CS/SE 4352.0U1

Team Send Help

Project Phase II Final

Fareen Chowdhury
Fxc150930

Ross McNew
Rlm160030

Midhat Wahab
Maw160030

https://github.com/fareench/CS_4352_Project
http://www.utdallas.edu/~rlm160030/search_engine.html

1. Introduction

1.1 Project Overview

The goal of this project is to create a program that replicates the use of a search engine. The program should allow the user to interact with a functioning web search engine, Cyberminer. Cyberminer will accept a list of keywords using object-oriented architecture. The keywords will be read into the system and stored in the memory. These keywords will then be used to search for existing URL links that lead to pages containing these keywords. These links will be returned to the user to select from. Concurrently, Cyberminer will have another system called the KWIC (Key Word in Context) system as a subcomponent. The KWIC system will accept a set of lines where each line contains a URL and a descriptor. The URL specification and format will be described in the functional requirement section below. The descriptor should hold relevant keywords that can be used to return corresponding URL links to the user upon search. The KWIC system will perform the circular and alphabetical shift function on a given query from the user. The KWIC index system shall accept no more than 1 ordered line, where each line is an ordered set of no more than 30 words and each word is an ordered set of characters. The number of characters shall not exceed over 300 characters per line. Each line shall be "circularly shifted" by repeatedly removing the first word and appending it at the end of the phrase. Once the circular shift function has been performed, the program must output the results on the screen step by step for the user to view. The system should eliminate any noise words such as "the, a, or, of" etc. before passing the data to the alphabetical function.The data should then be carried over to the alphabetical function. Once the alphabetical function is performed, the KWIC index system shall output a listing of all lines in ascending alphabetical order with their corresponding URL links. The program should then provide the option for the user to restart the program or terminate it.

1.3 Project Progress and Dependency

Part II of this project builds upon phase I of the project: the KWIC System. The Cyberminer uses the KWIC software system as a component in order to keep track of the database made by our team, which contains descriptors and their URLs.

1.4 References

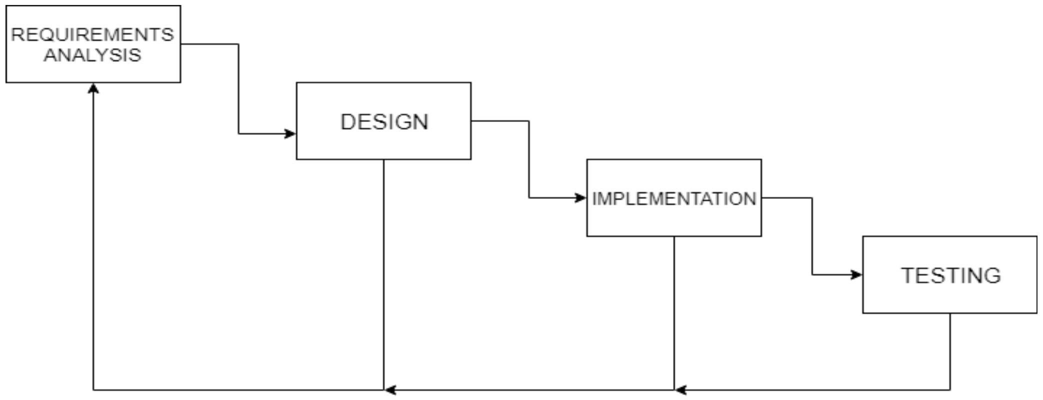
Project Phase II: Cyberminer Software Architecture
http://utdallas.edu/~chung/CS4352/Project2.pdf
Course Homepage
http://utdallas.edu/~chung/CS4352/syllabus.htm

1.5 Definitions/Acronyms

KWIC	Key Word In Context
CYBERMINER	Web Search Engine
CIRCULAR	Cyclic Pattern
SHIFT	Incrementing or decrementing
INPUT	User data
ALPHABETICAL	Relating to the alphabet
APPEND	Attach
DESCRIPTOR	A piece of text and the URL that leads to the page that the text describes
URL	List of string predefined as having the template: 'http://' identifier '.' Identifier '.' ['edu' | 'com' | 'org' | 'net']

2. Project Organization and Managerial Process

2.1 Process Model



Waterfall Model- this model will be used to individually develop each stage of our program in order to keep within the time constraints. The Design and Implementation stages will be repeated until thoroughly developed, and will be tested repeatedly throughout.

2.2 Organizational Structure

I. Fareen Chowdhury
II. Ross McNew
III. Midhat Wahab

Deliverable	**Team Leader***
Interim Project II	I
Final Project II	II

* Team Leader will be in charge of submitting required documents

2.3 Project Responsibilities

Responsibilities and tasks are evenly distributed amongst the team members. Each team member is subjected to fulfilling their assigned responsibilities in a timely manner and within the allotted budget.

2.4 Management objectives and priorities

<u>Objectives:</u>

Perform required task description

<u>Priorities:</u>

- Delivered on time
- Functional and compatible
- Meets all requirements
- Not exceeding costs
- Efficient and of high quality
- Fulfills customer satisfaction

2.5 Assumptions, dependencies, and constraints

Users will have a brief understanding of KWIC system and the implementation.
Dependency: User input
Constraints: Final due date is on July 30th

2.6 Risk Management

1. Miscommunication
Risk: Low

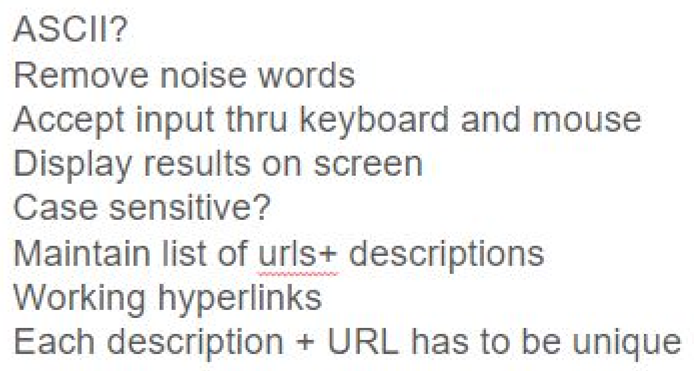
2. Time Manageability
Risk: Medium
Description: Short summer semester time constraints

3. Scheduling
Risk: Medium
Description: Conflicting class and work timings

3. Project Details

3.1 Requirements Specifications and Extensions

Functional Requirements:


ASCII?
Remove noise words
Accept input thru keyboard and mouse
Display results on screen
Case sensitive?
Maintain list of urls+ descriptions
Working hyperlinks
Each description + URL has to be unique

- The system, Cyberminer, will accept keywords from the user as an input and generate a corresponding URL links containing the keywords entered.
    - Cyberminer will perform a case sensitive search.
    - Cyberminer will retrieve working URLs that will take the user to an existing website.
    - The system should allow user to specify the search by using "OR/AND/NOT". These words will be used as a filter to increase the chances of accurate results.
    - The system should allow multiple web search engines to run concurrently.
    - The system should maintain the relevance and accuracy of the links and delete any link that may be outdated or no longer relevant.
    - The system should allow the user to view the number of results on each page and access every page of the web search.
    - Just like a common search engine, the system should autofill the search for user and correct any typographical errors made by the user.
- The system, Cyberminer, will work simultaneously with a KWIC (Key Word In Context) system that will maintain a database with all the URL links that will exist in the system. Each URL link will have its corresponding descriptor.
- The KWIC system will read in ordered set of lines where each line contains a URL identifier and a descriptor with information about the URL.

  - Each URL will start off with “http://” and end with one of the common URL extensions such as “edu, gov, net, com”. Between the start of the link and the extension, the URL must have a descriptor that identifies the URL and makes the web search engine accessible to the URLs that exist in the system.
  - The identifier may contain letters, which can be both lower or upper case and numeric digits of all kind.
  - The KWIC system will perform a circular shift function and alphabetical function on the identifier of the URL for efficiency.
  - The system will accept no more than one input at a time, each input is 30 words long with no more than 300 characters per line.
  - The circular shift will take the first word of the line and append it to the end of the sentence until the very last word has become the first word of the sentence.
  - The noise eliminator will remove all the noise words such as “or, of, the, a” for rigorous results.
  - The alphabetical shift will rearrange the sentence alphabetically so the matching of user’s keywords is faster and more efficient.
- The result is stored and displayed on the screen. The user is prompted to add another line of sentence or terminate the program. If another line of sentence is entered all the functions will be performed on it accordingly and the result of the new line will be displayed with the result of the previous function. The alphabetizer function will alphabetize the overall result of ALL the attempts made using the system and display an alphabetized list of every sentence.

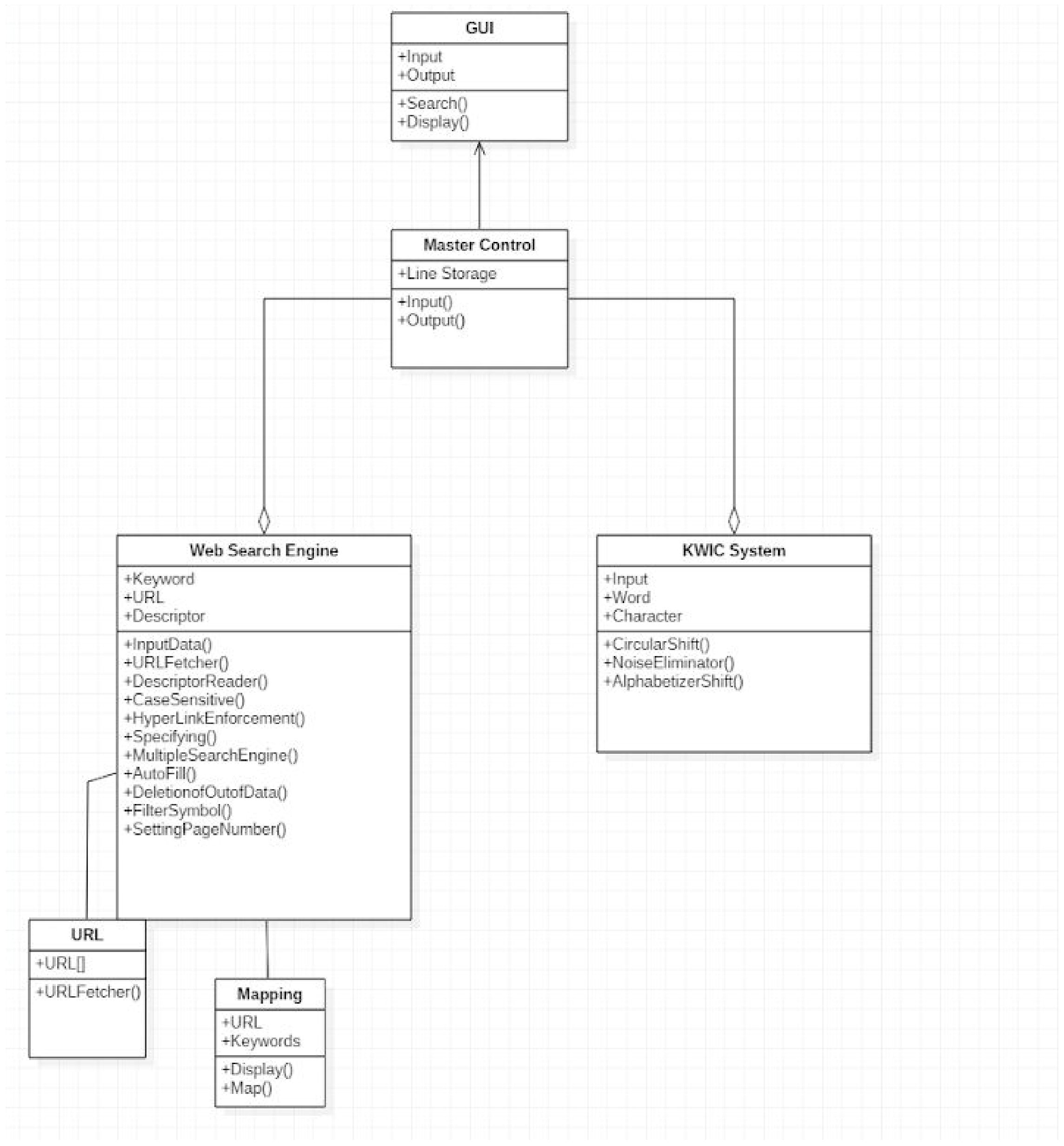
Non-Functional Requirements:

- Cyberminer should be understandable: The user should focus on entering a sentence and clicking a button to run the search engine once the sentence is fully typed. While the user is typing the sentence the system should autofill the search and correct any typos made on user’s end. User should see the “Search” button right next to the search bar to easily access it.
- Cyberminer should be portable: The web search engine should be pulled up on any type of device such as Macbook, Windows, Mobile Version. The search engine should work on any browser such as Chrome, Internet Explorer, Firefox.
- Cyberminer should be enhanceable: Encapsulation, a fundamental part of object-oriented programming, might help in this case. The program should depend on large-scale components instead of smaller, internal or implementation details. The system shall also

be easily adjustable or can be built upon to add more functions without changing much of the core program.

- Cyberminer should be reusable: The system is built on object-oriented architecture and the system should be able to reuse part of it for different functionality of the project for better use.
- Cyberminer should have good performance: The system include noise eliminator after the circular shift so the data that is carried over to the alphabetizer function has fewer words to alphabetize which will help the performance overall.
- Cyberminer should be user-friendly: The system should display the results clearly and prompt user to take the next step by generating clear instructions on every part of the process. The user should know where to enter the sentence and click the search button. The user should be given the option to either enter a new sentence or terminate the search. The steps should be clear and easy to perform.
- Cyberminer should be responsive: The system should not take longer than 3 seconds to display the results after the user has entered the input.
- Cyberminer should be adaptable: The system should look similar to most search engines so the user is familiar with it.

3.1b Preliminary Class Diagram



*Class Diagram subject to change as the program becomes fully developed

3.2a Architectural Specification for the KWIC System

Components: The KWIC system from Project 1 will be used and built upon for this project. The following components make up the KWIC System that will allow the user to build the index from which the search engine will yield results. The components or objects that the program depends on:

Input: Receives data from the user through the keyboard/screen interface

Master Control: Will be used to initiate the program.

Output: Will be used to store results in memory in order to be accessed by the Cyberminer system.

Line Storage: Will be used to store the data input in proper structure

Circular Shift: Will perform the circular shift function on direct input into an ordered list

Noise Word Eliminator: Will remove noise words from the input lines. Noise words will be

Alphabetizer: Will alphabetize on direct input into a sorted list

Connections: There exists a connection between all components of the program.

Connection 1: Master Control will call upon Input's setup function to begin the program

Connection 2: Input will wait for data from the input medium

Connection 3: Master Control will call upon Line Storage setup function

Connection 4: Line Storage will receive data from Input & store each line and each word

Connection 5: Master Control will call upon Circular Shift's setup function

Connection 6: Circular Shift will receive its input from Line Storage, perform the circular shift function and store each character, word, and line.

Connection 7: Master Control will call upon Alphabetizer's setup function

Connection 8: Alphabetizer will call upon circular shift to receive the input required to perform an alphabetize function.

Connection 9: Master Control will call upon Output's setup and print function.

Connection 10: Output will call upon data from Circular Shift and from Alphabetizer

Connection 11: Output will send the data, organized, to the output medium.

Constraints:

Several constraints apply to the components of the program:

1) The program cannot accept more than 300 characters in one run. Since it is a low level program with a small size array and in order to allow the program to meet the performance criteria from non-functional requirements, it is necessary to limit the number of characters that will be read into the program.

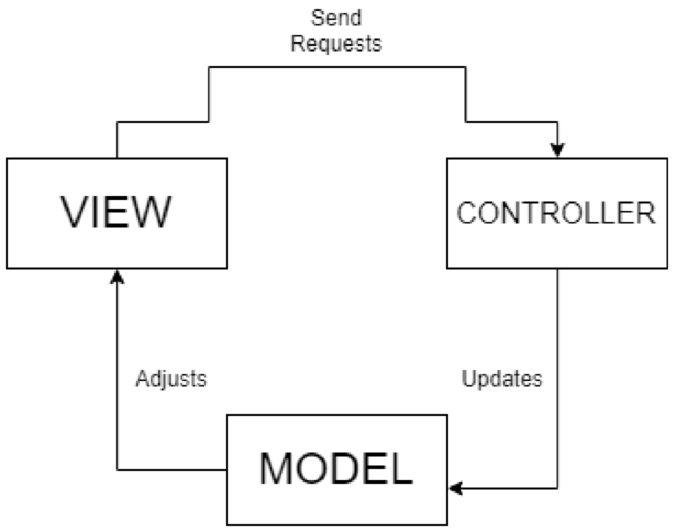
2) The program cannot accept more than 30 words in one run. Since it's a small program the same restrictions that apply to the character limit applies to the word limit.
3) The program will accept 1 sentence at a time. The program is designed to apply the functions after 1 full sentence is entered. Once the results are displayed the program will prompt the user for a new sentence. Multiple sentences cannot be read into the program simultaneously.

3.2b Architectural Specification for Cyberminer

Many styles were considered when researching through the functional and nonfunctional requirements for the project. Two of the top considered architecture patterns are Object-Oriented and MVC (Model-View-Controller).

Model-View-Controller

Rationale: The MVC pattern would be helpful in a client-oriented program, as they would not have to worry about what happens behind the scenes and instead can focus more on the user interface.



The model comprises of data and functions that are used on the input given, the KWIC System, and the Cyberminer implementation. The view sends requests to the controller through user-chosen functions. The view in our case would be the webpage that the user will be accessing. The controller updates the model according to what the user has chosen.

Advantages:

The code will be easy to follow along, as each part will be separated according to function and role in the overall program.

The code will be easily scalable.

The code will be reusable

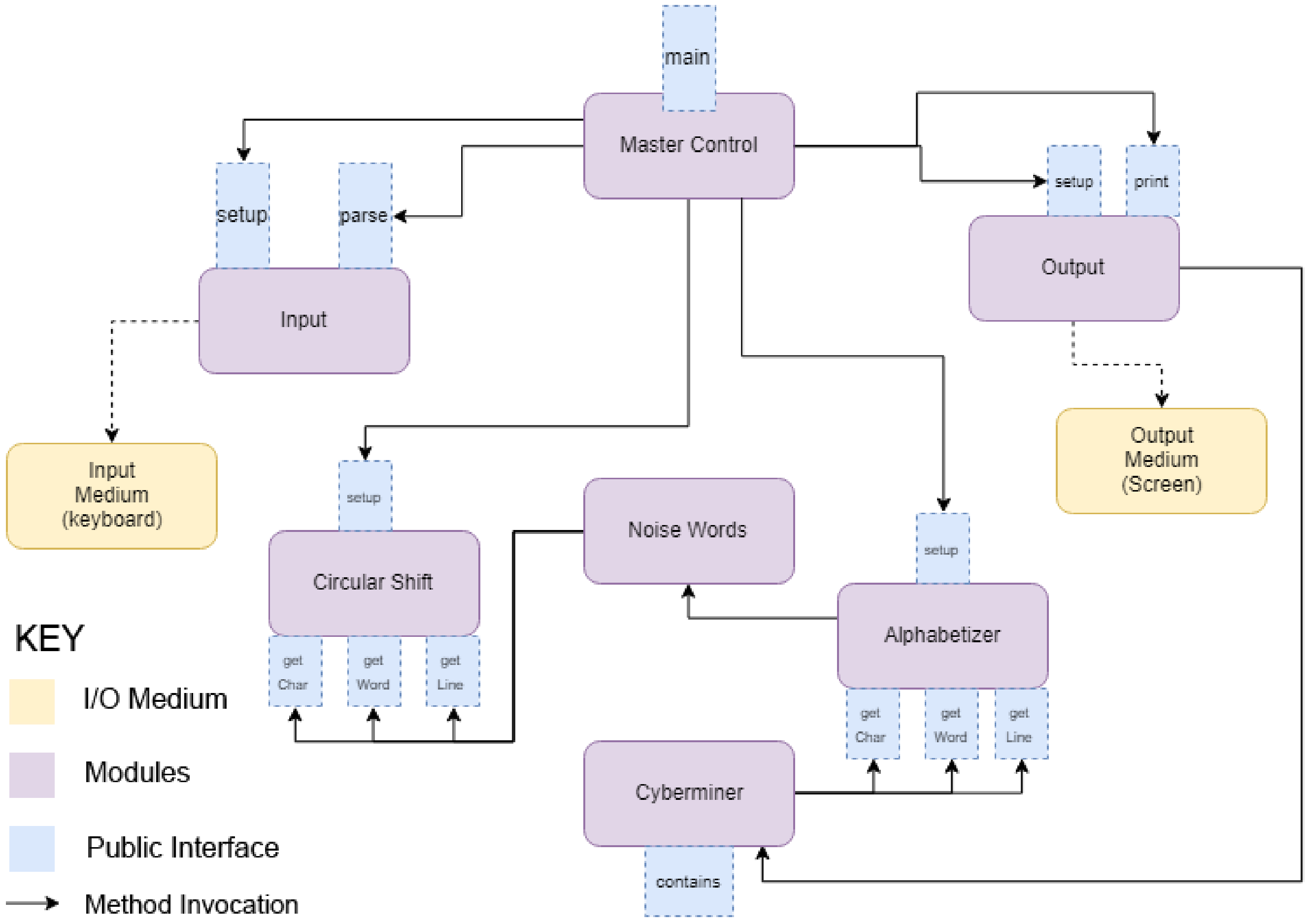
Disadvantages:

Complexity of project increases drastically

Each component must be tested properly and in different “languages”, or developer needs to have knowledge on multiple programming technologies

Abstract Data Type Style

Rationale: The Abstract Data Type style has already been implemented in the KWIC system and can easily be adjusted to use in the Cyberminer system as well.



Advantages:
KWIC System was built with ADT architecture
Code will have clear structure and connections
Code will be easily maintainable and testable
Code will be reusable and remove the need for redundancy

Disadvantages:
Code will use a lot of space, as multiple modules will require a copy of the input data
Response time will also be affected

The following components, connections, and are built upon the 3 C's of the KWIC System, described above:

Components: The components or objects that the program depends on:

Cyberminer: Will match input against keywords and find the closest, if any, matching results and display the link of the result.

Noise Words: All words that our team has decided will the "noise words" or dismissable words that will be removed from the input lines.

Connections: There exists a connection between all components of the program.

Connection 1: Cyberminer will be able to receive input from the alphabetical shift results

Connection 2: Noise words will be accessed by Alphabetical shift in order to remove the noise words from the results.

Connection 3: Output display functions will access the Cyberminer results in order to display results for keywords entered into the input medium,

4. Technical Process

4.1 Methods, Tools, Techniques

J2EE, JAVA, Javascript

4.2 Software Documentation

Final and public documentation will be pushed onto Github and private documentation will be edited on a private Google Drive.

5. Table with Deliverable and Due Dates

Deliverable	Due Date
Interim Project II	July 11th
Final Project II	July 30th

Our team has worked hard and long on this assignment, and had our ups-and-downs. Learning to adjust what we already know in order to implement Javascript was a difficult endeavor and one that we will continue to work on in the future. Still, our team did not give up and supported each other through the assignment or even simply emotionally.

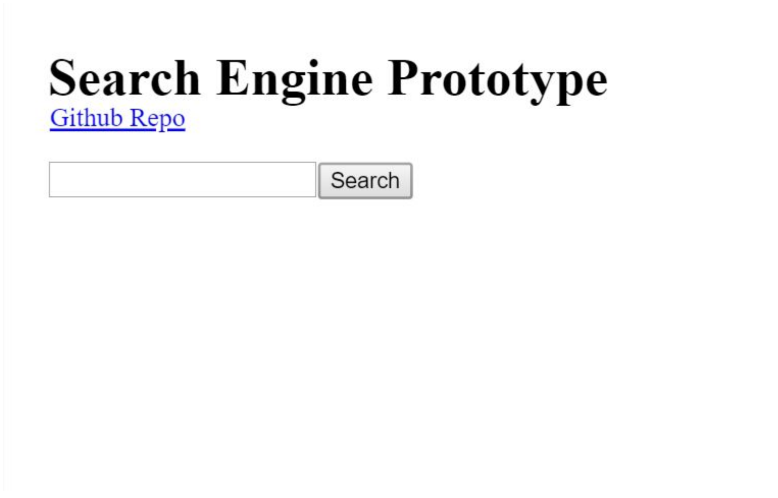
Mock User Manual

This document will help users follow the steps to access our program as well as achieve the desired outcome of the program.
Note: Current pictures are placeholders until html page is finalized.

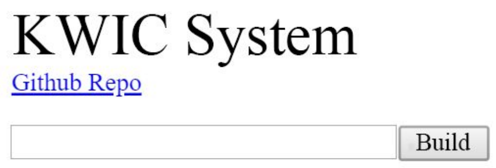
Note (2): URLs used in this case are the following (and are recommended to be used by other users)

- CS 4352 Website http://utdallas.edu/~chung/CS4352/syllabus.htm
- Testing Center Website https://ets.utdallas.edu/testing-center
- Stack Overflow Website https://stackoverflow.com/
- The University of Texas Homepage https://www.utdallas.edu/
- The University of Texas Admissions https://www.utdallas.edu/admissions/
- The University of Texas Academics https://www.utdallas.edu/academics/
- The University of Texas Galaxy https://www.utdallas.edu/galaxy/

To access the program, users must visit the following link:
http://www.utdallas.edu/~rlm160030/search_engine.html
Once you click on the link it will look like this:



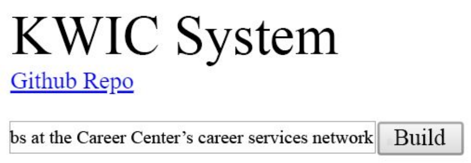
Users will click on the hyperlink that will be added to access the KWIC System:



KWIC System

Github Repo

The KWIC System page is intended to allow the building of an index by repeatedly entering URLs and descriptions in the following format
URL|description
URLS will follow the format: ‘http://’ identifier ‘.’ Identifier ‘.’ [‘edu’ | ‘com’ | ‘org’ | ‘net’]
Descriptions can be a length of strings, characters, and numerical values.
In this example, the entered text reads: https://utdallas.joinhandshake.com|Find jobs at the Career Center’s career services network.

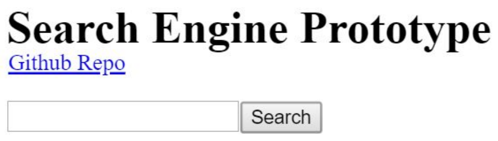


KWIC System

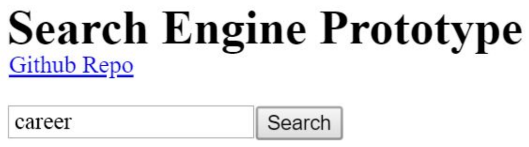
Github Repo

bs at the Career Center’s career services network

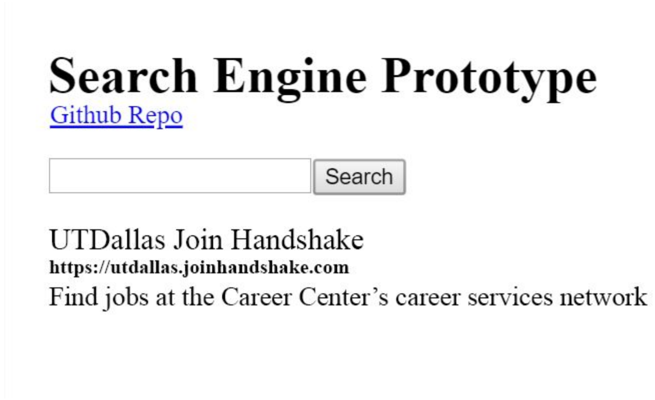
Clicking Build will update the index and allow the user to enter more URL and descriptors. A hyperlink will be added to take the user back to the Cyberminer page.



At this page, the user can type in keywords or phrases that will be used to search through the index.



Clicking ‘search’ will go through the index to find the entered text in any of the descriptions or URLs. The following result will show up.



UTDallas Join Handshake
https://utdallas.joinhandshake.com
Find jobs at the Career Center’s career services network