Project Development and Deployment Readiness Assessment Checklist

For Assessment of: [WASC ESSAY # 2]	
Agency/Client Name	Dr. Anne Marie Todd
Project Name	WASC Essay#2
Phase III Release Date	NA (No client till October 26, 2012)
Phase IV Release Date	PHASE IV/November 12, 2012
Phase V Release Date	PHASE V/ November 25, 2012

Project Development and Deployment Readiness Assessment Checklist

Criteria (Please Answer All Below)	Yes / No / NA
a. Are system requirements documented?	Yes
b. Have system requirements been reviewed and approved by the designated approvers?	No
c. Has the system design been reviewed and approved by the designated approvers?	No
d. Are software requirements documented?	Yes
e. Have software requirements been reviewed and approved by the designated approvers?	No
f. Has the website design been reviewed and approved by the designated approvers?	No
g. Is there a Requirements Traceability Matrix indicating traceability between requirements, design, and testing?	No
h. Do test planning documents that describe the overall planning efforts and test approach exist?	Yes
i. Is testing, as specified in the test planning documents, complete?	No
j. Are test results documented?	Yes
k. Is product/website defect-free?	No
l. Have all remaining defects been documented?	Yes
m. Is product acceptance sign-off (e.g., Final Acceptance) complete?	No
n. Is the product in compliance with documented security standards?	No
o. Have security activities been implemented or completed?	No
p. Have planned configuration audits been executed?	No
q. Have configuration audit results been documented?	Yes
r. Have planned data creation/conversion activities been executed, or are they on schedule to be completed as planned?	Yes
s. Have planned training activities been executed, or are they on schedule to be completed as planned?	NA
t. Are documents to be produced for the purpose of aiding in installation, support, or use of the product complete, published, and distributed, or are they on schedule to be completed, published, and distributed prior to deployment?	Yes
u. Are transition to support activities complete, or are they on schedule to be completed as planned?	No
v. Are activities for notifying stakeholders (clients/administration) of the release on schedule to be completed as planned?	Yes
w. Are activities to enable the operation and maintenance of the product on schedule to be completed as planned?	Yes
x. Have site preparation activities been completed?	Yes
y. Have environment preparation activities (e.g., correct OS, memory, etc.) been completed?	Yes
z. Is the selected software technology for the project listed on the enterprise's technology catalog, or has the appropriate authority approved the exception?	NA
aa. If the project requires purchased application software products, are all license agreements complete?	NA
bb. If the project requires purchased application software products, are all maintenance agreements in place and documented?	NA
cc. If the project requires purchased software products, have those items been installed in the production environment and tested?	NA
dd. Is the selected hardware technology for the project listed on the enterprise's technology catalog, or has the appropriate authority approved an exception?	NA

Project Development and Deployment Readiness Assessment Checklist

ee. If the project requires purchased hardware products, have those items been installed and tested?	NA
ff. If the project requires purchased hardware products, has all base application software been installed and tested?	NA
gg. If the project requires purchased hardware products, are all maintenance agreements in place and documented?	NA
hh. Is the production environment staged and prepared for release of the product for operational use?	No

Using This Template

To create a deliverable from this template:

1. Replace [bracketed text] in the tool header area at the top of page i (Contents page) with the same project and agency information as on the cover page.

   Note: Please do not remove or modify content in the footer area.

2. Complete the entire template. Each section contains abbreviated instructions, shown in italics, and a content area. The content area is marked with a placeholder symbol (⇒) or with a table. Relevant text from other project deliverables may be pasted into content areas.

   Note: Please do not remove the italicized instructions.

[Dr. Anne Marie Todd/SJSU WASC]

[WASC ESSAY No. 2]

VERSION: [4]	REVISION DATE: [11/28/2012]
VERSION: [3]	REVISION DATE: [11/23/2012]
VERSION: [2]	REVISION DATE: [11/15/2012]
VERSION: [1]	REVISION DATE: [11/08/2012]

Approver Name	Title	Signature	Date
Dr. Anne Marie Todd	*Professor,Communication Studies*		11/15/12

Note: This will be signed off by November 30, 2012

Blank Page

-Ezekiel Calubaquib

Section 1. Overview

1.1 Purpose

Identify the purpose of the Deployment Plan and its intended audience.

The purpose of the development plan are the following:

- Lay out the entire plan and steps for WASC Essay No. 2 project development.
- Explain the development of a fully functional java app that scans program plan.
- Explain WASC website structure
- Keep track of the dependencies and implementations added or discarded.
- Clarify and inform steps to client and to the subsequent CS100W Spring team.
- Instructions to maintain the website
- Includes all codes and resources used in the project

1.2 Business Context

Describe the business process that will be modified as a result of the deployment specified in the Deployment Plan.

The initial business process is to determine how to make a program scanner proposed by Dr. Todd. This proposed scanner should be able to read program plan documents. The documents would then be given a score for each of the 5 proficiencies based on the keywords (ref keywords.txt – see appendices). Results would then be outputted.

The second business process is to make the WASC Project website. This website would be shared with WASC Lit Review team—a separate adjunct project not done by our team.

The website would house the program scanner application together with all the documentation and references used for the app. In a foreseeable future the scanner would be available to be used by any type of educational institutions.

Business process that will be modified are the following:

- The use of Google Apps (ie. Google app engine vs Google drive)
- Updating program plan scanner (java app) to the needs of the client
- Updating keywords use to asses each program plan

- Scoring of each program plan basis (Statistic equation used)
- Updating website and its contents
- Increasing organization, communication, and delegation of work to team.
- Removing unnecessary and unimplemented objectives
- Necessary but unimplemented objectives documented for future team (ex. Google Analytics)

[Ezekiel Calubaquib]

1.3 Summary

Provide a summary of the Deployment Plan. Include an overview of activities necessary to get the website into a production environment such as installation, configuration, and initial operational activities.

The following are necessary steps in order to clarify the project, divide work, delegate task, develop and design website, and releasing the final product.

1. Meet with client and define project
    - Define 5 proficiencies

      - what they are / what are they used for / how the group could use them to grade a program plan/ why are they important?

    - Define what program plan scanner is
        - Define what program plan scanner (java app) should do
        - Define needs for the type of visual and result outputted
        - Define keywords / synonyms of keywords to be used
2. Submit project analysis (Phase I documents – Project Analysis given by Akshat: Phase 1 managers) to client in order to clarify clients needs and want.
3. Document Phase I Client Action Plan

   - List of the overview of activities/”sprints” hypothetical proposal to be performed from

     design to deployment of website. These include:

        - Website functions (ie. Information about program plan scanner)
        - Research to enable students to make the program plan scanner. These includes research in:

- Google Analytics, Google Drive, Java, Parsing PDF to Text (iText), Jquery, Json, Javascript, InDesign, Google App Engine.

- Project development and testing.
  - App Development
    - Making the scanner using Javascript (see Appendices for all codes)
      - Keyword Analyzer class capable of performing keyword statistical analysis on text. Calling parseText multiple times without a reset will continuously build word counts. Also, multiple keyword files can be read into the tree.
      - Pdf Extract class capable of translating pdf files into text files
      - Prefix Tree class to store keywords. Keywords have their corresponding weights, rubric, and how many times they have been found stored in the leaf nodes
      - WASC Engine Servlet class which is going to be the "main" class for the App Engine. It is going to listen on a certain port for an Http Request and should receive a pdf file. It is then going to be doing the work in order to figure out the appropriate rubric scores and return the final result as a JSON object.
  - App Testing
    - Includes testing of keywords for the program plan scanner (java app)
      - if it reads large program plan files
      - if it reads large set of keywords
    - Includes testing parsing pdf to txt using open source “iText”
    - Includes testing server (Google app engine)
    - Testing all website links and upload files
    - Testing most departmental program plan given by Professor Debra
- Website Launch
  - Including poster image file in front page

- Format output from .json file outputted by program plan scanner (java app)
- Revamp new links
- Add all documents from phase I-5 online and test
- Test for error / broken links

4. Plan and do WBS (Approval needed by professor and team)

   - Work breakdown sheet. A chart made using bubbl.us for an overview of all task needed to be done by each members of the group (see Appendices)

5. Document Phase II Design Worksheet

   - Purpose to lay out the real planned objectives to perform to successfully deploy website. (See Phase 2 design worksheet).

6. Delegate Task
   1. Front End (Website and Interface)
      1. Design and implement web layout in OUCampus
      2. Upload all documentation in OUCampus
      3. Treat .json file to be readable
   2. Back End (Java App & Server)
      1. Obtain Keywords (Given by client and additional keywords provided by team and approved by client)
      2. 5 proficiencies / Rubrics (Given by client)
      3. Calculate using an equation for scores/weight of variables for the 5 proficiency (provided by statistic major students)
      4. Make algorithm / java application that reads in pdf, counts keyword and score each program plan document (Java App & iText)
      5. Use a server to store the java app(Google App Engine)

         – Successful incorporation to server gives a link wascengine.appspot.com which could be embedded in OUCampus website.

7. Embed program plan scanner (java app) tool in website using iframe (OUCampus)
8. Test online application/ upload files in OUCampus
9. Finish all Phase I-II documentation updates to be uploaded in OUCampus
10. Release tool with website for full deployment
11. Client approval / Sign off

[Ezekiel Calubaquib]

Section 2. Assumptions, Dependencies, Constraints

2.1 Assumptions

Describe the assumptions about the current capabilities and use of the website when it is released live.

The website would have a place for users to:

- Describe WASC Essay Project both Essay No. 2 and Lit Review
  - About to page section of website
- Have links for the pertinent 5 proficiencies / rubrics of WASC
  - 5 proficiencies page section to describe each rubrics and a link for people to download each files in pdf format
- Provide links, documents (Phase I-V), and resources used in making the tool
  - Documentation page section of the website under WASC Essay no.2 project section (Documentation page is shared with Lit Review adjunct project from other team)
- Have a page for the java app scanner where user inputs files manually.
  - Program plan scanner section of the website alpha stage
- Show scores of the program plan document scanned without graphs
  - Program plan scanner section of the website would show result in same page whenever a file or files are submitted.
- Provide link or info about the client and names of people who contributed
  - WASC project section to describe Essay No. 2 project which includes client's page.

2.2 Dependencies

Describe the dependencies that can affect the deployment of the website.

Dependencies that can hinder or slows process of deployment are:

- Dependent on using Google Apps or any type of open source.
- OUCampus template based and limitation with iframe html
- OUCampus permission/access block which constrains manipulation of website
- Google App Engine dependent (server where the java scanner was placed)

  - public server which can be discontinued by Google (not in our control...ex. Yahoo Geocities discontinued by Yahoo)
- External Jars (Java) for using iText (Pdf to Text)
- GSON plugin for implementing json.
- Program plan scanner dependent on reading pdf files only
- Program plan scanner dependent on keywords approved by Dr. Todd. (Must need case study for future groups to show why those keywords must be used)
- Partnering with Lit Review team (One website for both teams)
  - Sign off/on problem (ex .if a member from Lit Review team (other group) forget to sign off (light bulb still yellow for editing page) in OUCampus, Essay No.2 team wont be able to update the page that is still signed on from the other group. Therefore, both teams need to depend and work with each other.
- Unavoidable change of plans
  - Change of needs / Research fails / Implementation fails
- Broken website links
- Time management
  - Students time availability and willingness to contribute / Group meeting

2.3 Constraints

Describe factors that limit the ability to deploy the website.

- Change of clients needs / Deviation from original plans

- ex. Google Drive supposed to be used for storage and OCR but the group used Google App Engine and iText instead.

- Java app scanner failure (one wrong line/text in code can cause whole program to crash)
- Clients availability
  - Time free / office hours or availability for the group
  - Appointment to clarify or adjust project
  - Contact through email
- Working with Lit Review on OUCampus (broken links / forgetting to sign off—stated above in dependencies)
- Reliance with Google app engine (has its limits ex. server crash)

[Ezekiel Calubaquib]

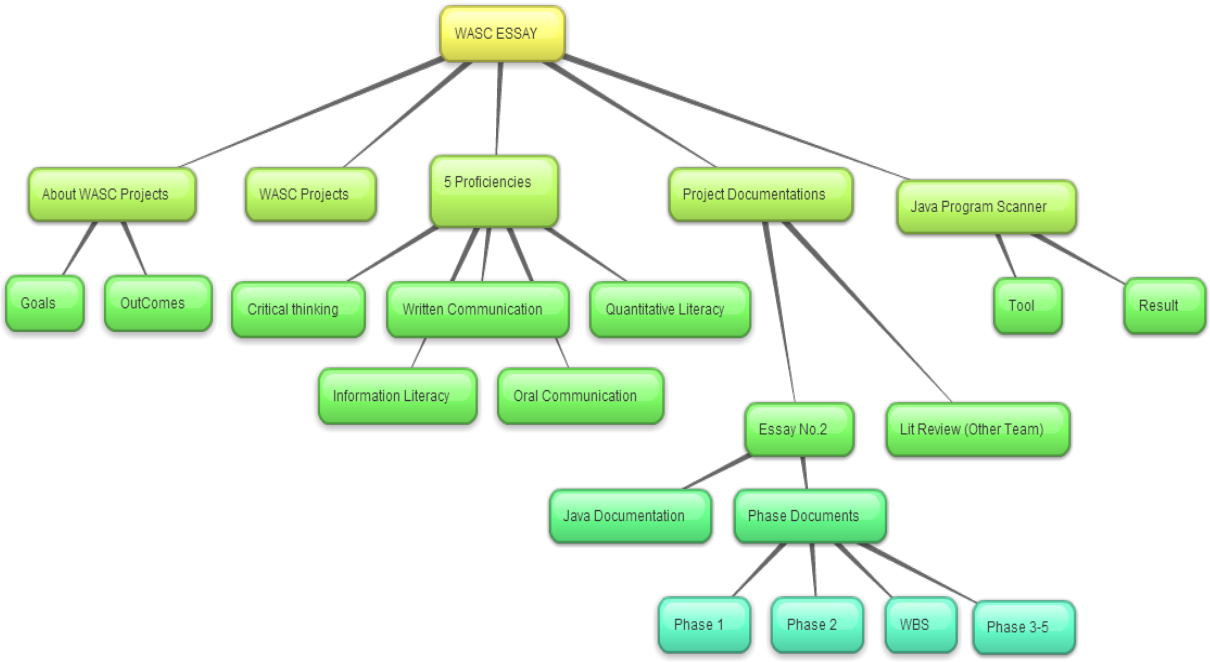
Section 3. Operational Readiness

3.1 Deployment Diagram

Provide and describe a figure that depicts where all pages will reside within the operational site.

The pages would reside under http://dev.sjsu.edu/wascproj

WEBSITE STRUCTURE



WASC (Site Map)

- About WASC – general information about the project
  - Goals – Contains goals of WASC
  - Outcomes – Information on the outcomes of both WASC projects
- WASC Projects - Defines projects
  - Lit Review – information about Lit Review Project
  - Essay No.2 – information about Essay No.2 Project
- 5 Proficiencies – Rubrics used for the Java Application

  *downloadable pdfs below are linked
  - Critical Thinking
  - Written Communication
  - Quantitative Literacy
  - Information Literacy
  - Oral Communication

- Project Documentation – Documents for all phase and java app
  - Java Documentation
  - Phase Documentation
    - Phase 1 (Website Worksheet, Gannt Chart & questionnaire)
    - Phase 2 (Design Worksheet)
    - WBS
    - Phase 3-5 Development Plan
- Java Program Scanner – tool for scanning pdf departmental program plan
    - Tool
    - Result

[Ezekiel Calubaquib]

3.2 Site Preparation

Describe the preparation required for the site on which the system will operate. Define any changes that must occur to the operational site and specify features and items that should be modified to adapt to the new product. Identify the steps necessary to assist the customer in preparing the customer's designated sites for installation of the accepted website.

*No changes to operational site because there was no initial site to build upon

Preparation requires:

- Layout approval for the content of the website
- Information about projects (small descriptions)
- Team coordinated needs w/ Lit Review (what both teams would do with the website since it's a coordinated website)
- Finished program plan scanner java application that is hosted on the Google App Engine, so that the tool could be embedded inside WASC project website for use.
- Navigation Links and website content
- All documentations uploaded according to the approved layout

Steps/Info to navigate site

- *Intro Page – contains the necessary introduction to WASC projects and quick links*

- *About WASC – contains detailed information about WASC projects, our goals and outcome of both project (Lit Review and Essay No. 2)*
- *5 Proficiencies – Contains all the 5 rubrics/proficiencies the whole Essay No. 2 and Lit Review project relies on. These are the most important documents that defines the program scanner.*
    - *Importance of Rubrics*
        - *The rubrics define what the program plan scanner should do for the later final product.*
        - *Each rubric is provided keywords used for the program plan scanner to separate each scanned file(s) in pdf to their respective proficiencies.*
            - *Importance of keywords*
                - *Given by Dr. Anne Marie Todd (see Keyword Rubrics)*
                - *The keywords are the extracted from the rubrics, which can be easily transformed to an automatic program. There are two levels of keywords categorized by their importance, and different weight was given to different levels.*
                - *Additional keywords, as seen from keywords.txt are added by group members but are approved by Dr. Todd.*
                - *keywords.txt format*
                    - *The format is "keyword, level#, rubric#"*
                    - *keyword is the word we are going to search in the file*
                      level# is 1 or 2, level 1 is more important and has larger weight
                      rubric# is the where the keyword is counted
                - *Keywords, which are based of the 5 proficiencies, are the basis of equation for scoring plans*
                    - *Equation*

                        1. The five rubrics are evenly weighed, suppose the total score is 4, and then the total score is the average of 5 rubrics.
                        2. For the score of each rubric, the information needed are:

                    - The number of the first level words --N1

- The number of the second level words -- N2
- The number of words in this website -- NFrom these two we can get
- The total number of the related words -- N=N1+N2
- The density of first level words -- d1=N1/N, if N=0, then d1=0
- The density of second level words -- d2=N2/N, if N=0 then d2=0
- The density of the related words -- dr=N/Nt

  e.g. In our test.pdf rubric1, N1=8, N2=9, N=17, Nt=2512, d1=0.47, d2=0.53, dr=0.0068

The formula to calculate the score of this rubric is:

Score = 4*(d1+a*d2+b*d1*d2)*((1+dr)^c) , if the result is greater than 4, the score=4

Where a, b, c are the coefficients, please use a=0.25,b=1-a=0.75, c=2 for all the rubrics. In the example, the score is 3.2

- *Project Documentation – where all documents/plans for the project are collected can be downloaded by clicking the link*
  - *Essay No. 2 Section*
    - Java scanner application – contains documented codes
    - Project Phase I-V documentations
      - Phase 1 documentation contains “Client Action Plan”, “Project Analysis”, and Gannt Chart
      - WBS contains work breakdown sheet
      - Phase 2 documentation contains “Design Worksheet”
      - Phase 3-5 documentation contains “Development Plan” (this document)
- *Java Scanner – tool used to process departmental programming plans*
    - *“Choose File” button would indicate where to upload the file as well as allow the user to choose the file to upload.*

- *The scanner is enabled to select multiple files by selecting 2 or more files at once. Click “Open” button afterward.*

- *Results would be shown after the submission of the file (but no graphs just .json text with the score of the program plan by each of the five proficiency) Results are calculated based on the provided rubrics and statistical equation provided by Chunbo and Wanzhen.*

  - *Equation (see calculateScore method under the KeywordAnalyzer.java code in Appendices)*

[Ezekiel Calubaquib/ Edward Ciotic]

3.3 Assessment of Deployment Readiness

Describe the method for use in assessing deployment readiness.

Deployment readiness can only be done by:

1. *Java Scanner error test. This is done by uploading various files to verify that the scanner works according to the specification (ex. Small 100kb and big 6mb pdf files). This test demonstrates that the java scanner functions without any errors.*
2. *Verify that the website does not contain any broken links and that all content is accessible.*
3. *Verify that all the documentations Phase I-V, java codes, and other additional documents or images has been uploaded to the website.*
4. *Publish all unpublished pages in OUCampus*
5. *Prof. Debra Caires' website assessment and approval*
6. *Dr. Todd (Client) approval*

3.4 Product Content

Identify the configuration audits and reviews to be held after the website is tested and accepted and before the website is installed in an online environment.

Configuration would include the following:

- Accuracy and reliability of separation of programming plans by each proficiencies from the equation provided by stats major Chunbo and Wanzhen in the KeywordAnalyzer.java.
- Configuration audit on program scanner if it accepts program plan pdf of most sizes—see its limits and time-outs due to reliance on Google app engine server.
- Assessing easy to download and location of all documentation which describes the procedures of how the project was done, processed, and managed.
- Assessing how user-friendly the navigation of the website is.
    - Are topics/ links easy to find?
- Errors in computing and results of the program plan scanner tested and verified.
    - ex. are the results reliable? Are results accurate enough that if one would grade a school or college departmental program plan by hand would give same result?

3.5 Deviations and Waivers

Provide additional information regarding any waivers and deviations from the original Software Requirements Specification/Website Configuration.

Deviation from the alpha plan included:

- Implementing Google Analytics (would be implemented by next group CS100w Spring)
    - Using Google Analytics would show the results of the scanned document into:
        - Graphs and/or Charts.
            - Separating each scanned program plan documents into their respective proficiencies.
            - Scores of a program plan based on each of the five proficiencies
- Relying on Google developers help to enable Google Drive to be used to store and read of docs, pdf or any type of file via its embedded OCR feature.
- Reliance on OCR (*note iText was used instead)
- Reliance on .pdf files; reading .doc files are not currently supported by the java application. (Future subsequent team would need to implement a way to enable the java program plan scanner to read docs or any type of files acceptable by the use of a plugin like iText or something similar)
- Using Google drive for server (*note used Google app engine instead)
- Using Debra Caires CS100W/200W Server (*note used Google app engine instead)
- Scanning specific sections of a program planning document (Lit review implementation. Future subsequent team would have to implement a way to scan section of a program plan and asses/ score the documents there differently --- See Lit review team's proposal)
- Using jQuery/Ajax to parse the .json output result given by the program plan scanner (java app)to a more readable output text. (*note if Google Analytics is implemented this would not have to be done by subsequent group unless if needed or is applicable)

[Ezekiel Calubaquib/ Edward Ciotic]

Section 4. Data Creation/Conversion

Describe activities for creation of the physical database and population of that database with initial data. Identify the scope of work that must be accomplished. Include subsections that

describe data that must be loaded and legacy data that must be transferred prior to deployment of the website. Include a description of a data validation process.

Scope of work:

Two or more people in the group was designated a java class or two to work on. The following are:

1. Akshat Kukreti - PDF to Text using iText
    - Conversion of pdf files to text to enable the java application to read off pdf files submitted using the form
2. Michael Riha - Keyword Analyzer, Prefix Tree
    - Reads of keywords.txt to be used by java app
    - Reads of words from the pdf that was parsed using iText
    - Scores individual document by equation provided by Wanzhen and Chunbo (stats majors)
    - Made methods to enable output
3. Tim Stullich, Eddward Ciotic, and Micheal Keats – Google App Engine Server
    - Stores the data/ java app to run on server.
    - Used GSON plugin to convert the array of results to .json
    - Outputted .json text format to enable easy manipulation of data for later use.
    - Use .json and parse it to jQuery for a much readable format result or use for Google Analytics to show charts/graph (Unimplemented – would be done by subsequent group)

    NOTE: Validations are done inside the java classes (see codes in Appendices)

Data creation processes are going to be handled by the following:

1. Processing uploaded files with the proficiency keywords (keywords.txt) to the java app (KeywordAnalyzer.java with the WASC_EngineServlet.java).
2. The java app would then utilize a tree data structure for verifying and reading that the uploaded file contains the proficiency keywords.

- Synonyms of the keywords would be also be included and scored accordingly.

3. Upon uploading and submission of pdf file(s), the application will scan the file(s) and make a count of all the primary, secondary, and non applicable keywords and count their occurrence and their respective scores from the five proficiencies.

4. Output separate each program plan and scores are given from each of the five proficiencies.

    - Results are outputted in .json text format (see WASC_EngineServlet.java in Appendices)

5. Data are going stored and discarded at once after each pdf(s) submission

6. Validations are done by methods within the java application (see codes in Appendices).

- NOTE: See codes in Appendices for full view of the specific procedures handled by the java application.

[Ezekiel Calubaquib/Edward Ciotic]

Section 5. Phase Rollout

Describe activities for each phase function rollout or a phased user base rollout (make sure you foreshadow what the next team must accomplish).

Essay No.2 FALL 2012 team

Phase I-III – N/A didn't get to meet client till a week before phase III ended

- Received 5 proficiency documents for in the prior week
- Team research on using Google Analytics and Google Drive
- Ask Google Developers on their forum for help

Phase IV – Treated as a combination of Phase I-IV

- Project clarification and needs specified by client Dr. Todd
- Keywords from 5 proficiencies/rubrics obtained from Dr. Todd
- Java application (reading in text files and searching for keywords)
- iText researched and used to convert pdf's to file
- Google app engine (server) researched and made to store the java app
- GSON plugin researched and implemented to output json format
- jQuery researched but not used

Phase V – Last Phase

- OUCampus made a new first WASC Project website template for both Essay No. 2 and Lit Review Projects.
- Front End Google App Engine file uploader finished.
- Program plan scanner with help of google app engine is embedded
  - link to google app engine http://wascengine.appspot.com/
- Poster for WASC made also tweaked for website use.

- Uploaded all documentation
- Scanner and website release.

Next (Subsequent CS100/200W team)

- Need to incorporate Google Analytics to the java app and illustrate the statistics, median values, standard deviations of programming plans based on their specific proficiencies through Graphs and charts.
  - Use of the .json format could be used or not depending on the application of Google Analytics
  - jQuery (if needed) to show results not formatted in json.
- Need to implement (for Lit Review) a way for the java application to scan a specific section of documents
  - Specific sections would be graded higher than non specific sections of the document submitted. (See Lit Review proposal in website)
- Gather all programing plans that was scanned to their specific criteria from the 5 proficiency (* implemented with Google Analytics)
- Implement a way to read other types of files other than .pdf whichever is applicable
- Increase accuracy from keywords and equation used to score each program plan
- Provide reasoning/ case study why the keywords or java program plan scanner is reliable
- Document revision changes

[Ezekiel Calubaquib / Micheal Keats]

Section 6. Training and Documentation

6.1 Training

Describe the plans for preparing and conducting training for the purpose of training all stakeholders (your client) on the use of the product.

⇒ Since we only have one person to train (Dr. Todd), the training will be very straightforward.

- Hands-on demonstration of the final program guiding Anne Marie through how to use it
- Give an overview of the results page features and how to use it effectively
- If the client wants, we may explain the backend in relatable terms
- Answer any questions the client may have

6.2 Documentation

6.2.1 Documents

Identify and describe each document that will be produced for the purpose of aiding in installation, support, or use of the product.

⇒ We will produce a tutorial document or video of some sort that essentially covers the training from section 6.1. We will also create a document which details the technical aspects of the system including information about the backend and frontend interfacing. This document will be intended for those who work on the project after us. Our code is and will be documented.

- Tutorial document or video (OUCampus)

  http://its.sjsu.edu/training/

- Technical document for future support of the project
- Documented code

6.2.2 Documentation Activities

Describe activities required to develop each document.

⇒ Both documents will require the backend/frontend to be finalized. Once the results page works we could start working on the tutorial document, but for the technical document we will need everything to be final because the code and backend are subject to change until then.

[Michael Riha]

Section 7. Transition to Support

7.1 Resource Requirements

Identify and describe the resources required to support the operational website and the interrelationships of those resources to the operational website.

⇒

- The website will run on SJSU's website using OUCampus
- The site will need temporary storage for uploading documents before sending them to the keyword analyzer
- We will interface this website with Google AppEngine using the OAuth protocol, the relation will be a simple back and forth from the website to AppEngine and back

7.2 Recommended Procedures

Describe any procedures, advice, and lessons learned that may aid the support organization in supporting the operational website and the production online environment.

⇒ None

7.3 Expected Areas of Change

Describe the expected or anticipated areas of change to the operational website.

⇒

- WASC Lit Review and Essay projects will be hosted on the website
- Project information such as the five proficiencies and the rubrics will be on the WASC Essay portion of the site
- There will be another page for uploading documents for analysis. Once documents are uploaded, it changes into a results page.

7.4 Transition Activities

Describe the plans for transitioning the operational website to the support organization.

⇒ We will provide account credentials to administer the Google AppEngine backend and any other information needed (OUCampus?).

[Michael Riha]

Section 8. Notification of Deployment

Describe the method of notifying all stakeholders/clients of the successful release of the website. Identify stakeholders/clients and groups requiring notification.

After the successful release of the website, a notification will be sent to our client, Dr. Anne Marie Todd, via email. An email would be also sent to Professor Debra Caires for approval.

Steps:

1. Check all procedures (ie. scanner app, website...etc.) are done.
2. Email client for meeting.
3. Present Client website with program plan scanner.
4. Sign off development plan documents/proposal.
5. Email client released website and app.
6. Release of project and approval of Debra Caires.

[Charles Long]

Section 9. Operations and Maintenance Planning

Describe the maintenance and operations activities for the website.

- Website is created and maintained using OU Campus
- Basics for using OU Campus can be found at: http://www.sjsu.edu/webservices/training/oucampus/onlinetraining/
- For the WASC website, every individual page has a different folder in OU Campus
- You can edit meganav, the navigation bar, on the top of the page, by going to ~/includes/meganav.inc
    - Please be careful when making changes to this file, unexpected results happen if you don't follow exactly what OU Campus wants.
- When moving folders in OU Campus, be sure to check all associated pages for broken links, OU Campus does not automatically refractor existing pages to account for the move.
- The site index located on the front page of this website needs to be republished every time you create a new page to account for the directory changes
- There is no real maintenance you need to do to the site unless you plan on adding or removing content
    - All uploaded content will be found in the production server found by clicking on the pages link in OU Campus followed by clicking on the production tab (default is the staging tab)
- Note: all hard copies of code java and website are printed and/or stored in a CD/thumb drive given to Debra from the group

[Charles Long]

Section 10. Release Planning

10.1 Release of the Website

Describe the processes for release of the website for operational use.

1. Finalize test on Java App.
2. Embed all documentations from both teams (Essay No.2 and Lit Review).
3. Test all links if broken.
4. Tutorial for client.
    - Discussed during a meeting for signing off development plan documents.
5. Approval from client Dr. Anne Marie Todd and Professor Debra Caires.
6. Publishing from finished product in Oucampus.

10.2 Contingency Planning

Describe the contingency plan to be executed if problems occur during deployment activities.

If embedded tool fail we would have the Google app engine still running from a temporary backup page and another Google app engine server. And since a user only needs the link to embed the app then we can upload it to another site and link the users there.

If all plans fail to be executed we would have everything documented for other team to accomplish.

*Note OUcampus can revert to previous files. In addition all html and app are stored and hard copy is available in this document.

10.3 End of Life Plannings

Describe the plan for disposition or retirement of the old website and the execution of the new website.

N/A the website for the project is non-existent.

[Ezekiel Calubaquib/Micheal Keats]

Section 11. References

Provide a list of all documents and other sources of information referenced in the Deployment Plan and utilized in developing the Deployment Plan. Include for each the document number, title, date, and author.

Document No.	Document Title	Date	Author
1	Critical Thinking Rubric	10/11/12	WASC
2	Oral Communication	10/11/12	WASC
3	Information Literacy	10/11/12	WASC
4	Quantitative Literacy	10/11/12	WASC
5	Written Communication	10/11/12	WASC
6	Keywords for 5 Proficiencies	10/15/12	Dr. Anne Marie Todd
7	Using Google Analytics for Improving Library Website Content and Design: A Case Study	10/23/12	Wei Fang
8	Information Visualization and Visual Data Mining	10/23/12	Daniel A. Keim
9	Data Mining Technology Across Academic Disciplines	10/23/12	Lesley Farmer
10	Metadatapedia: A Proposal for Aggregating Metadata on Data Archiving	10/23/12	David M. Nichols Michael B. Twidale Sally Jo Cunningham
11	Extracting significant Website Key Objects: A Semantic Web mining approach	10/23/12	Juan D.Vela´squez Luis E.Dujovne, Gaston L'Huillier
12	Beyond Description: Converting Web Site Usage Statistics into Concrete Site Improvement Ideas	10/23/12	Julie Arendt, Cassie Wagner
13	Web Analytics and the Art of Data Sum-marization	10/23/12	Archana Ganapathi, Steve Zhang
14	Monitoring web traffic source effectiveness with Google Analytics	10/23/12	Beatriz Plaza
15	Click Analytics: Visualizing Website Use Data	10/23/12	Tabatha A. Farney

Document No.	Document Title	Date	Author
16	JSON (for Google App Engine) http://www.json.org/javadoc/org/json/JSONObject.html	11/15/12	Google
17	iText http://itextpdf.com/support.php#book	11/08/12	iText Software Corp.
18	Keywords for 5 Proficiencies (revised)	11/25/12	Dr. Anne Marie Todd
19	Java scanner server class (includes native java scanner app and pdf to text) https://github.com/tstullich/WASC-Engine.git	11/28/12	Tim Stullich, Eddy Cioitc, Michael Keats
20	Java scanner and pdf to text (iText) https://github.com/michaelriha/WASC-Essay/	11/28/12	Michael Riha, Akshat Kukreti

[Chunbo Tan]

Section 12. Glossary

Define all terms and acronyms required to interpret the Deployment Plan properly.

Back End (programming): *the end stage of a process*

- http://en.wikipedia.org/wiki/Front_and_back_ends

Critical thinking: a habit of mind characterized by the comprehensive exploration of issues, ideas, artifacts, and events before accepting or formulating an opinion or conclusion

- WASC 5 Proficiency

Google AppEngine: a platform as a service cloud computing platform for developing and hosting web applications in Google-managed data centers

*- http://en.wikipedia.org/wiki/Google_App_Engin*e

Google Drive: a file storage and synchronization service by Google that was released on April 24, 2012

- http://en.wikipedia.org/wiki/Google_Drive

Google Analytics: a service offered by Google that generates detailed statistics about the visits to a website

- http://en.wikipedia.org/wiki/Google_Analytics

Front End (programming): the initial stage of a process, an interface between the user and the back end

- http://en.wikipedia.org/wiki/Front_and_back_ends

GitHub: a web-based hosting service for software development projects that use the Git revision control system

- http://en.wikipedia.org/wiki/GitHub

Information Literacy: ability to know when there is a need for information, to be able to identify, locate, evaluate, and effectively and responsibly use and share that information for the problem at hand adopted from The National Forum on Information Literacy

- WASC 5 Proficiency

iText: a free and open source library for creating and manipulating PDF files in Java

- http://itextpdf.com/itext.php

HTML: hypertext markup language: a text description language that is used for electronic publishing, esp on the World Wide Web

- *http://dictionary.reference.com/browse/html*

Java: a high-level, object-oriented computer programming language used especially to create interactive applications running over the Internet.

- *http://dictionary.reference.com/browse/java*

JSON (.json) : is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

- *http://www.json.org/*

jQuery : jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is designed to change the way that you write JavaScript.

- *http://jquery.com/*

Oauth: an open standard for authorization

- *http://en.wikipedia.org/wiki/OAuth*

OCR: Optical character recognition, usually abbreviated to OCR, is the mechanical or electronic conversion of scanned images of handwritten, typewritten or printed

- *http://en.wikipedia.org/wiki/Optical_character_recognition*

Oral Communication: a prepared, purposeful presentation designed to increase knowledge, to foster understanding, or to promote change in the listeners' attitudes, values, beliefs, or behaviors.

- *WASC 5 Proficiency*

OUCampus: the leading web content management system for higher education institutions

- *http://omniupdate.com/products/oucampus/*

Quantitative Literacy (QL): a "habit of mind," competency, and comfort in working with numerical data. Individuals with strong QL skills possess the ability to reason and solve quantitative problems from a wide array of authentic contexts and everyday life situations

- *WASC 5 Proficiency*

WASC: Western Association of Schools and Colleges (WASC), one of six regional associations that accredit public and private schools, colleges, and universities in the United States

- http://www.wascweb.org/

Written communication: the development and expression of ideas in writing, involve working with many different writing technologies, and mixing texts, data, and images.

- *WASC 5 Proficiency*

[Chunbo Tan]

Section 13. Revision History

Identify changes to the Deployment Plan.

Version	Date	Name	Description
1.0	11/07/02	First Publication	The first draft has been published and further changes will be noted here in the future.
2.0	11/14/02	Second Revision	Documentation for finished application except for google app engine (tool needed to embed java app to website)
3.0	11/23/02	Third Revision	Tweaked wording and added codes in appendices
4.0	11/27/12	Fourth Revision	Added in more detailed content in order to be more descriptive. Finalized all parts of the deployment plan

Section 14. Appendices

Include any relevant appendices (completed Gantt chart and team member contact information in table form gets placed here).

Name	Email	Github Handle
Akshat Kukreti	akshatkukreti@gmail.com	akshat
Charles Long	chars.long@gmail.com	charslong
Chunbo Tan	chunbo.tan@students.sjsu.edu	NA
Edward Ciotic	ciotic@gmail.com	ciotic
Ezekiel Calubaquib	ezekit@yahoo.com	ezekit
Michael Keats	michaelkeats@gmail.com	keatsfonam
Michael Riha	rihamichael@gmail.com	michaelriha
Tim Stullich	timstullich@gmail.com	tstullich
Wanzhen Wu	wanzhen719@gmail.com	NA

[Tim Stullich]

Code:

KeywordAnalyzer.java

```
package com.sjsu.wascengine;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.SortedSet;
import java.util.TreeSet;
import com.sjsu.wascengine.PrefixTree.Node;

/**
* A KeywordAnalyzer that is capable of performing keyword statistical analysis
```

```
* on text. Calling parseText multiple times without a reset will continuously
* build word counts. Also, multiple keyword files can be read into the tree.
* @author Michael Riha
*/
public class KeywordAnalyzer
{
   private SortedSet&lt;String&gt;[][] keywordsUsed;
   private PrefixTree keywordTree;
   private int[][] wordCounts;
   private double[] scores;
   private int totalWords;

   private static final int RUBRICS = 5, WEIGHTS = 2;
   private static final int a = 6, b = 1, c = 4;

   /** Constructs a KeywordAnalyzer with no statistical data */
   @SuppressWarnings({ "unchecked", "rawtypes" })
   public KeywordAnalyzer()
   {
      keywordsUsed = new SortedSet[RUBRICS][WEIGHTS];
      wordCounts = new int[RUBRICS][WEIGHTS];

      for (int i = 0; i &lt; RUBRICS; ++i)
      for (int j = 0; j &lt; WEIGHTS; ++j)
      {
         keywordsUsed[i][j] = new TreeSet();
         wordCounts[i][j] = 0;
      }

      scores = new double[RUBRICS+1];
      keywordTree = new PrefixTree();
      totalWords = 0;
   }

   /** Resets word statistics and keyword occurrences in the tree */
   public void reset()
   {
      for (int i = 0; i &lt; RUBRICS; ++i)
      for (int j = 0; i &lt; WEIGHTS; ++i)
```

```
      {
         keywordsUsed[i][j].clear();
         wordCounts[i][j] = 0;
      }

      scores = new double[RUBRICS+1];
      keywordTree.reset();
      totalWords = 0;
   }

   /** Creates a new, empty keyword tree */
   public void purgeKeywords() { keywordTree = new PrefixTree(); }

   /**
   * Adds the keywords from the specified file to the keyword tree
   * Each line should contain one entry in the format: keyword,weight,rubric
   * Example: Critical,2,1 would create an entry with weight 2 for rubric 1
   * @param filename the file to read
   */
   public void readKeywordFile(String filename) throws FileNotFoundException, IOException
   {
      BufferedReader br = new BufferedReader(new FileReader(filename));
      try
      {
         String line = br.readLine();
         String[] parts = line.split(",");
         keywordTree.add(parts[0], Integer.valueOf(parts[1]), Integer.valueOf(parts[2]));
         while (true) // while (line != null) causes NullPointerException
         {
            line = br.readLine();
            if (line == null) break;
            parts = line.split(",");
            keywordTree.add(parts[0], Integer.valueOf(parts[1]), Integer.valueOf(parts[2]))
;
         }
      }
      finally { br.close(); }
   }
```

```
/**
* Parses each word in the input and updates the keyword statistics fields
* @param words Plaintext alphabetic non-numeric words. one word per entry
*/
public void parseText(ArrayList&lt;String&gt; words)
{
   Node values;
   int rubric, weight;
   for (String word : words)
   {
      if(word.length() != 1)
      {
         values = keywordTree.find(word);
         if (values != null)
         {
            rubric = values.getRubric();
            weight = values.getWeight();
            ++wordCounts[rubric - 1][weight - 1];
            keywordsUsed[rubric - 1][weight - 1].add(word);
         }
      }
      ++totalWords;
   }
}

/**
* Calculates each rubric score and returns them in order with the total
* rubric score (simple average) at the end
* @return an array of scores between 0.0 and 4.0. [r1, r2, r3, r4, r5, Tot]
*/
public double[] calculateScores()
{
   double score, sum = 0.0;
   for (int i = 0; i &lt; RUBRICS; ++i)
   {
      score = calculateScore(wordCounts[i][0], wordCounts[i][1], totalWords);
      scores[i] = score;
      sum += score;
   }
```

```
    scores[RUBRICS] = sum / RUBRICS;
    return scores;
}

/**
* Calculates an individual rubric score based on the number of weight one
* and two words as well as the total number of words in the document
* @param weightOneCount the number of weight one words
* @param weightTwoCount the number of weight two words
* @param totalWordCount the total word count
* @return a rubric score between 0.0 and 4.0 based on the formula
* score = (N/a)*((1+d1)^b)*((1+dr)^c)
* N: Total number of keywords
* d1: Density of weight one words (weightOneCount / N)
* dr: Density of related words (N / totalWordCount)
*/
public static double calculateScore(int weightOneCount, int weightTwoCount,
int totalWordCount)
{
    if (totalWordCount &lt;= 0) return 0.0;
    double N = (double) weightOneCount + weightTwoCount;
    double dr = N / totalWordCount;
    double d1 = (N == 0) ? 0 : weightOneCount / N;
    return Math.min(4, (N/a) * Math.pow(1+d1, b) * Math.pow(1+dr, c));
}

/**@param keyword the word to look for
* @return the number of times the keyword was found
*/
public int getKeywordOccurrences(String keyword)
{
    return keywordTree.findNoIncrement(keyword).getOccurrences();
}

/**@return 2 dimensional array of sorted sets containing the keywords for
keywordsUsed[rubric][weight] */
public SortedSet&lt;String&gt;[][] getKeywordsUsed() { return keywordsUsed; }

/**@return Scores 0-4 in this format [rub1, rub2, rub3, rub4, rub5, total] */
```

```
   public double[] getScores() { return scores; };

   /**@return the wordcounts for each [rubric][weight] */
   public int[][] getWordCounts() { return wordCounts; }

   /**@return the total number of words parsed since reset */
   public int getTotalWords() { return totalWords; }
}
```

PdfExtract.java

```
package com.sjsu.wascengine;

import com.lowagie.text.DocumentException;
import com.lowagie.text.pdf.PdfReader;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Iterator;

/**
* @author Akshat Kukreti
*/
public class PdfExtract {
   private static final int START_SIZE = 2000;
   public static ArrayList&lt;String&gt; convertToText(InputStream fileStream) throws Docu
mentException, IOException
   {
      boolean brackopen = false;
      boolean brackclose = false;

      //Stores the words extracted from the pdf
      ArrayList&lt;String&gt; words = new ArrayList&lt;String&gt;(START_SIZE);

      //Accumuates a word
      StringBuilder word = new StringBuilder();

      if(fileStream == null){
         System.exit(1);
      }
```

```
try{
   PdfReader reader = new PdfReader(fileStream);
   int numpages = reader.getNumberOfPages();
   for(int i=1; i&lt;=numpages; i++){
      byte[] pagecontent = reader.getPageContent(i);
      int contentlength = pagecontent.length;
      char[] charcontent = new char[contentlength];

      for(int j=0; j&lt;contentlength; j++){
         char c = (char)pagecontent[j];
         charcontent[j] = c;
      }

      //Escaped characters
      for(int k=0; k&lt;contentlength; k++){
         if(charcontent[k] == '\\'){
            charcontent[k] = ' ';
            charcontent[k+1] = ' ';
         }
      }

      for(int l=0; l&lt;contentlength; l++){
         if(charcontent[l] == '('){
            brackopen = true;
            continue;
         }
         if(charcontent[l] == ')'){
            brackclose = true;
         }
         if(brackopen && !brackclose){
            char c = charcontent[l];
            if((int)c &gt;= 65 && (int)c &lt;= 90 ||
            (int)c &gt;= 97 && (int)c &lt;= 122 ){
               char lowerc = Character.toLowerCase(c);
               word.append(lowerc);
            }
            else if(c == '-'){
               word.append(c);
```

```
                }
                if(c == ' '){
                    String wordstring = word.toString();
                    if (!wordstring.isEmpty()){
                        words.add(wordstring);
                        word.delete(0, word.length());
                    }
                }
            }
            if(brackopen && brackclose){
                brackopen = false;
                brackclose = false;
            }
        }
    }
}
catch(IOException e){
    System.out.println(e.toString());
}

//Correcting split up words and adding them
ArrayList&lt;String&gt; unsplitwords;
unsplitwords= new ArrayList&lt;String&gt;(START_SIZE);
Iterator&lt;String&gt; wordit = words.iterator();
while(wordit.hasNext()){
    String word1 = wordit.next().toString();
    if(word1.charAt(word1.length() -1) == '-' &&
    word1.length() &gt; 1){
        String word2 = wordit.next().toString();
        word1 = word1.replace('-',' ');
        String concatword = word1.concat(word2);
        concatword = concatword.replaceAll(" ","");
        unsplitwords.add(concatword);
    }
    else if(!word1.equals("-")) {
        unsplitwords.add(word1);
    }
}
```

```
      return unsplitwords;
   }
}
```

PrefixTree.java

```
package com.sjsu.wascengine;
/**
* A PrefixTree to store keywords. Keywords have their corresponding weights,
* rubric, and how many times they have been found stored in the leaf nodes
*
* @author Michael Riha
*/
public class PrefixTree
{
   private Node root;

   /** Constructs an empty PrefixTree */
   public PrefixTree() { root = new Node(); }

   /**
   * A node in the tree which has up to 26 children and a weighted value
   * If the weight is 0, then the node is not a leaf on the tree
   * Also stores the associated rubric and how many times a word has been used
   */
   public class Node
   {
      private int rubric;
      private int weight;
      private int occurrences;
      private Node[] children;

      private Node()
      {
         rubric = 0;
         weight = 0;
         occurrences = 0;
         children = new Node[26];
      }
```

```
    public int getRubric() { return rubric; }
    public int getWeight() { return weight; }
    public int getOccurrences() { return occurrences; }
    public Node[] getChildren() { return children; }
}

/**
* Adds a word to the PrefixTree with corresponding weight and rubric values
* Does not allow duplicates or words whose prefix is already in the tree.
* @param word the word to add
* @param weight the weight of the word
* @param rubric which rubric the word is associated with
* @return true if the word was added, false if it was already in the tree
*/
public boolean add(String word, int weight, int rubric)
{
    word = word.toLowerCase();
    Node cur = root;
    int next_idx;

    for (int i = 0; i &lt; word.length(); ++i)
    {
        // add the node. if is already weighted then return false
        next_idx = (int) (word.charAt(i) - 'a');
        if (cur.children[next_idx] == null)
        if (cur.weight == 0)
        cur.children[next_idx] = new Node();
        else return false;
        cur = cur.children[next_idx]; // descend the tree
    }
    // reached a leaf so set the weight
    if (cur.weight == 0)
    {
        cur.rubric = rubric;
        cur.weight = weight;
        return true;
    }
    else return false; // already in the tree
}
```

```
/**
* Find a word in the tree iteratively, if it exists, including wildcards
* E.g. if "critic" is in the tree, "critically" will map to that
* @param word the word to look for
* @return the leaf node associated with this word, null if not found
*/
public Node find(String word)
{
   word = word.toLowerCase();
   Node cur = root;
   Node next;

   for (int i = 0; i &lt; word.length(); ++i)
   {
      if (cur.weight &gt; 0) // "wildcard" found early
      {
         ++cur.occurrences;
         return cur;
      }
      if(word.charAt(i)=='-')
      ++i;
      next = cur.children[(int) word.charAt(i) - 'a'];
      if (next == null) // word not in tree
      return null;
      cur = next; // descend the tree
   }
   if (cur.weight &gt; 0)
   {
      ++cur.occurrences;
      return cur;
   }
   else return null;
}

/**
* Same as find but does not increment the number of occurrences of word
* @param word the word to look for
* @return the leaf node associated with this word, null if not found
```

```
   */
   public Node findNoIncrement(String word)
   {
      word = word.toLowerCase();
      Node cur = root;
      Node next;

      for (int i = 0; i &lt; word.length(); ++i)
      {
         if (cur.weight &gt; 0) // "wildcard" found early
         return cur;
         next = cur.children[(int) word.charAt(i) - 'a'];
         if (next == null) // word not in tree
         return null;
         cur = next; // descend the tree
      }
      if (cur.weight &gt; 0)
      return cur;
      else return null;
   }

   /** Sets the number of occurrences of each word in tree to 0 recursively */
   public void reset() { resetHelper(root); }

   private void resetHelper(Node node)
   {
      if (node.occurrences != 0)
      node.occurrences = 0;
      else
      for (int i = 0; i &lt; 25; ++i)
      if (node.children[i] != null)
      resetHelper(node.children[i]);
      else break;
   }
}
```

WASC_EngineServlet.java

```
package com.sjsu.wascengine;
```

```
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.SortedSet;

import javax.servlet.ServletException;
import javax.servlet.http.*;

import org.apache.commons.fileupload.FileItemIterator;
import org.apache.commons.fileupload.FileItemStream;
import org.apache.commons.fileupload.servlet.ServletFileUpload;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.lowagie.text.DocumentException;
/**
* This is going to be the "main" class for the App Engine. It
* is going to listen on a certain port for an Http Request and
* should receive a pdf file. It is then going to be doing the
* work in order to figure out the appropriate rubric scores and
* return the final result as a JSON object.
*
* @author Tim Stullich
*
*/
@SuppressWarnings("serial")
public class WASC_EngineServlet extends HttpServlet
{
   private final int RUBRICS = 5;
   private final int WEIGHT_CATEGORIES = 2;
   public void doPost(HttpServletRequest req, HttpServletResponse resp)
   throws IOException, ServletException
   {
      //Gonna try and capture the files that were sent
      //through the request servlet
      try{
```

```
        ServletFileUpload upload = new ServletFileUpload();
        //Sets the MIME type to be JSON
        resp.setContentType("text/plain");

        FileItemIterator iterator = upload.getItemIterator(req);

        JsonArray resultsArray = new JsonArray();
        while (iterator.hasNext())
        {
           FileItemStream item = iterator.next();
           InputStream stream = item.openStream();

           //Checks whether there is a form field or not
           if (!item.isFormField())
           {
              //The work for the files will be done here
              resultsArray.add(analyzeText(resp, stream, item.getName()));
           }
        }
        Gson gson = new GsonBuilder().setPrettyPrinting().create();
        String json = gson.toJson(resultsArray);
        resp.getOutputStream().print(json);
     }
     catch (Exception e)
     {
        throw new ServletException();
     }
  }

  public JsonObject analyzeText(HttpServletResponse resp, InputStream fileStream, String
filename)
  throws FileNotFoundException, IOException, DocumentException
  {
     // Test readKeywordFile
     KeywordAnalyzer instance = new KeywordAnalyzer();
     instance.readKeywordFile("testfiles/keywords.txt");

     // Get a test pdf and parse the text
     ArrayList&lt;String&gt; text = PdfExtract.convertToText(fileStream);
```

```
instance.parseText(text);

// Print a detailed report about the file
int totOne = 0, totTwo = 0, total = 0, swap;

//Create A JSON Object that will hold all the results
JsonObject results = new JsonObject();
results.addProperty("fileName", filename);

int[][] wordCounts = instance.getWordCounts();
SortedSet&lt;String&gt;[][] sets = instance.getKeywordsUsed();

//Calculates Some General Statistics
for (int i = 0; i &lt; RUBRICS; ++i)
{
   for (int j = 0; j &lt; WEIGHT_CATEGORIES; ++j)
   {
      if (j == 0)
      totOne += wordCounts[i][j];
      else
      totTwo += wordCounts[i][j];
      total += wordCounts[i][j];
   }
}

results.addProperty("totalKeywords", total);
results.addProperty("weight1Keywords", totOne);
results.addProperty("weight2Keywords", totTwo);
results.addProperty("totalWords", instance.getTotalWords());

//Scores for each individual rubric will be calculated here
double[] scores = instance.calculateScores();

/*This JSON array will hold JsonObjects that contain
*more information on each rubric including scores,
*weigths and the frequency of certain keywords.
*/
JsonArray rubricScores = new JsonArray();
for (int i = 0; i &lt; RUBRICS; ++i)
```

```
      {
         JsonObject rubricScore = new JsonObject();
         rubricScore.addProperty("rubric" + (i + 1) + "score", scores[i]);
         for (int j = 0; j &lt; 2; ++j)
         {
            rubricScore.addProperty("weight" + (j + 1) + "WordsUsed", wordCounts[i][j]);
            //Counts the frequency of each word based in each weight class
            JsonArray wordFrequency = new JsonArray();
            for (String word : sets[i][j])
            {
               swap = instance.getKeywordOccurrences(word);
               total += swap;
               JsonObject aWord = new JsonObject();
               //Add the word with the calculated frequency
               aWord.addProperty(word, swap);
               wordFrequency.add(aWord);
            }
            rubricScore.add("words"+ (j+1) + "Frequency", wordFrequency);
         }
         rubricScores.add(rubricScore);
      }
      results.add("rubricScores", rubricScores);
      return results;
   }
}
```

Keywords.txt

critical,2,1

discern,2,1

discerning,2,1

keen,2,1

analytic,2,1

analysis,2,1

synthesize,2,1

combination,2,1

combine,2,1

construct,2,1

constructive,2,1

integration,2,1

structure,2,1

unify,2,1

analyze,2,1

examine,2,1

breakdown,2,1

dissect,2,1

review,2,1

investigate,2,1

reason,2,1

assumption,1,1

expectation,1,1

guess,1,1

hypothesis,1,1

inference,1,1

postulate,1,1

presupposition,1,1

supposition,1,1

theory,1,1

issue,1,1

concern,1,1

matter,1,1

problem,1,1

subject,1,1

topic,1,1

reflect,2,1

display,2,1

evince,2,1

exhibit,2,1

express,2,1

manifest,2,1

reveal,2,1

perspective,1,1

aspect,1,1

objectivity,1,1

overview,1,1

prospect,1,1

viewpoint,1,1

interpret,1,1

define,1,1

depict,1,1

describe,1,1

elaborate,1,1

explain,1,1

explicate,1,1

illustrate,1,1

paraphrase,1,1

portray,1,1

represent,1,1

translate,1,1

evaluate,2,1

judge,2,1

assess,2,1

check,2,1

criticize,2,1

rate,2,1

size,2,1

weigh,2,1

implications,1,1

suggestion,1,1

conclusion,1,1

presumption,1,1

reasoning,1,1

conclude,1,1

deduce,1,1

infer,1,1

rationalize,1,1

suppose,1,1

argument,1,1

debate,1,1

disagreement,1,1

dispute,1,1

decision-making,2,1

controlling,2,1

directing,2,1

governing,2,1

managing,2,1

ruling,2,1

logical,1,1

congruent,1,1

consequent,1,1

deducible,1,1

justifiable,1,1

rational,1,1

integrate,1,1

concatenate,1,1

conform,1,1

conjoin,1,1

consolidate,1,1

coordinate,1,1

unite,1,1

connections,1,1

ally,1,1

associate,1,1

association,1,1

intermediary,1,1

network,1,1

apply,1,1

engage,1,1

execute,1,1

exploit,1,1

implement,1,1

practice,1,1

utilize,1,1

problem-solving,1,1

diagnostic,1,1

discrete,1,1

dissecting,1,1

inquisitive,1,1

investigative,1,1

judicious,1,1

questioning,1,1

research,2,2

experimentation,2,2

exploration,2,2

investigation,2,2

probe,2,2

quest,2,2

experiment,2,2

information,2,2

instruction,2,2

material,2,2

knowledge,2,2

message,2,2

notice,2,2

sources,2,2

evidence,2,2

support,2,2

reference,2,2

quotation,2,2

attribution,1,2

property,1,2

nature,1,2

quality,1,2

ascribe,1,2

belong,1,2

citation,2,2

excerpt,2,2

example,2,2

quote,2,2

organize,1,2

arrange,1,2

systematize,1,2

systematically,1,2

compose,1,2

constitute,1,2

catalogue,1,2

classify,1,2

codify,1,2

summary,1,2

concise,1,2

abstract,1,2

compact,1,2

succinct,1,2

context,1,2

framework,1,2

background,1,2

text,1,2

substance,1,2

credible,1,2

believable,1,2

conclusive,1,2

plausible,1,2

reasonable,1,2

reliable,1,2

audience,2,3

write,2,3

writing,2,3

language,2,3

crowd,2,3

spectator,2,3

viewer,2,3

witness,2,3

listener,2,3

author,2,3

compose,2,3

inscribe,2,3

record,2,3

transcribe,2,3

paper,2,3

articulation,2,3

expression,2,3

tongue,2,3

support,1,3

clarity,1,3

thesis,1,3

essay,1,3

paper,1,3

backing,1,3

base,1,3

foundation,1,3

evidence,1,3

groundwork,1,3

accuracy,1,3

comprehensibility,1,3

distinctness,1,3

lucidity,1,3

perceptibility,1,3

hypothesis,1,3

postulate,1,3

premise,1,3

presumption,1,3

proposal,1,3

propose,1,3

theory,1,3

article,1,3

composition,1,3

dissertation,1,3

study,1,3

script,1,3

report,1,3

note,1,3

critique,1,3

deliver,2,4

present,2,4

message,2,4

speak,2,4

speech,2,4

bring,2,4

convey,2,4

express,2,4

remit,2,4

address,2,4

declare,2,4

exhibit,2,4

expound,2,4

give,2,4

offer,2,4

perform,2,4

pose,2,4

proffer,2,4

proposition,2,4

show,2,4

state,2,4

submit,2,4

suggest,2,4

tender,2,4

information,2,4

memo,2,4

communication,2,4

conversation,2,4

discussion,2,4

expressing,2,4

intercourse,2,4

verbalization,2,4

credibility,1,4

style,1,4

believability,1,4

integrity,1,4

possibility,1,4

probability,1,4

reliability,1,4

satisfactoriness,1,4

tenability,1,4

trustworthiness,1,4

validity,1,4

sort,1,4

technique,1,4

diction,1,4

form,1,4

pattern,1,4

mathematical,2,5

calculation,2,5

quantitative,2,5

data,2,5

algebraic,2,5

algorithmic,2,5

arithmetical,2,5

computative,2,5

geometrical,2,5

numerical,2,5

scientific,2,5

trigonometric,2,5

estimate,2,5

computing,2,5

computation,2,5

counting,2,5

measurable,2,5

gauge,2,5

weighable,2,5

figures,2,5

datum,2,5

problem-solving,1,5

accurate,1,5

graph,1,5

trend,1,5

equations,1,5

examining,1,5

diagnostic,1,5

dissect,1,5

interpretive,1,5

investigative,1,5

precise,1,5

exact,1,5

explicit,1,5

TRUE,1,5

unmistakable,1,5

diagram,1,5

chart,1,5

orientation,1,5

progression,1,5

tendency,1,5

appengine-web.xml

```
<?xml version="1.0" encoding="utf-8"?>
        <appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
          <application>wascengine</application>   <!-- name of appsite inserted here-->
          <version>1</version>
```

```
<!--
  Allows App Engine to send multiple requests to one instance in parallel:
-->
<threadsafe>true</threadsafe>

<!-- Configure java.util.logging -->
<system-properties>
  <property name="java.util.logging.config.file" value="WEB-INF/logging.properties"/>
</system-properties>

<!--
  HTTP Sessions are disabled by default. To enable HTTP sessions specify:

    <sessions-enabled>true</sessions-enabled>

  It's possible to reduce request latency by configuring your application to
  asynchronously write HTTP session data to the datastore:

    <async-session-persistence enabled="true" />

  With this feature enabled, there is a very small chance your app will see
  stale session data. For details, see
  http://code.google.com/appengine/docs/java/config/appconfig.html#Enabling_Sessions
-->

</appengine-web-app>
```

Index.html (index html for Google App Engine)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
    <!-- The HTML 4.01 Transitional DOCTYPE declaration-->
    <!-- above set at the top of the file will set     -->
    <!-- the browser's rendering engine into           -->
    <!-- "Quirks Mode". Replacing this declaration     -->
    <!-- with a "Standards Mode" doctype is supported, -->
    <!-- but may lead to some differences in layout.   -->

    <html>
```

```
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>App Engine</title>

  </head>

  <body>


<form action="wasc_engine" method="post" enctype="multipart/form-data">
<label for="file">Filename:</label>
<input id="yeh" type="file" name="file" />
<br />
<input type="submit" id="submit" value="Submit" />
</form>
<br />

  </body>
</html>
```