigation) to display attractive articles (those related to the current article) to the user. Another similar website that this could be useful for would be news websites. Additionally, this system could drive attention to advertisements targeted at the specific user.

The Proposed System: Analysis

Expected Improvements: There are no expected improvements because there is no current system.

Disadvantages: Plaintext is the only acceptable format for input

The plaintext cannot be entered indirectly either through API or a link to another website

Limitations: Time is a limitation because we have a deadline

Communication is even more limiting because it is hard to find a time when we can all meet

Risks: A team member could withdraw from the class

The server that our project website is hosted on could go down

Alternatives and Tradeoffs: Alternative to our approach we could have created a desktop application for the same project

In this case we trade a wide customer-base for a central customer-base and maintain a distributed user-base

Additionally, we could have created a smart phone application for our project

We are trading accessibility for simplicity in the case of smart phone development

---

Template created by G. Walton (GWalton@mail.ucf.edu) on August 30, 1999 and last modified on August 15, 2000.

This page last modified by Pierre LaBorde (pierrelaborde@mail.ucf.edu) on September 22, 2010

Tag Cloud

Project Management Plan

COP 4331C PROC OBJECT ORIENTED SOFTWARE - Fall 2010

Modification history:

Version	Date	Who	Comment
v0.0	08/15/00	G. H. Walton	Template
v1.0	9/21/2010	Hector Colon	Added Tools and Environment to Plan for tracking
v.1.01	9/22/2010	Robert Bieber	Added Project Overview
v.1.02	9/22/2010	Jason Hochreiter	Added Reference Docs to Software Life Cycle

Team Name: Group 5

Team Members:

- Hector Colon hectorhector@gmail.com http://hec.to/r/tagcloud.html
- Pierre LaBorde pierrelaborde@gmail.com http://eustis.eecs.ucf.edu/~pi135483
- Robert Bieber robby@bieberphoto.com http://www.bieberphoto.com/groupfive/bieber/
- Chris Orchard corchard@cfl.rr.com http://eustis.eecs.ucf.edu/~ch525234/
- Jason Hochreiter jasonhochreiter@gmail.com http://eustis.eecs.ucf.edu/~ja612957/
- Charles Allen chip@knights.ucf.edu http://allencharles.weebly.com/

Contents of this Document

Project Overview

Cloud Five is a tag cloud generator. The project will begin as a simple web application that allows users to enter some text into a text box along with settings for the tag cloud, and then generates a tag cloud from the entered text, offering the user the ability to embed that cloud into a web page. Later, the project may offer a public API, so that a script generating a web page could automatically query the service to generate a tag cloud based on its latest data.

Reference Documents

For more information, please see the Concept of Operations.

Applicable Standards

- Coding Standard - we will follow the Python Style Guide, available at http://www.python.org/dev/peps/pep-0008/. This enforces coding structure, such as indentation and naming conventions. Moreover, we will require that our code be sufficiently documented, with broad comments at the beginning of each file explaining its specifications, inputs, and outputs, and a smaller section of similar comments for each major class and function in the code.
- Document Standard - all documents will be composed in the Times New Roman font with a font size of 12 points. All headings will be bold. Sentences will be spaced normally, and an extra newline will separate paragraphs. Each document other than our individual websites will begin with a modification history - with entries consisting of an author, date, modification comments, and version number - followed by a table of contents; the table of contents in webpages will contain links to the corresponding sections. An automated spell checker will analyze our documents for spelling errors. The group members will verify grammar manually. HTML files will be created based on the templates provided at http://www.eecs.ucf.edu/courses/eel5881/ProjectTemplates.html.
- Artifact Size Metric Standard - we will focus on the number of bytes for our source code; in this way, we limit the amount of information that may need to be transmitted on the web. Moreover, we will restrict each source file to one major module with any number of submodules to facilitate development. Documents will be described by the number of lines and pages, for these metrics provide a quick summary of the length of a document.

Project Team Organization

The group consists of Chris Orchard: corchard@cfl.rr.com, http://eustis.eecs.ucf.edu/~ch525234/, Robert Bieber: robby@bieberphoto.com, http://bieberphoto.com/groupfive/bieber/, Pierre LaBorde: pierrelaborde@gmail.com, http://eustis.eecs.ucf.edu/~pi135483", Hector Colon: hectorhector@gmail.com, http://hec.to/r/tagcloud.html, Charles Allen: chip@knights.ucf.edu, http://allencharles.weebly.com/, and Jason Hochreiter: jasonhochreiter@gmail.com, http://eustis.eecs.ucf.edu/~ja612957/.

Chris will handle requirements elicitation and documentation. Robert will maintain the website, including checking the spelling and grammar of the documents, and create our slideshows. Pierre will handle project and artifact organization. Hector will oversee program configuration and documenting progress and future work. Charles will be

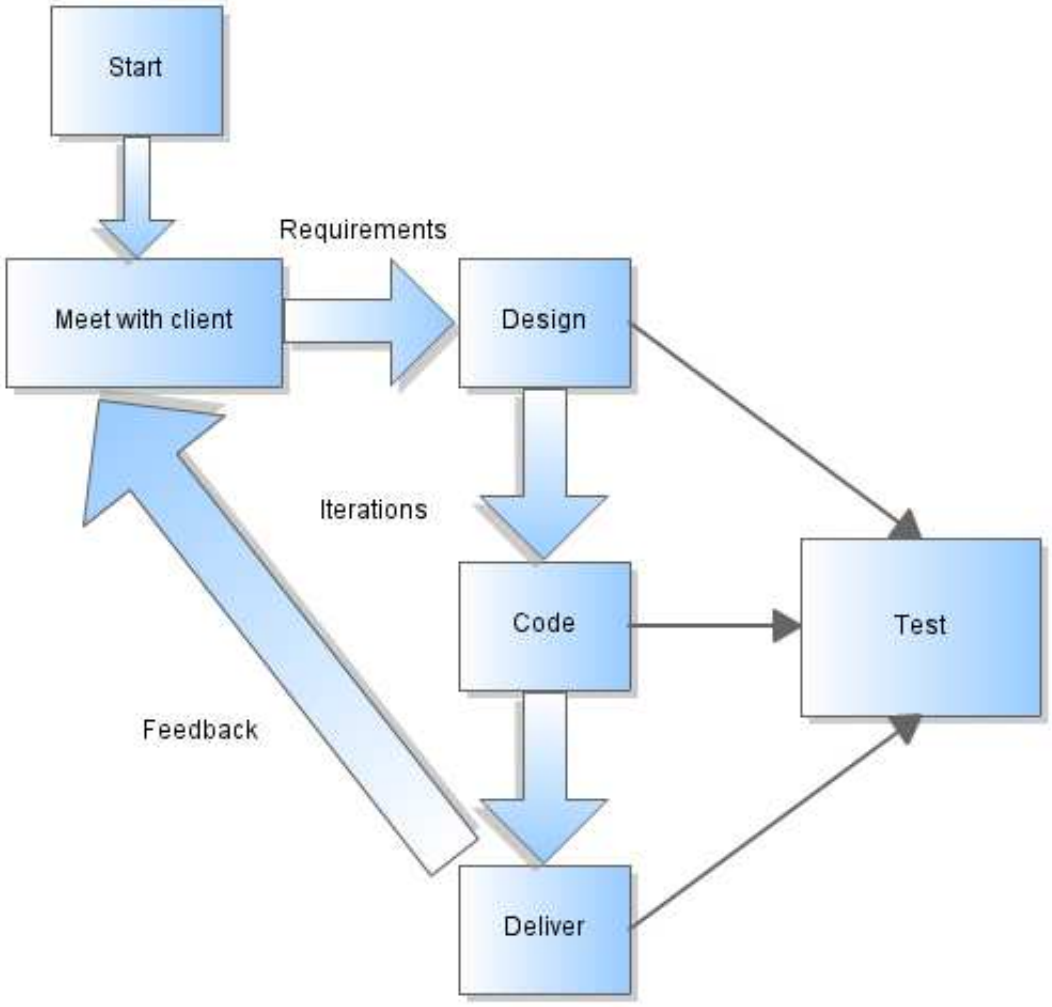
in charge of quality assurance and testing. Jason will be responsible for program design and training. All members will take part in programming tasks. Communication will typically be handled via email or IRC, though face-to-face meetings will be scheduled based on the availability of group members. In general, we will also be able to discuss the project before or after class.

Deliverables

Artifact	**Due Dates**
Meeting Minutes	11/30/10
Individual Logs	11/30/10
Group Project Management Reports	11/30/10
ConOps	9/23/10
Project Plan	9/23/10
SRS	9/23/10
High-Level Design	10/26/10
Detailed Design	10/26/10
Test Plan	9/23/10
User's Manual	11/30/10
Final Test Results	11/30/10
Source, Executable, Build Instructions	11/30/10
Project Legacy	11/30/10

Software Life Cycle Process

We will employ an agile model, for our customer has not revealed many desired features or functionality. This will allow us to implement prototypes quickly to demonstrate to our customer for review. We will begin by drafting plans based on an initial meeting with the customer. From there, we will begin developing prototypes in an incremental fashion, making changes to adhere to the customer's desires. We will try to involve the customer in the design process as much as possible to ensure that our software meets all expectations; at any stage in the process, we may revisit the planning stage to guide upcoming modifications. Testing will play a large role in our software life cycle: we will integrate testing into every phase of development.



Tools and Computing Environment

We will be using a mix of University resources, as well as our own. Programming will be done on Windows as well as Linux machines. The application will be implemented as a web app written in Python. We will be using Eclipse with the PyDev plugin and Notepad++ as our IDEs. Firefox and Google Chome will be used for viewing and testing the application.

Configuration Management

Gitorious, at gitorious.org will be used for source control. Automatic source control will be used by our IDEs, utilizing the git protocol. Documentation will also be source controlled on our gitorious project page at http://gitorious.org/cloud-five. Robert Bieber will be in charge of source control and is the owner of the gitorious project page. Project members will have well defined roles and tasks to prevent any problems with concurrent editing of the same section of code. Also, the object oriented style of programming we will use will allow members to work on seperate parts of code without changes causing errors in unrelated sections.

Quality Assurance

Quality assurance will be managed by Charles Allen. It will be performed at every milestone of the project and at the end of the project using preplanned test cases. Quality assurance will be performed by each member of the group, and results will be shared via email and the gitorious project page.

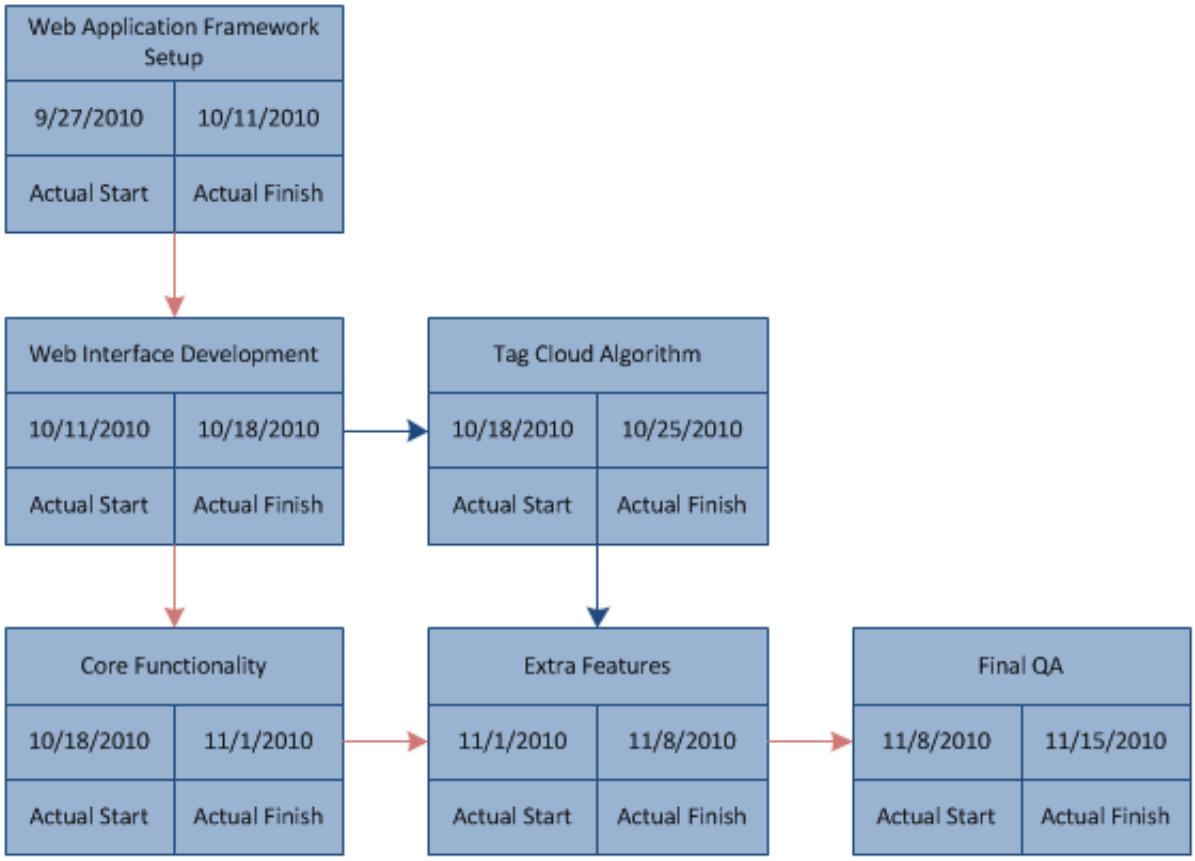
Risk Management

There are several risks in undertaking this project. First, is our inexperience with python could lead to some unforseen complications with our project. Also, we have relatively less experience with making web applications. There is also the risk of our group's orginizational style which is unstructured with most communication occuring online. While it may be a risk, it will hopefully benefit us as we will be able to maintaine agile through the development of the project.

Table of Work Packages, Time Estimates, and Assignments

- Web Application Framework setup - 2 weeks
- Interface Development - 1 week
- Tag Cloud Algorithm and Core Functionality - 1 week
- Extra Features - 1 week.

PERT Chart



Technical Progress Metrics

Requirements

- Number of base requirements: 5/5
- Number of extra requirements: 0/5

Design

- Number of UML Diagrams: 0/7

Code

- Number of classes: 0/3

- Number of methods: 0/20

Testing

- Number of test cases: 10/10
- Number of test cases passed: 0/10
- Number of bugs: 0/20

Plan for tracking, control, and reporting of progress

Team members will update their web pages and email their progress to the rest of the group weekly. Schedules and assignments will be coordinated through email. Any updates will be reflected on the project planning web page. Progress will be evaluated every week and each member is responsible for keeping on schedule.

Template created by G. Walton (GWalton@mail.ucf.edu) on Aug 30, 1999 and last updated Aug 15, 2000

This page last modified by Jason Hochreiter (jasonhochreiter@gmail.com) on September 23, 2010

Tag Cloud - cloudfive

Software Requirements Specification

COP4331C-001 Fall 2010

Modification history:

Version	Date	Who	Comment
v0.0	8/15/00	G. H. Walton	Template
v1.0	September 23, 2010	Chris Orchard	Initial Release

Team Name: cloudfive

Team Members:

- Hector Colon hectorhector@gmail.com http://hec.to/r/tagcloud.html
- Pierre LaBorde pierrelaborde@gmail.com http://eustis.eecs.ucf.edu/~pi135483
- Robert Bieber robby@bieberphoto.com http://www.bieberphoto.com/groupfive/bieber/
- Chris Orchard corchard@cfl.rr.com http://eustis.eecs.ucf.edu/~ch525234/
- Jason Hochreiter jasonhochreiter@gmail.com http://eustis.eecs.ucf.edu/~ja612957/
- Charles Allen chip@knights.ucf.edu http://allencharles.weebly.com/

Contents of this Document

Supporting Material

SECTION 1: Introduction

Software to be Produced:

- The team will produce software to generate a tag cloud in Python. A tag cloud is a visual display of the most frequently appearing words in a document; the font size and color of each word in the cloud will be affected by its frequency in the document. The user will be able to paste in text on a website in order to have it analyzed. A customizable "stoplist" will contain words that will be ignored by the software. Please see the Reference Documents for more information.

Reference Documents:

- Concept of Operations: http://www.bieberphoto.com/groupfive/conops.php
- Project Plan http://www.bieberphoto.com/groupfive/management.php

Applicable Standards:

- In order to run the software, users must have an active internet connection and an up-to-date web browser such as Mozilla Firefox or Google Chrome. Furthermore, the users must have installed Python runtime libraries. The corresponding requirements for these pieces of software are sufficient for running the tag cloud.

Definitions, Acronyms, and Abbreviations:

- None;

SECTION 2: Product Overview

Assumptions:

- Web hosting company will maintain all hardware
- Users have proficiency in web applications, text editing and manipulation, basic cut/copy/paste operations, and standard user input devices (mouse, keyboard, touch screen, etc.)
- There are no COTS modules available for our specific application

Stakeholders:

- Dr. Turgut and Himanshu Pagey are the customers of the system and are interested in seeing all of us successfully develop this project.
- Every member of our team is a stakeholder because we are all part of the development team and want the project to succeed so that we will do well in the class.
- Our potential users are also stakeholders because if the system succeeds then they will gain some new functionality that they were seeking.

Event Table:

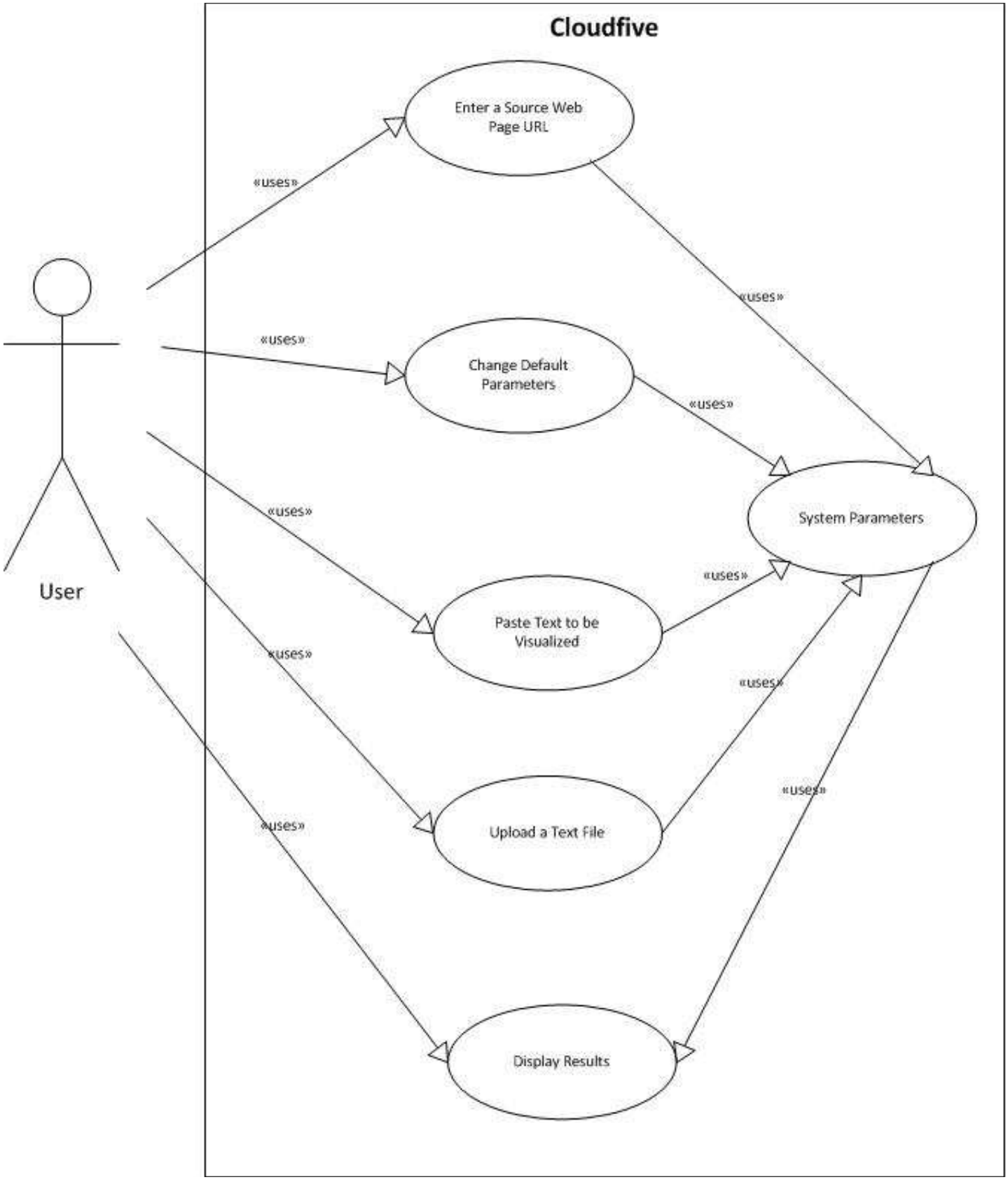
- <An event table identifies all the external events to which the software must respond. This is a first step in determining the required overall system functionality. The event list should be consistent with the context diagram and the interest of each stakeholder. Make sure that exceptions are considered.>
- <Use the following table format:>

Event Name	External Stimuli	External Responses	Internal data and state
Run *Cloudfive* application	User opens Web page containing the *Cloudfive* application	An instance of the *Cloudfive* application starts running and displays the input options / default start page to the user	Waiting for input

Select maximum number of words to show	User clicks inside of the "Maximum Number of Words to Show" box and enters desired integer	Number displayed in "Maximum Number of Words to Show" box changes to user-entered number	"Maximum Number of Words to Show" variable is set to the value entered by the user. Waiting for input
Select minimum frequency of the words to be included in the output	User clicks inside of the "Minimum Frequency" box and enters desired integer	Number displayed in "Minimum Frequency" box changes to user entered number	"Minimum Frequency" variable is set to the value entered by the user. Waiting for input
Select whether or not to display the word count next to the words in the output display	User clicks either on the "yes" or "no" dot next to "Show Frequencies"	The "dot" selected is filled with a smaller black dot to indicate the selection	"Display Frequencies" variable is set to the user selection. Waiting for input
Select whether or not to group similar words together in the output	User clicks either on the "yes" or "no" dot next to "Group Similar Words"	The "dot" selected is filled with a smaller black dot to indicate the selection	"Group Similar Words" variable is set to the user selection. Waiting for input
URL as Source	User clicks inside the "Web Page URL" box and enters a URL for the text source	URL is echoed to the screen in the box where it was entered	Waiting for input
File as Source1	User clicks inside the "Upload a File" box and enters a complete path and file name for the file to upload	path and file name is echoed to the screen in the box where it was entered	Waiting for input
File as Source2	User clicks on the "Browse" button next to the "Upload a File" box, navigates to desired file, and selects file to upload	path and file name is echoed to the screen in the box where it was entered	Waiting for input
Text as Source	User clicks inside the "Paste Text to be Visualized" box and pastes previously cut or copied text into the "Paste Text to be Visualized" box	Pasted text is echoed to the screen in the box where it was pasted	Waiting for input
Display Results	User clicks the "Display Results" button	System reads input source, calculates output, and sends result to the display	Input source variable set to user-defined value, tag cloud calculated and sent to output routine. Waiting for input

Use Case Diagram:

-



Use Case Descriptions:

- Modify Default Parameters
    1. Use case initiated by user when user clicks on an option to change
        1. Select the maximum number of words to show
        2. Select the minimum frequency of the words to be included in the output
        3. Select whether or not to display the frequency next to the words in the output display

4. Select whether or not to group similar words together in the output
2. After changing parameters, the new parameters will be used as the input to the “Display Results” use case

- Enter a Source Web Page URL
    1. Use case initiated by user when user clicks inside the “Web Page URL” box
        1. User enters a URL for the web page containing the text for the input to the tag cloud generator
        2. System reads in text from URL entered by user
        3. Text is used as input to the “Display Results” use case
- Upload a Text File
    1. Use case initiated by user when user clicks inside the “Upload a File” box or clicks on the “Browse” button to search for a file
        1. User clicks in the “Upload a File” box and enters a complete path and file name to upload
            1. system reads file and the text is used as input to the “Display Results” use case
        2. User clicks on the “Browse” button to search for a file
            1. User searches for desired text file to use for input and clicks on the file name in the window
            2. System reads file and the text is used as input to the “Display Results” use case
- Paste Text to be Visualized
    1. Use case initiated by user when user clicks inside the “Paste Text to be Visualized” box
        1. User pastes previously cut or copied text into the “Paste Text to be Visualized” box
            1. System reads text and uses as input to the “Display Results” use case
- Display Results
    1. Use case initiated by user when user clicks the “Display Results” button
        1. When user clicks the “Display Results” button, the system uses the default parameters or the parameters chosen in the “Modify Default Parameters” use case to calculate the resulting tag cloud and display the result to the screen

---

SECTION 3: Specific Requirements

<The following template must be used for <u>each</u> requirement: >

<u>3.1 Functional Requirements</u>:

No: 1
Statement: The system shall display the Tag Cloud result within 5 seconds of clicking on the "Display Results" button.
Source: Customer requirement
Dependency: None
Conflicts: None
Supporting Materials: None
Evaluation Method: Use system timer to calculate the response time for 30 test cases. Calculate the average response time.
Revision History: Chris Orchard, 09-22-10, initial release

No: 7
Statement: The system shall differentiate words with higher frequencies by font size on the tag cloud display.

Source: Customer requirement
Dependency: None
Conflicts: None
Supporting Materials: None
Evaluation Method: 10 Test cases for each input format. Set "Display Frequencies" input parameter to "yes". Verify that the font sizes properly correspond to the frequencies of the words displayed in the tag cloud output.
Revision History: Chris Orchard, 09-22-10, initial release

-

3.2 Interface Requirements:

No: 2
Statement: The system shall accept input text in three different formats. 1) Upload a file, 2) Pasted text that was previously copied, 3) A web page URL pointing to the input text
Source: Customer requirement
Dependency: None
Conflicts: None
Supporting Materials: None
Evaluation Method: 10 Test cases for each input format. Verify that tag cloud output matches known test cases.
Revision History: Chris Orchard, 09-22-10, initial release

No: 3
Statement: The system shall allow the user to enter the "Maximum Number of Words" to display in the tag cloud output.
Source: Customer requirement
Dependency: None
Conflicts: None
Supporting Materials: None
Evaluation Method: 10 Test cases for each input format. Each input case must have a greater number of words than the "Maximum Number of Words" input parameter. Count number of words in tag cloud output to verify that it matches the "Maximum Number of Words" input parameter.
Revision History: Chris Orchard, 09-22-10, initial release

No: 4
Statement: The system shall allow the user to enter the "Minimum Frequency" of words to display in the tag cloud output.
Source: Customer requirement
Dependency: None
Conflicts: None
Supporting Materials: None
Evaluation Method: 10 Test cases for each input format. Each input case must have select words placed throughout the text that match the "Minimum Frequency" input parameter. Count number of words in tag cloud output with minimum frequency to verify that it matches the "Minimum Frequency" input parameter.
Revision History: Chris Orchard, 09-22-10, initial release

No: 5
Statement: The system shall allow the user to select whether or not to display the frequency next to the words in the cloud tag output.
Source: Customer requirement
Dependency: None
Conflicts: None
Supporting Materials: None
Evaluation Method: 10 Test cases for each input format. Each input case must have at least 20 words with known frequency. Verify that the frequency numbers displayed in the tag cloud output match the known frequencies when "Show Frequencies" input parameter is set to "yes".
Revision History: Chris Orchard, 09-22-10, initial release

No: 6
Statement: The system shall allow the user to select whether or not to group similar words together in the cloud tag output.
Source: Customer requirement
Dependency: None
Conflicts: None

Supporting Materials: None
Evaluation Method: 10 Test cases for each input format. Each input case must contain at least 10 different sets of similar words. Verify similar word grouping in tag cloud output when "Group Similar Words" input parameter is set to "yes".
Revision History: Chris Orchard, 09-22-10, initial release

3.3Physical Environment Requirements:

No: 8
Statement: The application shall run on any web server
Source: Customer requirement
Dependency: None
Conflicts: None
Supporting Materials: None
Evaluation Method: Run application on UCF server and verify proper operation, run application on Team web server and run all previously defined system tests.
Revision History: Chris Orchard, 09-22-10, initial release

3.4 Users and Human Factors Requirements:

- The system supports only one type of user, a website owner who desires a tag cloud to embed in their pages.
- The typical user is expected to have a basic knowledge of the workings of the Internet, given that they operate a website themselves. A guide will be given for the basic operation of the tag cloud application, but no specific users will need to be trained, as this project will operate as a public website. In later revisions, the project may support a public API, in which case the user will also have access to documentation for the API's interfaces.
- Disabled users may need to use the product with assistive technologies, so standard web accessibility guidelines will be followed in development.
- The system must detect users attempting to misuse the system by crashing it or accessing other users' data by carefully enforcing escaping of input and monitoring the size of all input to avoid overflowing buffers.

3.5 Documentation Requirements:

- Our project will require only on-line documentation.
- As a public web service, there is no physically available client to provide paper documentation to. On-line documentation will be available to all users of the service.
- The audience for the documentation consists of other web developers, so it is expected that they understand the basic workings of web sites and their development.
- The project may support a public API in later revisions, in which case both a guide to using the web interface and a detailed description of the public API will be required. The public API description may be auto-generated using a tool like pydoc.

3.6 Data Requirements:

- Font size will be calculated using the formula referenced in Supporting Materials. Font size must be calculated to the nearest point-size.
- For security and privacy purposes, no data should be retained under any circumstances. The product will produce output based solely on the input provided during a single use.

3.7 Resource Requirements:

- Web hosting for the project provided by Robert Bieber. The hosting company attends to all the physical and personnel resources needed to keep the site running.
- No additional funding is needed.
- The project will be developed using group members' computers and a free software web stack (Apache server and the Python programming language).

3.8 Security Requirements:

- Access to the system will be public and uncontrolled.
- All users' data will be isolated inherently because none of it will be stored: all user data is lost after serving each request.
- The system will not need backups because it does not store data.
- Physical precautions are the domain of the hosting company.

3.9 Quality Assurance Requirements:

- The product must be able to accept any input without crashing. If input is out of acceptable boundaries, that condition should be detected and reported as an error to the user.
- The product should stand up to thorough testing of out-of-bounds inputs.
- The system should detect and reject out-of-bounds or otherwise illegal inputs.
- Availability and response times will all be subject to the web host's systems.

SECTION 4: Supporting Material

- Formula for font scaling

Template created by G. Walton (GWalton@mail.ucf.edu) on Aug 30, 1999 and last updated Aug 15, 2000

This page last modified by Jason Hochreiter (jasonhochreiter@gmail.com) on September 22, 2010;

Tag Cloud - cloudfive

Test Plan

COP4331C-001 Fall 2010

Modification history:

Version	Date	Who	Comment
v0.0	08/15/00	G. H. Walton	Template
v0.1	9/15/10	Pierre LaBorde	Template draft
v0.1.1	9/16/10	Pierre LaBorde	Typos corrected
v1.0	9/20/10	Pierre LaBorde	Template finalized

Team Name: cloudfive

Team Members:

- Hector Colon hectorhector@gmail.com http://hec.to/r/tagcloud.html
- Pierre LaBorde pierrelaborde@gmail.com http://eustis.eecs.ucf.edu/~pi135483
- Robert Bieber robby@bieberphoto.com http://www.bieberphoto.com/groupfive/bieber/
- Chris Orchard corchard@cfl.rr.com http://eustis.eecs.ucf.edu/~ch525234/
- Jason Hochreiter jasonhochreiter@gmail.com http://eustis.eecs.ucf.edu/~ja612957/
- Charles Allen chip@knights.ucf.edu http://allencharles.weebly.com/

Contents of this Document

SECTION 1: Introduction

- The goal of the testing phase will be to ensure that the software functions as intended, is efficient, and is easy to use. Efficient will be defined as functioning across the range of accepted input, from 0 words to a maximum that will not exceed one hundred thousand words and will be determined at a later date by the customer. Easy to use will be defined after a prototype is presented to the customer, before this the interface will be modeled off of an example site that the customer referenced - http://tagcrowd.com/./LI>

Reference Documents:

- Concept of Operations http://www.bieberphoto.com/groupfive/conops.php
- Project Plan http://www.bieberphoto.com/groupfive/management.php
- SRS http://www.bieberphoto.com/groupfive/srs.php

SECTION 2: Description of Test Environment

The software is intended to function as a webpage and thus should be cross-platform. It will specifically be primarily tested on windows 7 but the later stages of testing will see to ensuring that it functions on linux and mac os as well. The developers will do the majority of the functionality and efficiency testing. The customer will be responsible for clearly defining ease of use during the first prototype phase of testing.

SECTION 3: Stopping Criteria

Any errors found during a testing session will be written down and testing will continue until a critical error is found, or until the testers have exhausted all primary tests described below as well as testing random input from news sites as part of stress testing there will be no checked "right" answer for the random test cases. This will repeat until no errors are found in the primary test cases. If no errors are discovered during the primary test cases, then the main part of stress testing will occur, including simultaineous access with large input from multiple users at aproximently the same time. Stress testing will proceed for no more than one hour. "Good enough to deliver" will be defined as passing all ideal condition tests, while still functioning properly under stress testing.

SECTION 4: Description of Individual Test Cases

Test File Links: KnownNumbers.txt : http://www.weebly.com/uploads/5/2/8/1/5281011/knownnumbers.txt
AllEqualTestCase.txt : http://www.weebly.com/uploads/5/2/8/1/5281011/allequaltestcase.txt
AllEqualTestCase2.txt : http://www.weebly.com/uploads/5/2/8/1/5281011/allequaltest2.txt
MinimumTestCase.txt : http://www.weebly.com/uploads/5/2/8/1/5281011/minimumtestcase.txt
MaximumTestCase.txt : http://www.weebly.com/uploads/5/2/8/1/5281011/maxtestcase.in

Test 1:

- To test that the program properly calculates the number of occurences of a word
- We will use the file KnownNumbers.txt
- This will initially be run on windows before the program goes online. It will also be used at all later stages of development.
- The expected results for all 3 tests that will be run using this file are explained in the file itself

Test 2:

- To test that the program properly handles the minimum test case without crashing.
- We will use MinimumTestCase.txt for this.
- This will initially be run on windows before the program goes online. It will also be used at all later stages of development.
- The expected output is simply the single word that is in the file.

Test 3:

- To test that the program properly handles the maximum test case without crashing or running for more than five seconds without any output.
- We will use MaximumTestCase.in for this.
- This will be run while still on windows and before the customer decides on the maximum case they would like to implement for the final project. This test case will be altered to fit those specifications later and then rerun when the system is online.
- The test is to ensure that the program properly handles large inputs, so the input was randomly generated. The output of this test will be checked based on whether or not it outputs at all, and how quickly it runs.

Test 4:

- To test that the program properly handles a tie in word frequency, such as multiple words appearing the same number of times.
- We will use AllEqualTest.txt and AllEqualTest2.txt for this.
- This will be run on windows before the system goes online, it will also be checked at each major stage of development to ensure the program is still functioning properly.
- In these tests all words occur the same number of times, so the program should output them with the same font size, and if the tester puts a bound on the number, n, to be output before running it that is less than the total number of words in the test case then it should list the first n alphabetically ordered words should be listed.

Test 5:

- To test that the program properly outputs only words that occur more than or equal times to a number specified by the user at run time.
- We will use KnownNumbers.txt for this.
- This will be run on windows before the system goes online, it will also be checked at each major stage of development to ensure the program is still functioning properly.
- Output for this case is described in the file as the second test to be run with this file.

Test 6:

- To test that the program properly outputs only words that do not exist on the stop list as defined by the user.
- We will use KnownNumbers.txt and MaximumTestCase.in for this.
- This will be run on windows before going online once, then the KnownNumbers case will be run at each major stage of development. The MaximumTestCase will be run once, and after the customer defines the maximum case he wishes to use, we will then trim our test case to that specification and run that test at each major stage of devealopement.
  - This should output only words that do not exist on the stop list.

Template created by G. Walton (GWalton@mail.ucf.edu) on March 28, 1999 and last modified on August 15, 2000.

This page last modified by Jason Hochreiter (jasonhochreiter@gmail.com) on September 22, 2010