# **Project II: Cyberminer**

CS 4376.0U1

Project Phase 2 - Project Plan

Team Name: Summer Searchers

Team Members:

Yong hyun Lee (yxl129430, yxl129430@utdallas.edu)

Matthew McCoy (mom210002, mom210002@utdallas.edu)

Fionne Tran (fxt190001, fxt190001@utdallas.edu)

Andres Mancillas (axm180134, axm180134@utdallas.edu)

Parsa Zilaee (pxz170230, pxz170230@utdallas.edu)

Alaa Jalali (amj180003, amj180003@utdallas.edu)

Team URL: https://github.com/CS4376/Cybermining

# **Table of Contents**

[**Project II: Cyberminer 0**](#_gcw0bmlfpyg9)

[**Table of Contents 1**](#_1sitqgb5b97d)

[**Introduction 2**](#_r5yqrvn4nt1s)

[**Project Road Map 2**](#_axlmvstyuxqv)

[**Domain Modeling & Requirements 3**](#_d1zj1lgz2vvg)

[Domain Modeling 3](#_r7f63k93ybds)

[Requirements 3](#_preox5fazjac)

[Extensions and Clarifications to Functional Requirements 3](#_vxxf6cdxg9x3)

[Extensions and Clarifications to Non-Functional Requirements 4](#_yr1fajrz33mu)

[**Architectural Design Specification 5**](#_lkbhk1abf2mp)

[Use Case Diagram 5](#_iiyul8ocpoq9)

[Sequence Diagram 6](#_8fxsorf6l1zb)

[Class Diagram 7](#_l458zy9hqxfc)

[State Transition Diagram 8](#_56u7n6ndygwy)

[Activity Diagram 9](#_dr9le8903nvs)

[**Prototype Implementation 10**](#_zgd95j93orxd)

[Example Screenshots of Cyberminer 11](#_u4oujxvhzo77)

[Limitations of Cyberminer 12](#_rh5avr7mibjz)

[What’s next for Cyberminer? 13](#_s9fsqkoc3kk6)

[**Group Member Roles for Project 2 14**](#_jbekx98jvwcv)

[**References 15**](#_1id73px4x9x)

# 

# 

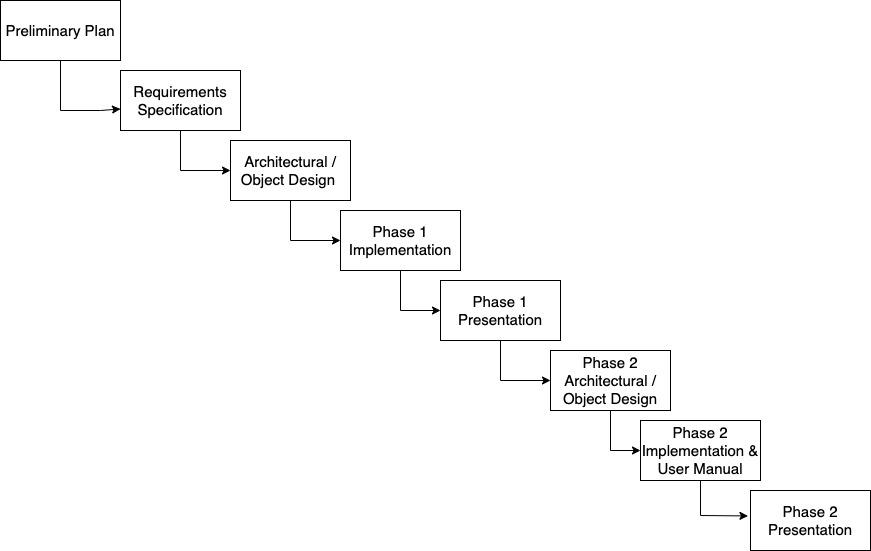
# 

# 

# **Introduction**

Cyberminer is a web search engine. For example, Cyberminer has hyperlink enforcement, accepts user input (number of results, keywords), performs searches, and displays search results. To create Cyberminer, we utilized object-oriented analysis and design principles. To create the UML diagrams, our team used Astah. Our programming language of choice for creating our prototype was Python. The contents of our project can be accessed through the team’s web page (GitHub) and/or device (e.g. PC, laptop)

# **Project Road Map**



# **Domain Modeling & Requirements**

## Domain Modeling

* Organizational/Individual Users, Cybermining Administrations, advertisers, researchers, data analyzers, competitors

## Requirements

### **Extensions and Clarifications to Functional Requirements**

**The Cyberminer System shall:**

* Accept a list of keywords and return matching URLs
* Support case-sensitive search and preserve input format
* Enable hyperlink enforcement for easy website navigation
* Allow specifying OR/AND/NOT search modes
* Run multiple search engines concurrently
* Delete outdated URLs and their descriptions
* List query results in ascending alphabetical order or other specified criteria
* Provide options to set the number of results per page and navigate between pages
* Implement autofill and typographical error correction
* Filter out non-meaningful symbols based on user configuration

1. Input and Output Handling: The input of Cyberminer should come from a user interface, whether it's a website or a command line, where keywords can be entered by the user. The input is processed and displays a list of URLs that can be seen by the user in the format of a website or a downloadable file.

2. Stakeholders and Different Features: Organizational users, individual users, advertisers, and Cyberminer administrators are some of the stakeholders in this project. The specific needs and requirements of each stakeholder group are taken into consideration. Different features or functionalities should be provided to cater to distinct requirements given by stakeholders. For example, organizational users might require advanced search options or analytics features, while individual users may prioritize ease of use and personalized settings.

3. Non-Meaningful Symbol Filtering: Users should be able to choose which symbols or characters that can be filtered out from the search keywords, and this can be done using a configuration option. This feature ensures that irrelevant symbols do not affect the search results.

### **Extensions and Clarifications to Non-Functional Requirements**

**The Cyberminer System shall:**

* Cyberminer should be easily understandable, portable, enhanceable, and reusable
* Ensure good performance for efficient search operations
* Design the system to be user-friendly, responsive, and adaptable
* Maintain the KWIC system's usability and readability

1. Understandable and User-Friendly: To make sure users can properly interact with Cyberminer, it should have clear and intuitive user interfaces, error messages, and documentation. The system should provide appropriate feedback and guidance to users during their interactions.

2. Portable and Reusable: For Cyberminer to be portable and reusable, it should be compatible with multiple platforms and operating systems. This allows for easy deployment and integration with different environments. It is also important to use modular and well-documented code to promote reusability and maintainability.

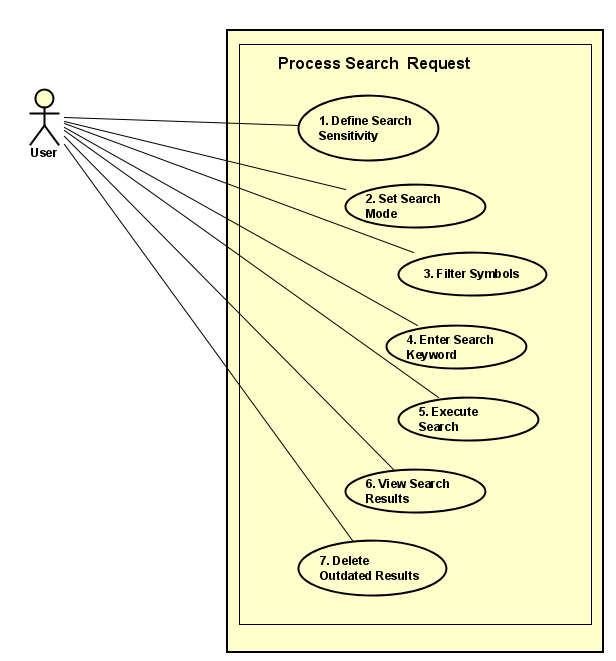
3. Enhanceable and Adaptable: The system should be designed in a way that allows for easy enhancements and modifications in the future. This can include using modular architecture, well-defined APIs, and proper documentation to facilitate future development and integration of new features.

4. Good Performance: Specific performance targets such as response time for queries, throughput (number of queries processed per unit of time), and scalability (ability to handle increased workload or data volume) can be provided to show the performance of Cyberminer.

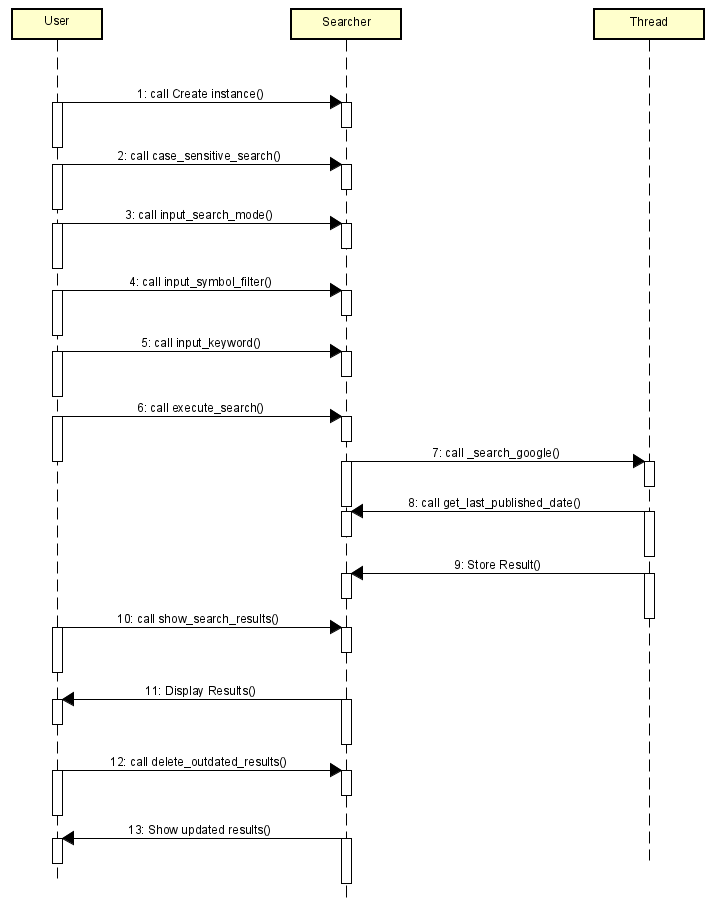
5. Responsive: It’s the system's responsiveness in terms of user interface interactions. Cyberminer should provide prompt and smooth responses to user actions, such as submitting a search query, navigating between pages, or adjusting settings.

# **Architectural Design Specification**

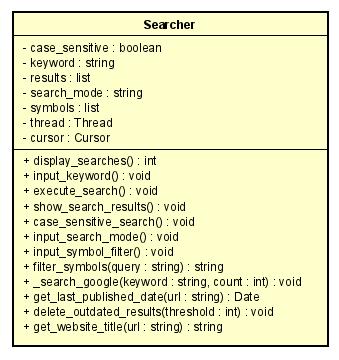
## Use Case Diagram

****

## Sequence Diagram

****

## Class Diagram

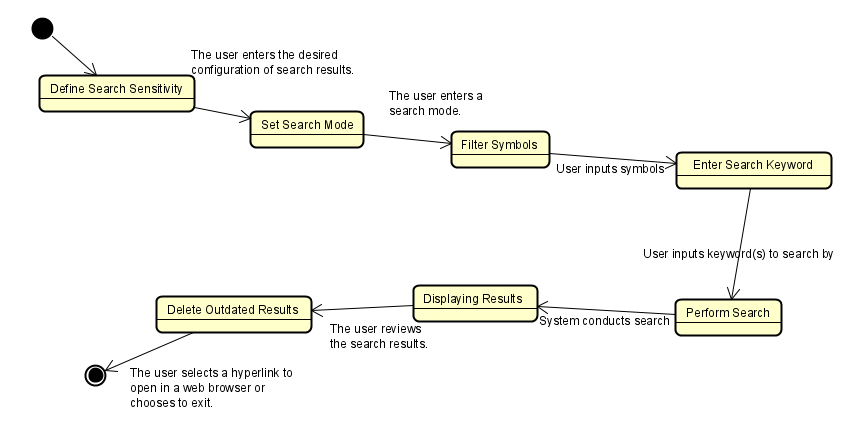
****

## 

## 

## 

## State Transition Diagram

****

## Activity Diagram

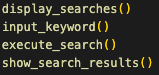
## 

# **Prototype Implementation**

How does the prototype work?

**A: The prototype works by utilizing Google to search.**

****

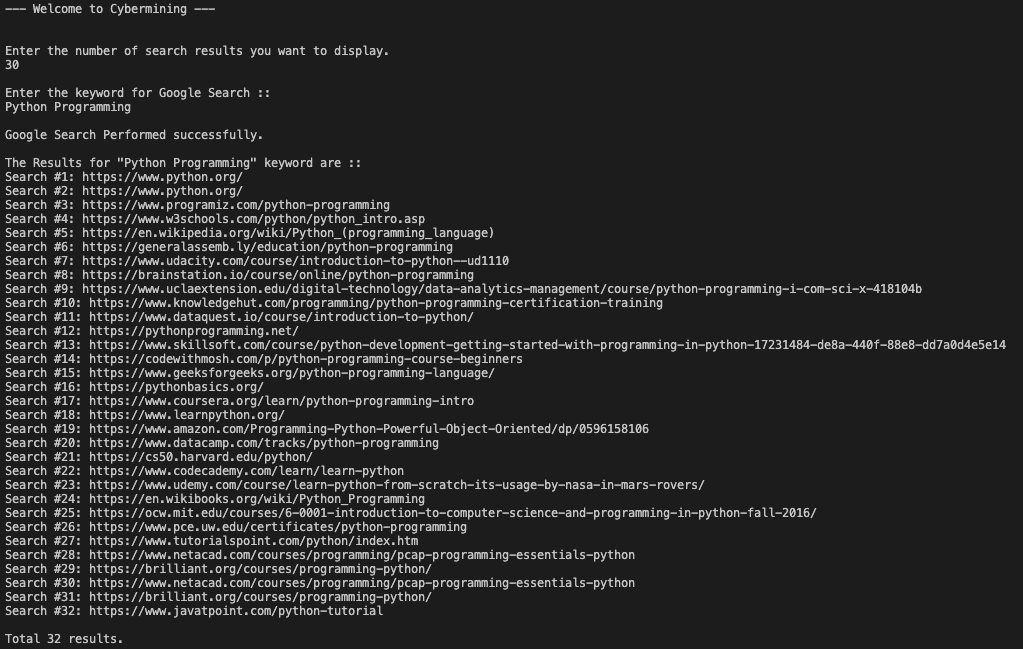
****

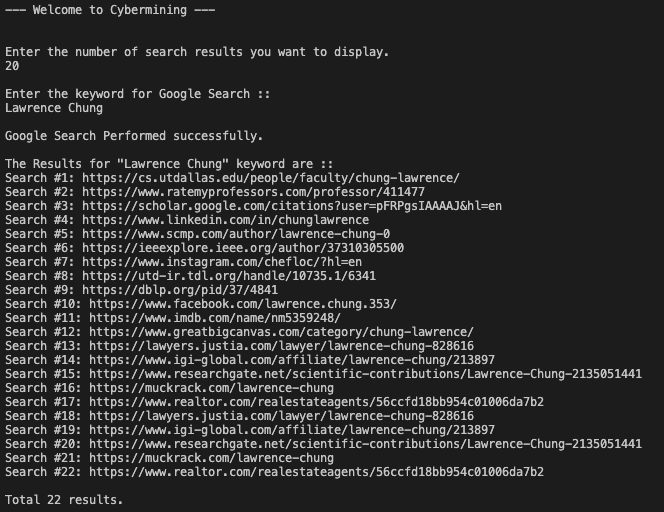
What can the prototype currently do?  
  
**A: The prototype can currently accept user input (number of results, keywords), perform searches, and display search results.**

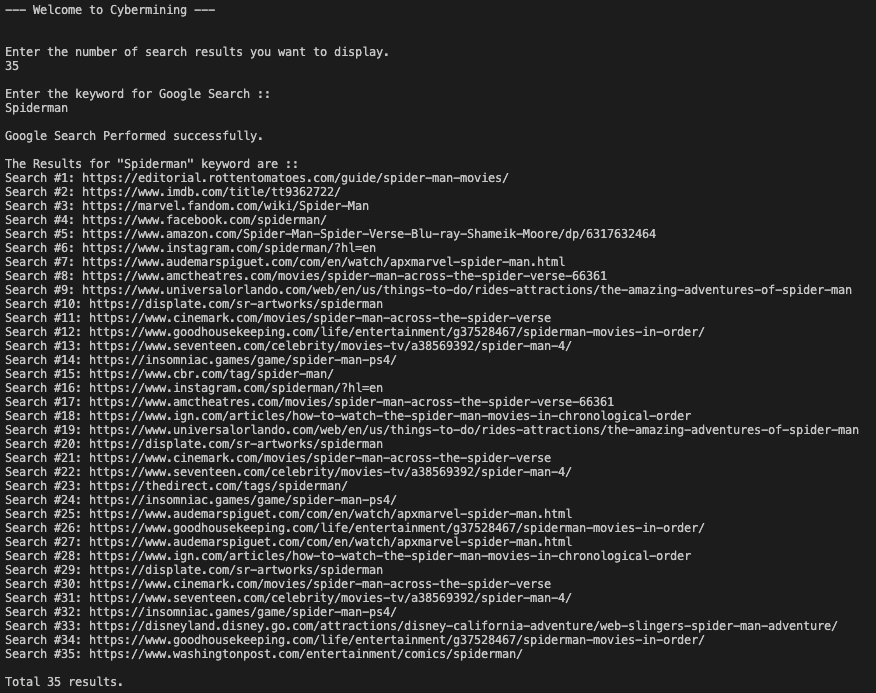
**As of June 28, several features have been added to Cyberminer:**

* **Hyperlink enforcement, which is activated by selecting an index to open a URL and opening it with the default web browser.**
* **Specifying the search mode with the keywords AND, NOT, OR.**
* **Case sensitive search**
* **Implementing multiple search engines concurrently**

## Example Screenshots of Cyberminer

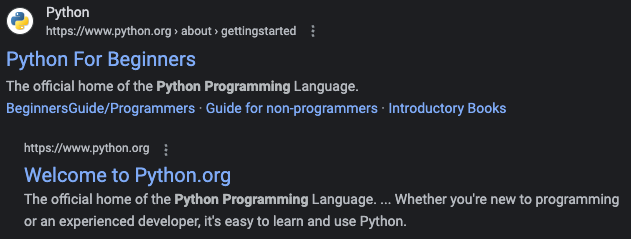


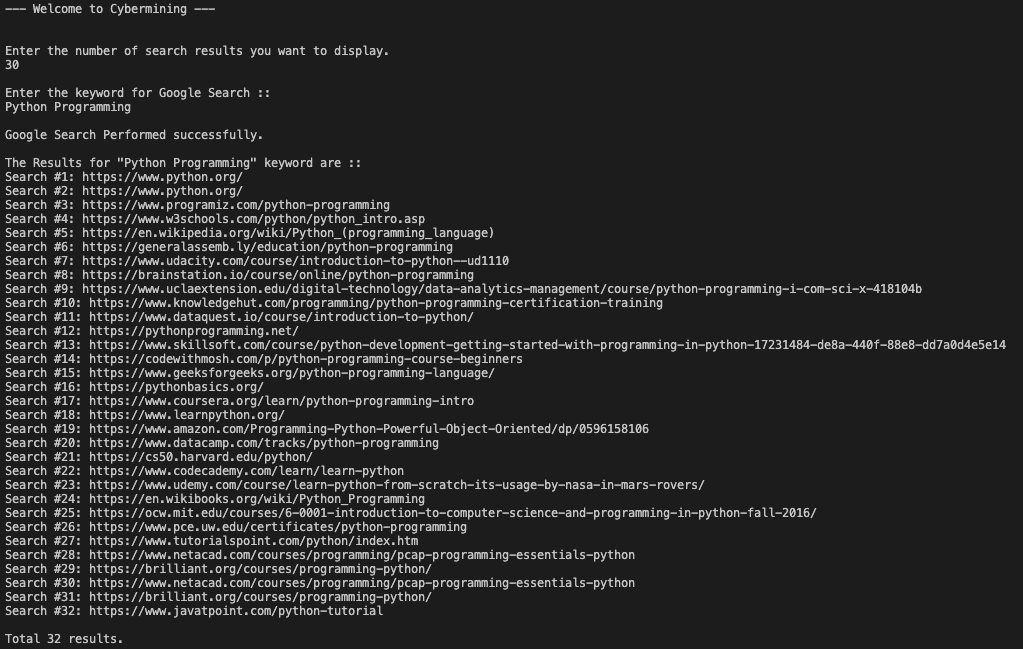




## Limitations of Cyberminer

The limitations of Cyberminer are that mismatches involving the number of results can occur. For example, 32 results would be displayed instead of 30 results since there is a duplicate result.

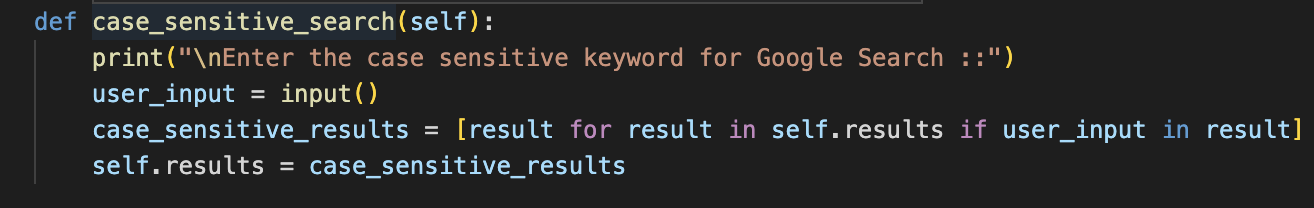




## What’s next for Cyberminer?

**To improve our prototype our next central focus will be :**

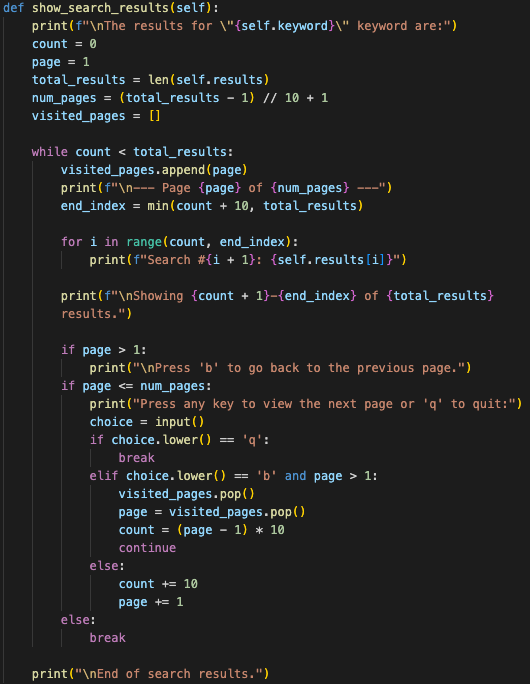
* Select function out of each iteration to see meta-information of the selected number of URLs



**Navigating between pages**

* Setting the number of results per page = 10.
* Users decide whether to go back or forth of the page.
* Displays current pages with total pages.
* Easy to add further implementation.





# **Group Member Roles for Project 2**

* Andres Mancillas & Alaa Jalali: Domain Modeling & Requirements, Project 2 Deliverable, and Project Road Map
* Matt McCoy: Architectural Design Specification
* Parsa Zilaee & Yong hyun Lee: Prototype implementation (Back-end design)
* Fionne Tran: Implementation of front-end design and User Manual

# **References**

[1] L. Chung, “Project1.” The University of Texas at Dallas, Richardson, May. 24, 2023