Eric Chan

Plagiarism Checker

Architecture and Design

Introduction

There will be two screens overall that will be displayed to the user in the Plagiarism Checker application, an input screen and an output screen. The input screen will be comprised of a file upload button, a text box and a submit button. A user may upload a text file and the contents of that text file will be displayed in the text box where it may be edited. The submit button will begin execution of the PHP script located at a web hosting server using whatever is inside the textbox as input. A copy of the input text will be processed by removing some of the most common words in the english language from that text. The resulting text will be split up every 32 words and place into an array to be used as queries for the Google Custom Search API. Each query passed through the Google Custom Search API will return a JSON object containing up to 10 websites. The top 5 websites (if there are at least 5) will be placed into an array labeled URLs and if any of the websites is already in the array, it will not be added in again. The script will then perform a matching algorithm on each website in the URLs array, extracting the text from each website and comparing it with the original copy of the input text. The matching algorithm will find all contiguous matching substrings between the input text and the text found on the website. If the contiguous matching substring contains at least 6 words, it will be added into an array and another array containing the number of matching words will be incremented by how many words are in that particular contiguous matching substring at the same index that the website is positioned in the URLs array. After the matching algorithm has finished running, the array of contiguous matching substrings will be placed into an array at the same index the website is positioned in the URLs array. After every website has gone through the matching algorithm, the script will then output the results to the user's display which will include the URL of each website, the percentage of the input text that match the website's text, and the list of all contiguous matching substrings with 6 or more words. At the top of the output screen will be a button that will take the user back to the input screen. This system architecture is most akin with the pipe and filter architecture.

The most time consuming process for this system is the matching algorithm between the input text and the websites returned to the script by the Google Custom Search API. So that requirement 7 can be fulfilled, rather than match with all 10 of the websites that are returned, the script will only match with the top 5 websites that are returned by the Google Custom Search API which will cut the run time by up to half. Since each query passed into the Google Custom Search API will be cut off after the first 32 words as well as since the Google Custom Search API only allows for a certain number of queries being made every single day, the input text is removed of all the most common words in the english language as well as split into groups of 32 words to maximize the number of relevant words per query while minimizing the number of queries.

Architecture and Design Philosophy

Performance was of utmost importance when formulating the architecture and designing the major process components. It was also very important that the application would consider the entire input text as well as consider the entire internet's collection of texts. Rather than splitting the input text into 32 word queries, splitting the input text into sentences and/or preprocessing the input text using natural language algorithms might have resulted in more accurate queries and therefore more accurate results, using more queries overall. However, the Google Custom Search API only allows for a limited number of queries each day so a compromise was made to create as few queries as possible while considering the entire input text. This was done by eliminating the most common words in the english language from the input text and making the queries as long as was allowed, 32 words. To use shorter and more queries would have also potentially resulted in more websites that had to be passed through the matching algorithm, which took up the majority of the runtime and would have hindered performance.

It was briefly considered at one point to create a search engine for the sole purpose of this application so as to not be restricted by the daily query limit. This thought was quickly discarded after considering the size of the relational database that would be required to store each website and all the words that would be found on each website as well as the time it would take to crawl through every website that existed.

If, in the future, additional user capacity or system load is required, additional web servers may be enlisted to host the same script. Because the Google Custom Search API only allows for 100 free search queries per day and up to 10,000 search queries per day with billing of $5 per 1,000 queries, increasing the load to allow for multiple users making frequent use of the Plagiarism Checker application will definitely require an implementation of a search engine. Doing so will require a relational database with a table which contains an entry for each website on the internet, a table for every word that can be found on the internet, and a table for each word that can be found on each website. Exabytes of storage may be required for such a database from estimates of Google's search engine database found online. A web crawler will also be needed to continuously update the database and possibly a database to archive past iterations of each website if the application is to fully locate evidence of plagiarism for the condemnation of the plagiarizer.

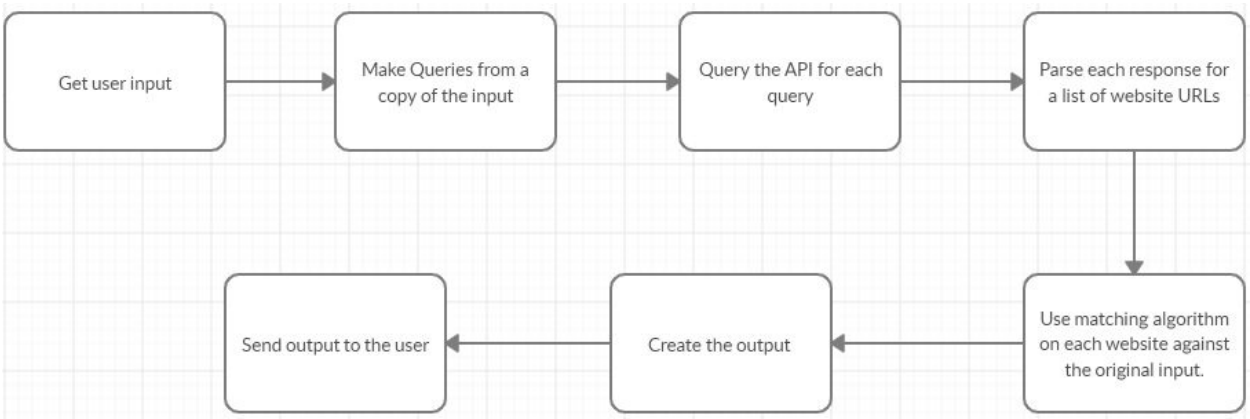
Architectural Views

Logical view:

There are 3 objects interacting with one another in the Plagiarism Checker application. The front-end display on the browser, the back-end processing at the web server, and the Google Custom Search API calls. The user will provide the input from the browser and will submit it into the PHP script hosted at the web server. The script at the web server will process this input into a number of queries which it will send to the Google Custom Search API. The Google API will send each of the responses back to the web server to be processed with the original input from the browser. The web server will finally output the results back to the user on their browser.

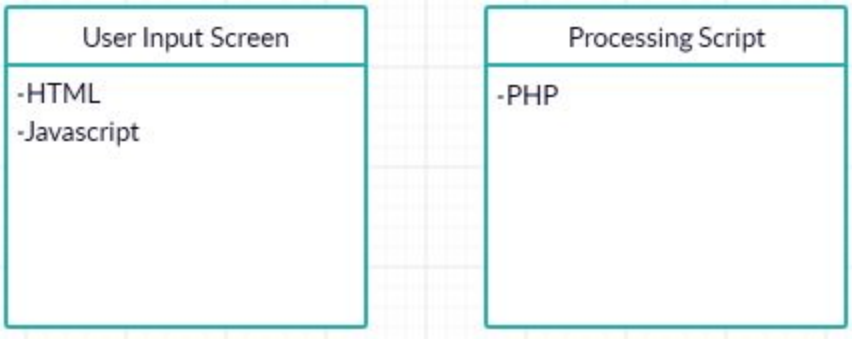
Process View:

The application will get the input from the user. It will then process that input and form it into a set of queries. Each query will be queried into the Google Custom Search API and the API will return a set of responses. The responses will be parsed to get a set of websites. Each website will be parsed to get the text data and matched against the original input from the user to get a set of matching strings and total number of matching words. The resulting information is then outputted to the user.



Development View:

The two major components which have to be implemented are the user input screen on the browser and the script to process the input and output the results. The user input screen will be implemented using HTML and Javascript. The script to process the input and output the results will be implemented using PHP.



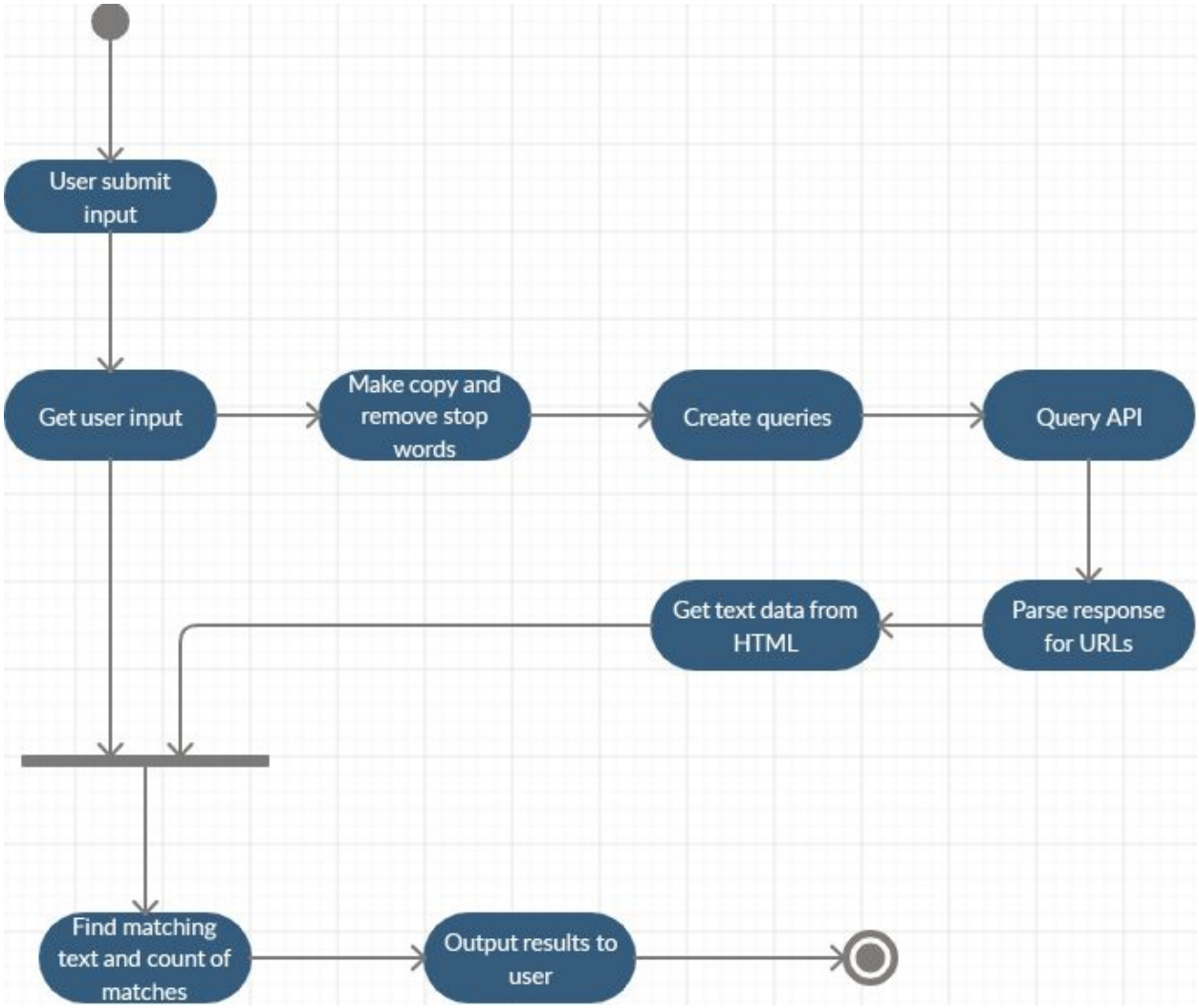
Physical View:

The HTML and Javascript for the user interface and the processing script will both be located at a web server. The user will connect to the web server to get the input screen, to send the input, and to get the results using the internet. The web server will also use the internet to query the API and get the response.

Design Models

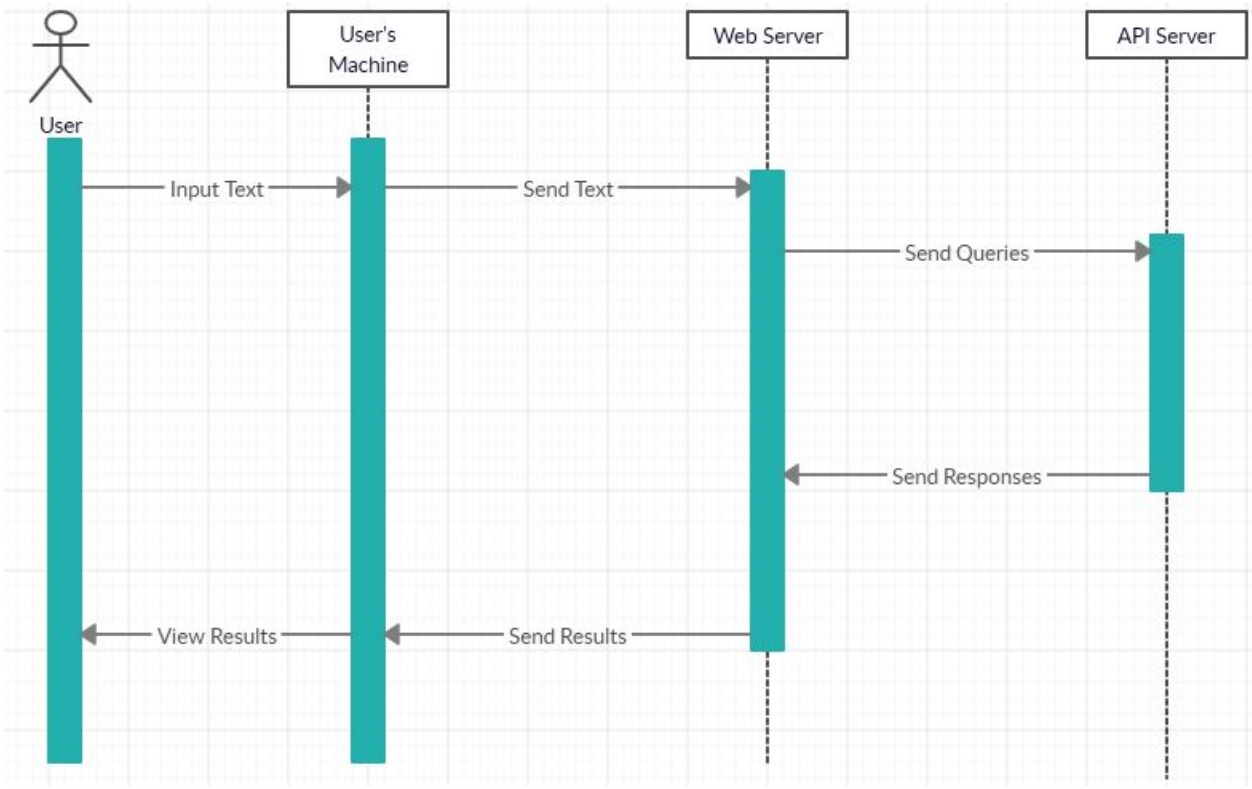
Context Model:

There are two systems being used in this application which are the user input system and the input processing system. The user submits the input from their machine. The input processing system will get the input from the submission and process it.



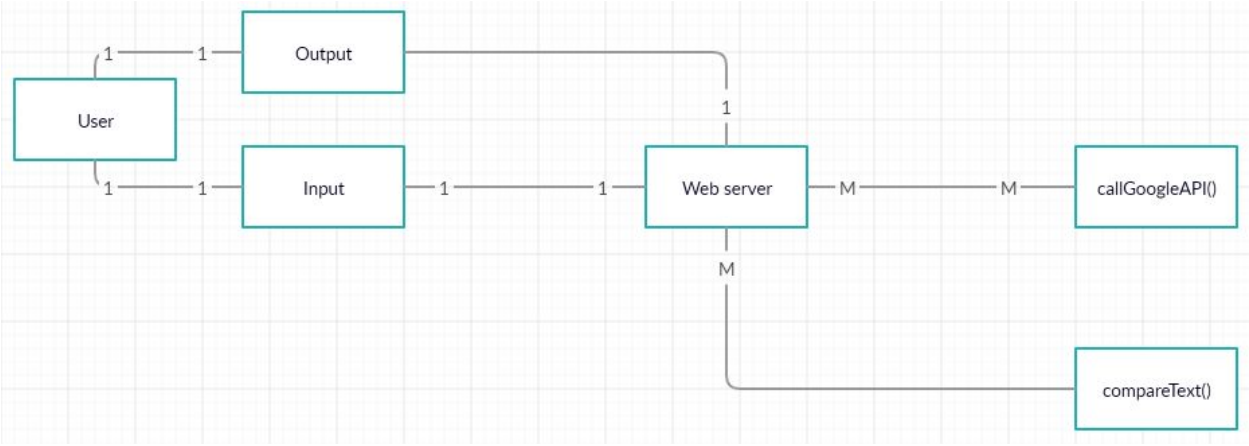
Interaction Model:

This application will be comprised of one actor, the user, and three agents, the user's machine, the web server, and the Google Custom Search API server. The user will only be interacting with their machine to access the user input system. The web server will be interacting with the user's machine for input and output as well as the Google Custom Search API server to get data from the internet for processing.



Structural Model:

For any given usage of this application, the user will send only one input. This input will be split into multiple queries, therefore multiple queries will be sent to the API server and multiple responses will be sent back. Multiple URLs will be parsed from all the responses to be passed into the text matching function. The output function will form a single output from all the data and send it to the user.



Behavior Model:

The input processing script on the backend will begin execution when the user submits the input. The backend will get the input data and create the queries. When the queries are created, they will then be sent to the API to get the response. When the script gets a response, the response will be parsed to get the URLs. When all the URLs have been found, each URL will then be passed into a function to compare the website text with the input text. When all the websites have been compared, the script will output the results to the user.