**Cyberminer**

**Team LPNX**

Xinhe Chen

YuWei Pai

Keith Nguyen

Hengbo Liu

**SE 6362.001**

**Preliminary Project Management Plan**

**Team Lead (Rotated):**

Team lead will be rotated by week.

**Team Members:**

Xinhe Chen (xxc170630)

YuWei Pai (yxp170730)

Keith Nguyen (kxn161730)

Hengbo Liu (hxl170010)

Roles:

|  |  |
| --- | --- |
| Xinhe Chen | Architectural Specification |
| YuWei Pai | Non-functional requirement  Manual |
| Keith Nguyen | Introduction/Functional Requirements  Specification of Program |
| Hengbo Liu | Cyberminer code |

**Team Website:**

**https://github.com/xinhechen/KWIC**

# 

[**Introduction**](#_k18wmq2rjupy) **4**

[**Requirement Specification**](#_8yxndes1m7ev) **4**

[Functional Requirements](#_gt0k4gqf92wx) 4

[Non-Functional Requirements](#_sf54ssxadaej) 7

[**Architectural Specification**](#_4igp9ydkd6k) **12**

[Architecture 1](#_p4rhm58ztc5w) 12

[Non-Functional Requirements](#_nghew8211l81) 13

[Architecture 2](#_bs5dhlowdt95) 15

[Non-Functional Requirements](#_77n3x8fdr4f) 15

[Architecture 3](#_ew9kqlpvroz4) 17

[Non-Functional Requirements](#_pmdbrafywu1v) 17

[Architecture 4](#_5cdivjewrde3) 19

[Non-Functional Requirements](#_c8mglcjr5oyd) 19

[Conclusion](#_l6bb7b9bcuz) 21

[**Specification of program**](#_9i92opwr17) **22**

[Program Objectives](#_cqye5qw8tukg) 22

[Output Specification](#_x6e65rr9kkxu) 22

[Required Input](#_j3kh76atczcu) 22

[Processing Requirements](#_1xydp6djcbwg) 29

[**Manual (How to install and run an example)**](#_dvnwrjx1asho) **29**

# 

# Introduction

This is the phase two of the semester project. This phase shall build on top of phase one KWIC system to create a web search engine called Cyberminer. Implementing this phase requires the use of an Object-Oriented architectural style and build a Java applet or a similar application. Cyberminer shall be hosted on the team’s webpage. The program shall satisfy both functional and non-functional requirements listed below by implementing different features.

# Requirement Specification

This part contains the functional requirements and non-functional requirements of the project.

## Functional Requirements

The functional requirements list describes what Cyberminer should do. Since Cyberminer uses KWIC, this also contains the functional requirements of KWIC. The list is as below:

1. Cyberminer shall accept a list of keywords and return a list of URLs that contains any of the given keywords.
2. Cyberminer shall use KWIC system to maintain a database of URLs and their descriptions.
3. The KWIC system shall receive a set of lines. Each line shall contain two parts: URL and the descriptor.
   1. The URL shall be in this format: http:// {identifier} . {identifier} . {edu | com | org | net}
   2. The identifiers can be a set of letter and digits: a-z, A-Z and 0-9.
   3. The descriptor shall have this syntax: identifier{‘ ‘identifier}\*
   4. The descriptor shall be circularly shifted by repeatedly removing the first word and appending it to the end of the line. Noise words shall be removed.
4. Cyberminer shall do the following:
   1. Allow case-sensitive search
   2. When the user click on the URL, the system shall open the link to the corresponding website.
   3. Cyberminer shall specify AND/OR/NOT search options.
   4. Multiple instances of Cybermine can be ran concurrently.
   5. Out-of-date URLs shall be deleted.
   6. Listing of result shall be displayed in different orders: alphabetical, most frequently accessed.
   7. The interface shall allow setting of how many results to show per page and allow navigation to next page.
   8. Cyberminer shall allow autofill while correcting typographical error and filtering out symbols that are not meaningful (not in the alphabet or a number)

|  |  |
| --- | --- |
| **Issued ID** | FR1 |
| **Name** | Must include the required components below |
| **Description** | * The system shall have a web interface. The system shall be able to run on Chrome and Firefox. * The system shall implement ADT architecture and functions. |
| **Features** | * The KWIC system has a website. * The website runs the same way on Chrome and Firefox. * The system implements the ADT architectures and functions as shown in the diagram. |

|  |  |
| --- | --- |
| **Issued ID** | FR2 |
| **Name** | Inputs |
| **Description** | System shall accept the listed inputs.. |
| **Features** | * The system shall accept an ordered set of line. * Each line is an ordered set of words. * Each word is an ordered set of characters. * The system shall be able to accept multi-line input. |

|  |  |
| --- | --- |
| **Issued ID** | FR3 |
| **Name** | Operation |
| **Description** | The system shall repeatedly remove the first word and appending it at end of line. |
| **Features** | * The getchar, setchar, CS-char, CS-word functions shall perform the actions listed above. |

|  |  |
| --- | --- |
| **Issued ID** | FR4 |
| **Name** | Output |
| **Description** | The output shall list all circular shifts of all lines in ascending alphabetical order. |
| **Features** | * The system shall sort the output in ascending alphabetical order and list the output. |

|  |  |
| --- | --- |
| **Issued ID** | FR5 |
| **Name** | AND/OR/NOT operation |
| **Description** | Cyberminer shall specify AND/OR/NOT search options. |
| **Features** | The system shall accommodate symbols AND (a space), OR (“|”) and NOT (“!”) in input |

|  |  |
| --- | --- |
| **Issued ID** | FR6 |
| **Name** | Deleting URL |
| **Description** | Out-of-date URLs shall be deleted. |
| **Features** | There shall be an option to delete URL/keyword pair from the web interface. |

## Non-Functional Requirements

Non-functional requirements for Cyberminer system are described as below. The following non-functional requirements are presented. There are five NFR soft goals that the system should accomplish: (1) Portability; (2) Enhanceability; (3) Reusability; (4) Good performance; (5) User-Friendly; (6) Adaptability; (7) Scalability; (8) Reliability; (9) Robustness. Each of the NFR softgoal is described in details.

|  |  |
| --- | --- |
| **Issued ID** | NFR1 |
| **Name** | Portability |
| **Description** | The system shall consider portability issue while developing the system. |
| **Type** | Softgoal |
| **Affecting NFRs / Operationalizations** | * Platform independence - web browser * Standard programming language * Operating system |

|  |  |
| --- | --- |
| **Issued ID** | NFR2 |
| **Name** | Enhanceability |
| **Description** | The system shall able to add additions of system function. |
| **Type** | Softgoal |
| **Affecting NFRs / Operationalizations** | * Noise word eliminator |

|  |  |
| --- | --- |
| **Issued ID** | NFR3 |
| **Name** | Reusability |
| **Description** | The components shall serve as reusable entitles. |
| **Type** | Softgoal |
| **Affecting NFRs / Operationalizations** | * Circular shift * Alphabetizer |

|  |  |
| --- | --- |
| **Issued ID** | NFR4 |
| **Name** | Good Performance |
| **Description** | The system shall be accomplished with limited response time, and sufficient memory space for adding additional users or data calculation. |
| **Type** | Softgoal |
| **Affecting NFRs / Operationalizations** | * Manageable Space * Low Response Time * Increased storage capacity * Use uncompressed/compressed format data * Process and retrieve data in limited time |

|  |  |
| --- | --- |
| **Issued ID** | NFR5 |
| **Name** | User-Friendly Access |
| **Description** | The system shall consider user-friendly access, and easy understanding manual is necessary. |
| **Type** | Softgoal |
| **Affecting NFRs / Operationalizations** | * Easy understanding user interface * User manual |

|  |  |
| --- | --- |
| **Issued ID** | NFR6 |
| **Name** | Adaptability |
| **Description** | The system shall adapt itself efficiently and fast to changed circumstances. |
| **Type** | Softgoal |
| **Affecting NFRs / Operationalizations** | * Component changing will not affect other components |

|  |  |
| --- | --- |
| **Issued ID** | NFR7 |
| **Name** | Scalability |
| **Description** | The system shall be scalable in terms of being able to process large amounts of data. |
| **Type** | Softgoal |
| **Affecting NFRs / Operationalizations** | * Distributed system * Database system |

|  |  |
| --- | --- |
| **Issued ID** | NFR8 |
| **Name** | Reliability |
| **Description** | The system shall be reliable in terms of providing the same set of controls and output to the user, every single time. |
| **Type** | Softgoal |
| **Affecting NFRs / Operationalizations** | * Database system supports atomic inserts and retrievals |

|  |  |
| --- | --- |
| **Issued ID** | NFR9 |
| **Name** | Robustness |
| **Description** | The system shall be robust enough to handle all sorts of erroneous scenarios and provide the corresponding errors. |
| **Type** | Softgoal |
| **Affecting NFRs / Operationalizations** | * Display detailed errors * Display warning messages |

# Architectural Specification

The cyberminer system consists of the KWIC system which allows storing of the data to the database through a webpage and a webpage that allows users to search for results through keywords. Since the system requires a GUI (view) the user can interact with, and it will display the results of the database search (model) based on what the user types in. The Model-View-Controller is considered as the architecture of this system.

## Architecture 1

The first architecture considered is shown in Figure 1. The user interacts with the webpage(view), typing in words they would like to search. The Controller retrieves information from the webpage and manipulates the DB Interactions(Model) to query the information from the database. DB Interactions returns this info to the controller and the controller updates the webpage. The KWIC system also stores data to the database through the controller.

This architecture is the best for the cyberminer system since it is great for enhanceability and reusability. It has some tradeoff with good performance for a controller but it allows low coupling of the modules.

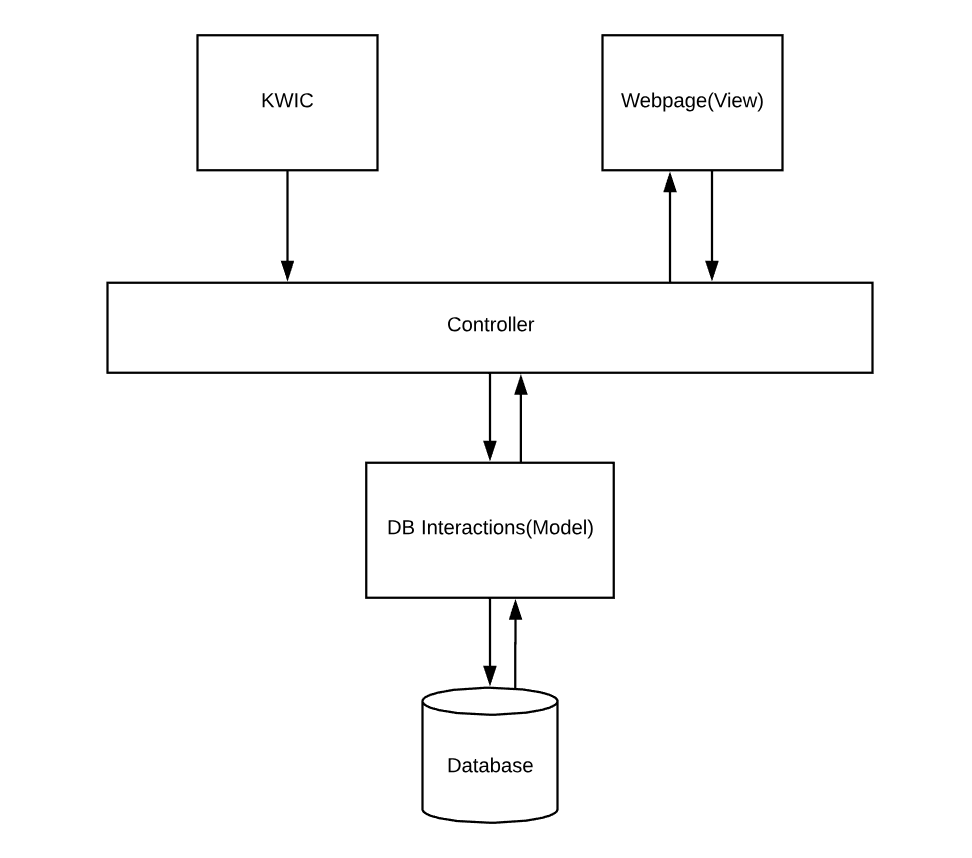


Figure 1. Architecture 1.

### Non-Functional Requirements

Portability

The system is extremely portably. It will work on any system that can run a web browser.

Enhanceability

The system is easily enhanceable. Adding functionality to any part does not affect the others. For example, adding a word filter would only require changes in the DB interactions.

Reusability

The architecture has high reusability. Each part make few assumptions about the others.

Good performance

The system takes in input from the user and the controller converts it to commands for the model. The model returns its results and the controller converts it to the view. This extra in between step requires extra data transfer and processing compared to a direct view to model communication,

User-Friendly

The system is very user friendly. The user only needs to know how to use a web browser. They do not need any knowledge of how the system works.

Adaptability

The system is adaptable. Every time the input changes, the controller is notified and retrieves information from the model to change the view.

Scalability

The system uses a database system so it is extremely scalable.

Reliability

The system is reliable, it uses a relational database which follows ACID.

Robustness

The system is robust. It takes into account misspellings and suggest words similar to the user input.

## Architecture 2

The second architecture considered (shown in figure 2) connects the KWIC system with the database directly instead of using the controller to control the inputs and outputs to the database. The searching part of the cyberminer will be the same as architecture 1.

The second architecture is not as good as the first one since it has lower enhanceability and adaptability.

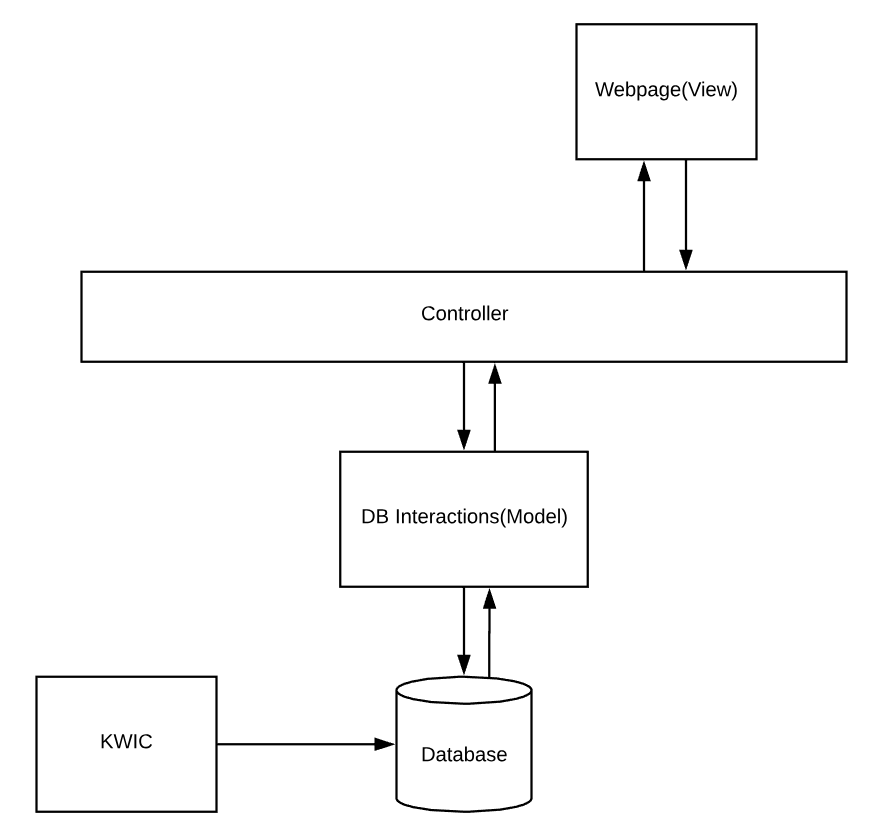


Figure 2. Architecture 2.

### Non-Functional Requirements

Portability

The system is extremely portably. It will work on any system that can run a web browser.

Enhanceability

This system is less enhanceable. When any changes to the database input and output requirements are expected both the controller and the KWIC system would need to be changed.

Reusability

The architecture has high reusability. Each part make few assumptions about the others.

Good performance

The system takes in input from the user and the controller converts it to commands for the model. The model returns its results and the controller converts it to the view. This extra in between step requires extra data transfer and processing compared to a direct view to model communication,

User-Friendly

The system is very user friendly. The user only needs to know how to use a web browser. They do not need any knowledge of how the system works.

Adaptability

This system has low adaptability since changing the input or output will require changes in the KWIC and DB interaction modules.

Scalability

The system uses a database system so it is extremely scalable.

Reliability

The system is reliable, it uses a relational database which follows ACID.

Robustness

The system is robust. It takes into account misspellings and suggest words similar to the user input.

## Architecture 3

The third architecture (shown in figure 3) is similar to the second architecture but the KWIC system interacts with the DB interactions component instead of directly with the database.

This architecture has similar problems to the second architecture. It has lower enhanceability since both the controller and the KWIC interact with the DB Interactions.

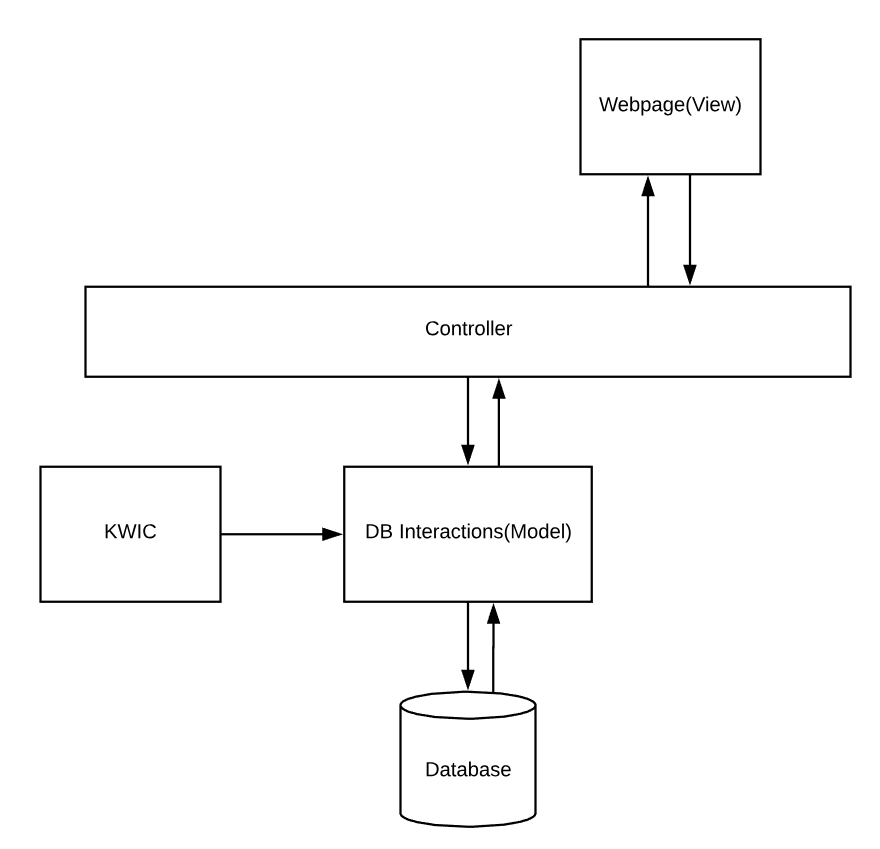


Figure 3. Architecture 3.

### Non-Functional Requirements

Portability

The system is extremely portably. It will work on any system that can run a web browser.

Enhanceability

This system is less enhanceable. When any changes to the database input and output requirements are expected both the controller and the KWIC system would need to be changed.

Reusability

The architecture has high reusability. Each part make few assumptions about the others.

Good performance

The system takes in input from the user and the controller converts it to commands for the model. The model returns its results and the controller converts it to the view. This extra in between step requires extra data transfer and processing compared to a direct view to model communication,

User-Friendly

The system is very user friendly. The user only needs to know how to use a web browser. They do not need any knowledge of how the system works.

Adaptability

This system has low adaptability since changing the input or output will require changes in the KWIC and Controller modules.

Scalability

The system uses a database system so it is extremely scalable.

Reliability

The system is reliable, it uses a relational database which follows ACID.

Robustness

The system is robust. It takes into account misspellings and suggest words similar to the user input.

## Architecture 4

The last architecture considered is shown in figure 4. It does not have a controller module and the KWIC system and Webpage interact with the database through the DB Interaction module.

This architecture is not has good because although it has better performance by eliminating a intermediate module (the controller) the job the controller had is left for the DB Interactions module which leads to high coupling.

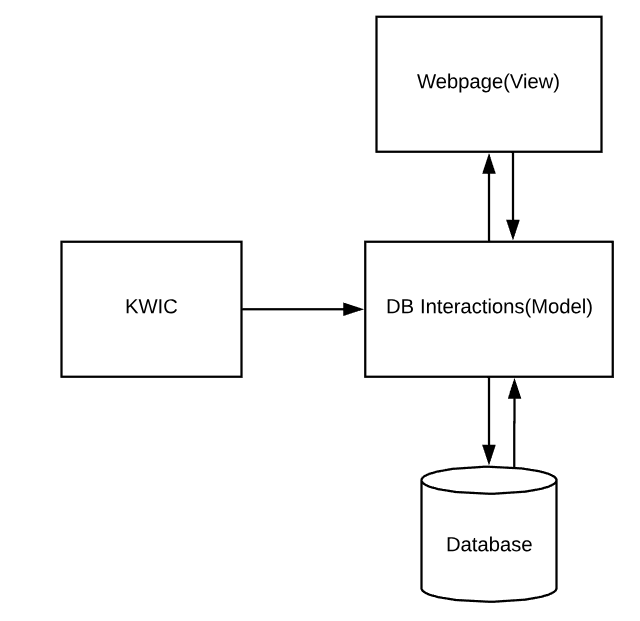


Figure 4. Architecture 4.

### Non-Functional Requirements

Portability

The system is extremely portably. It will work on any system that can run a web browser.

Enhanceability

The system is easily enhanceable. Adding functionality to any part does not affect the others. For example, adding a word filter would only require changes in the DB interactions.

Reusability

The architecture has high reusability. Each part make few assumptions about the others.

Good performance

The system takes in input from the user and the DB Interactions module will convert and query the database for the required information. It has good performance since it does not need to transfer data through many modules.

User-Friendly

The system is very user friendly. The user only needs to know how to use a web browser. They do not need any knowledge of how the system works.

Adaptability

Scalability

The system uses a database system so it is extremely scalable.

Reliability

The system is reliable, it uses a relational database which follows ACID.

Robustness

The system is robust. It takes into account misspellings and suggest words similar to the user input.

### Conclusion

The best architecture among the ones considered is architecture 1. Although is is more complicated due to a middle step (controller) and may use more processing time, it excels in all other aspects. It allows Decoupling of components allow components to be independent of each other so the code will have efficient code reusability. And parallel development. The separation of components allows developers to work on each component in parallel without affecting each other. Figure 5 shows the chosen architecture.

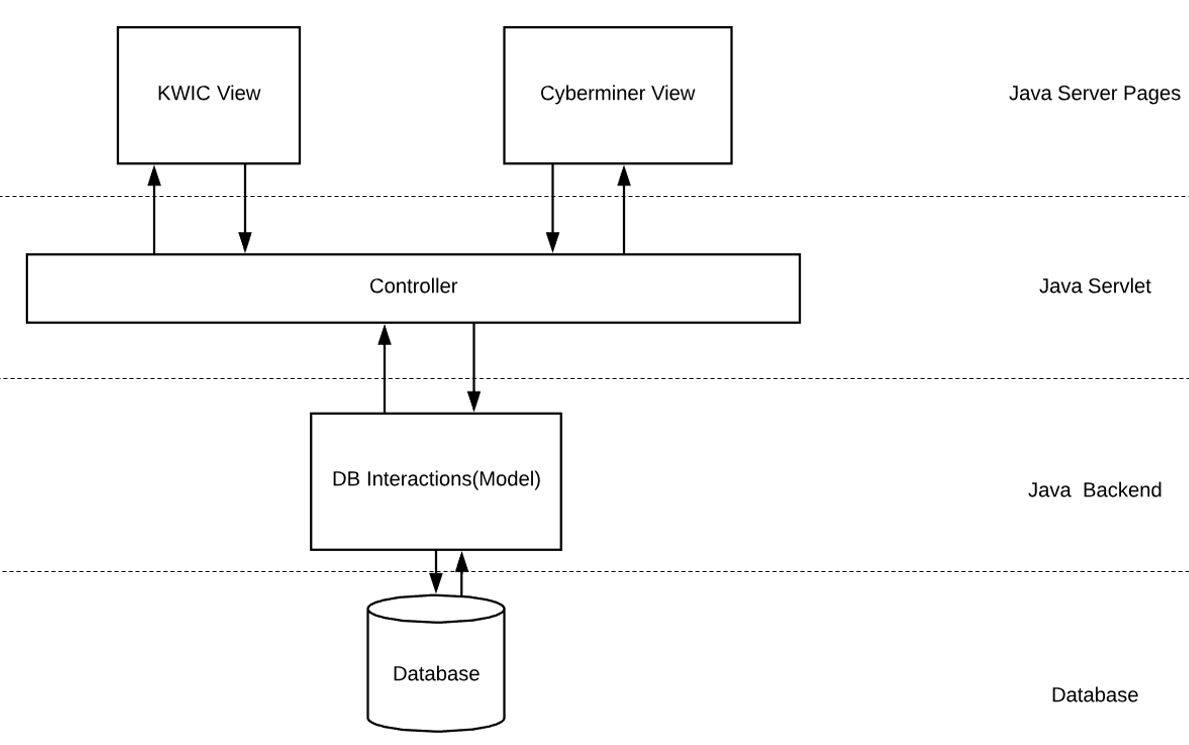


Figure 5. Chosen Architecture.

The View is divided into the KWIC view and the Cyberminer View. The KWIC view allows administrators to view, add or remove the URLs and their corresponding descriptions. The Cyberminer View allows users to do searches. The Views use Java Server Pages. The Controller gets requests from the views, send the information to the Model and receives the returned information from the Model. The Controller then updates the View. The Controller uses Java Servlet. The Model or DB Interactions uses Java and receives requests from the Controller. It will query the database and return the results to the Controller.

# Specification of program

## Program Objectives

The objective is to store the http links with a description in our database. Then, the KWIC system shall filter out the noise words and make the description searchable. Our system shall suggest these descriptions when anyone is using our search function. The suggestions shall locate the links using the suggestions and the description. Then the system shall serve these links to the person in a clickable format. The person then shall be able to use these links to access the webpage they are looking for.

There shall be the option to delete the link-description pairs from our database.

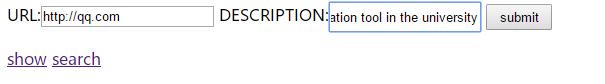
## Output Specification

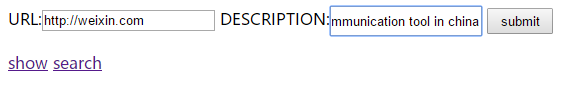
The result of the “show option shall show all the link and description pair from the URL.

The result set of the keyword search shall contain the links to the specified web pages. These links shall be clickable and the browser shall link them to the page.

## Required Input

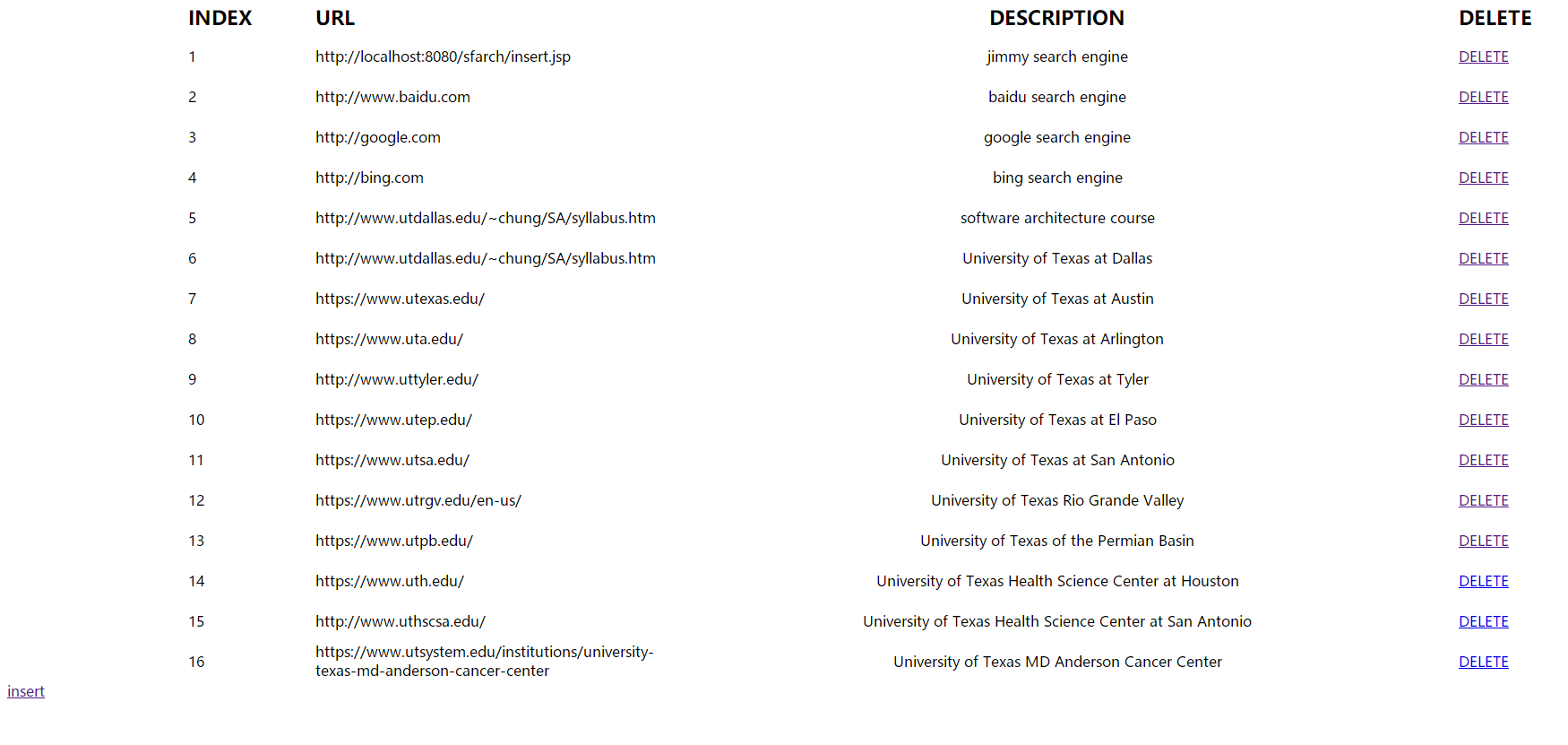
Input will contain a URL copied from Google search engine and a description of the URL. Once these two fields are filled, click “Submit”. This will make these fields into a pair and insert it into the database. Some examples are shown below:



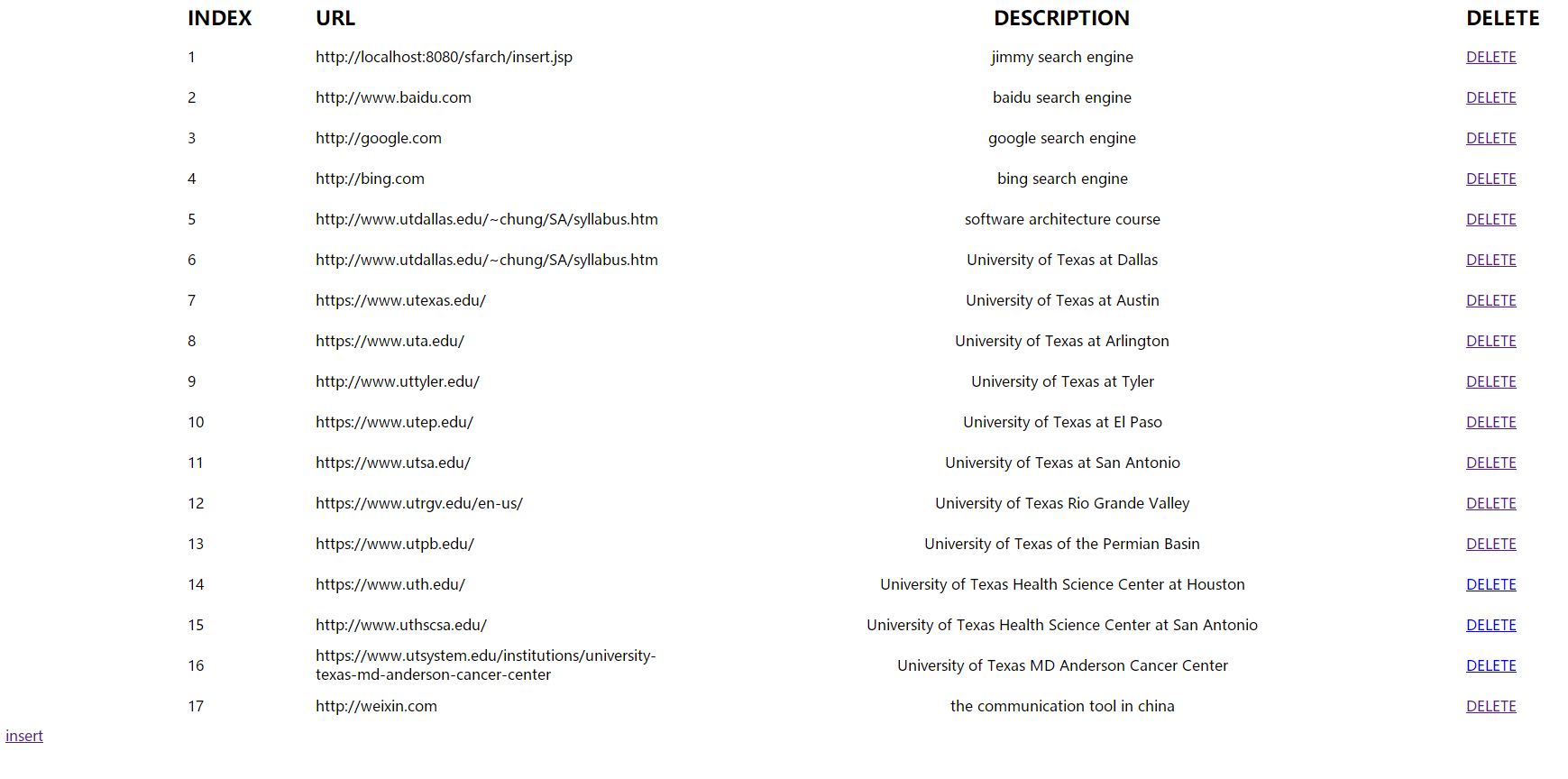


There will be two options: “show” and “search”.

Clicking on “show” will show all the pairs stored inside the database with their index. If there are too many results, the webpage will show them in pages. The result is in the image below:



After deleting the 17th data:

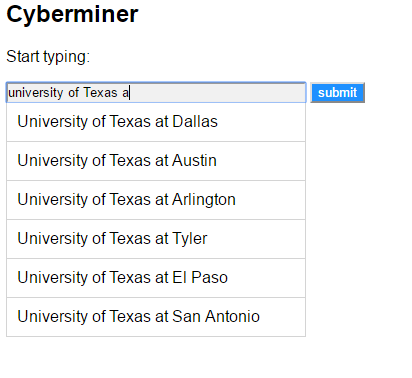
**

On this page, there shall be an option to delete the pair/tuple from our database. Clicking on this button named “DELETE” will send signal to delete it from our database. Refreshing the page by clicking F5 or Shift + F5 will show the result without the deleted pair.

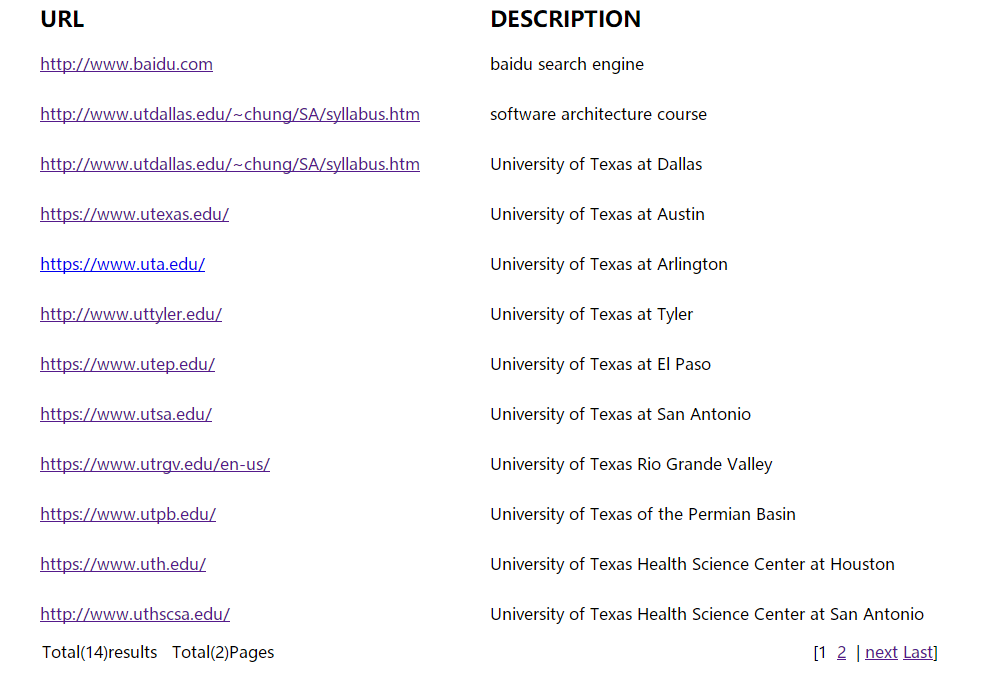
The “search” option will load a new page. This page contains a field to search the links using the descriptions stored in the database. These descriptions are the second part of the pair/tuple stored.

When a key is entered, the system will filter the keyword in the description and suggest relevant descriptions. For example: typing “se” will show suggestions which contain words beginning with “se”. These words do not have the be the first keyword. These suggestions can be chosen using mouse click. By clicking on the suggestions, the page will automatically submit the search query and display the relevant results which contain the description. This can result in a more limited result set. Example of searching with keyword:





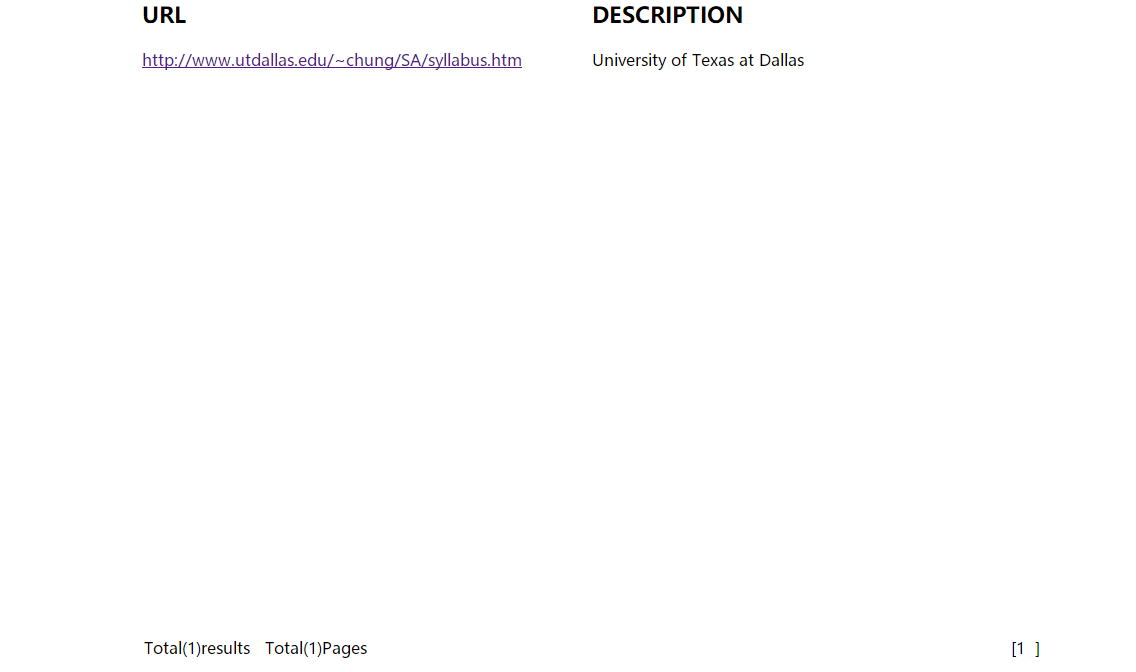
Searching using the input keywords instead of choosing the suggestions will result in a larger result set. The result page is shown here:



The system shall accept AND operator as a space. For example: searching for “un” and “dallas” shall be input as “un dallas”.



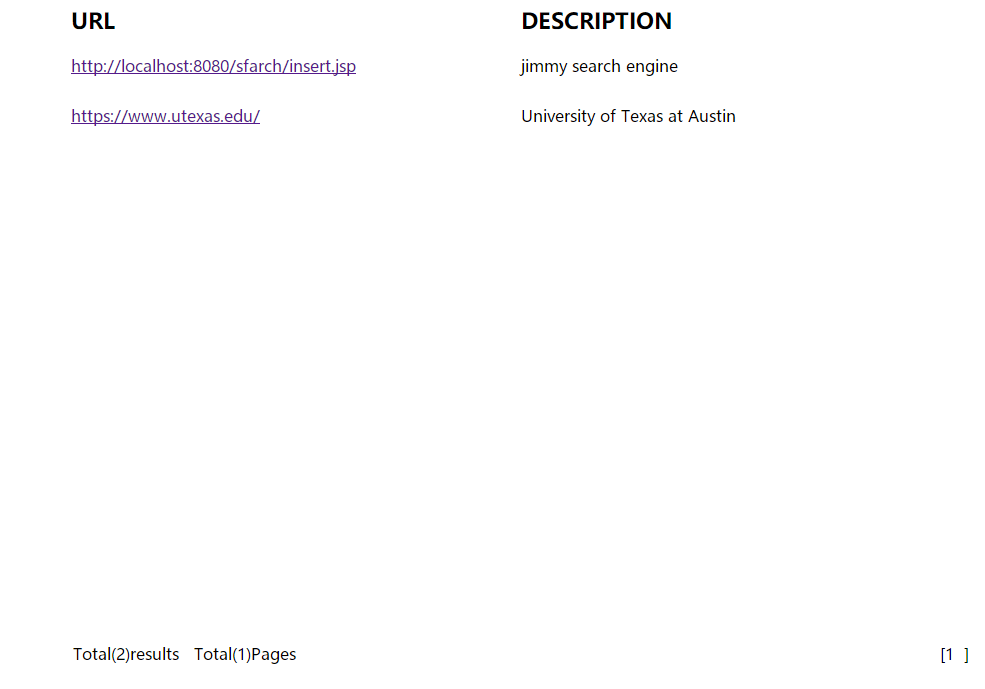
The result is usually any result that contains all keywords:



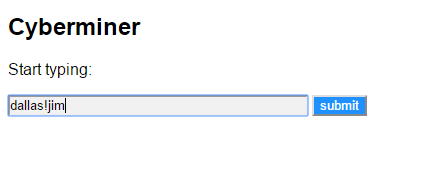
The system shall accept OR operator as “|”. For example: searching for “search” and “kwic” shall be input as “search|kwic”.

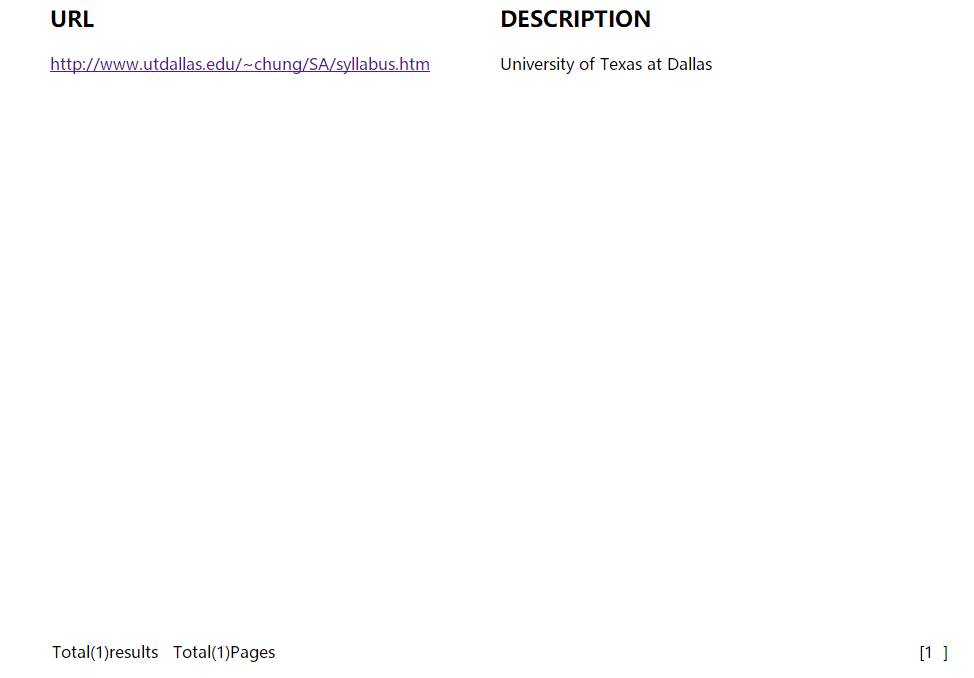


The OR operation contains more results.



The NOT operation is represented by exclamation mark (“!”). For example: searching for “not jimmy” shall be input as “!jimmy”





## Processing Requirements

The program uses java as a backend to process sentences input through the frontend website. The backend takes in a sentence, performs circular shift of the descriptions, noise elimination and alphabetization using the ADT architecture specified in the architecture section and stores the results in database. This shall be done when the pair is inserted into the database. The website can then retrieve everything from the database and outputs the results based on the requests from the browser.

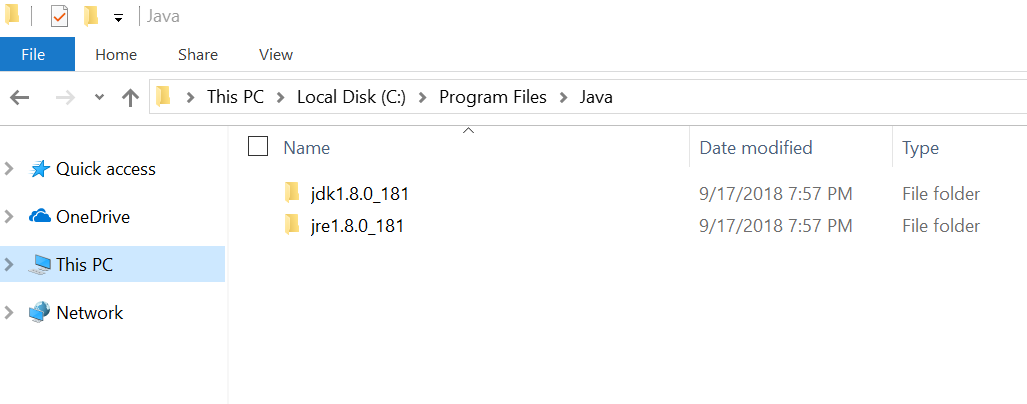
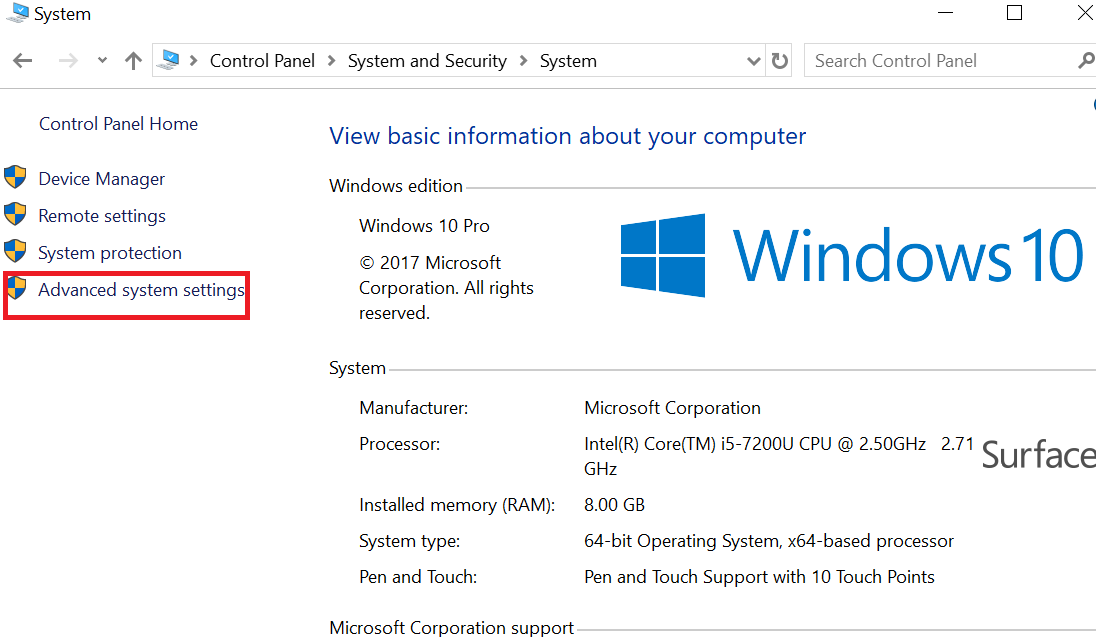
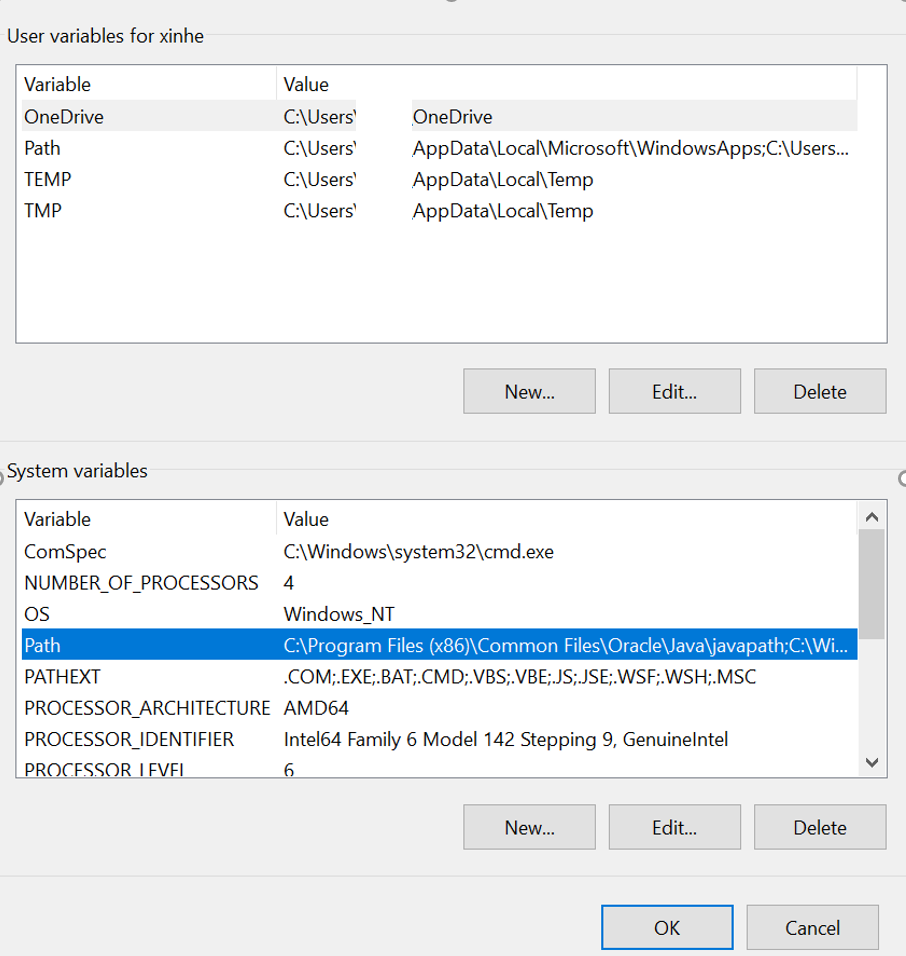
The system shall parse the search AND, OR and NOT operators into SQL query and send it to the database.

# Manual (How to install and run an example)

The source code for the CYBERMINER system can be found at <https://github.com/xinhechen/KWIC>

To run the code, JDK8, Eclipse, Apache-Tomcat, MySQL, XAMPP will be needed. The following will be a step by step guide on how to download, install and run the program.

1. JDK8

1. Go to <https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html> and download the correct version of jdk8 for your computer.
2. Run the installer which will install both the JDK and JRE
3. Go to C:\Program Files\Java to take note of your JDK installed directory. In this diagram it is "C:\Program Files\Java\jdk1.8.0\_181"
4. Go to Control Panel - System and Security - System - Advanced system settings 
5. Go to Advanced tab -Environmental Variables.
6. Under System Variables, scroll down Path and select it. Click Edit...
7. Click new and add your jdk bin directory (C:\Program Files\Java\jdk1.8.0\_181\bin for the computer shown in graph)

2. Eclipse

1. Download eclipse at <https://www.eclipse.org/downloads/>
2. Install eclipse

3.Apache Tomcat

1. Go to <https://tomcat.apache.org/download-90.cgi> and download the binary core installer
2. Run and install Apache Tomcat

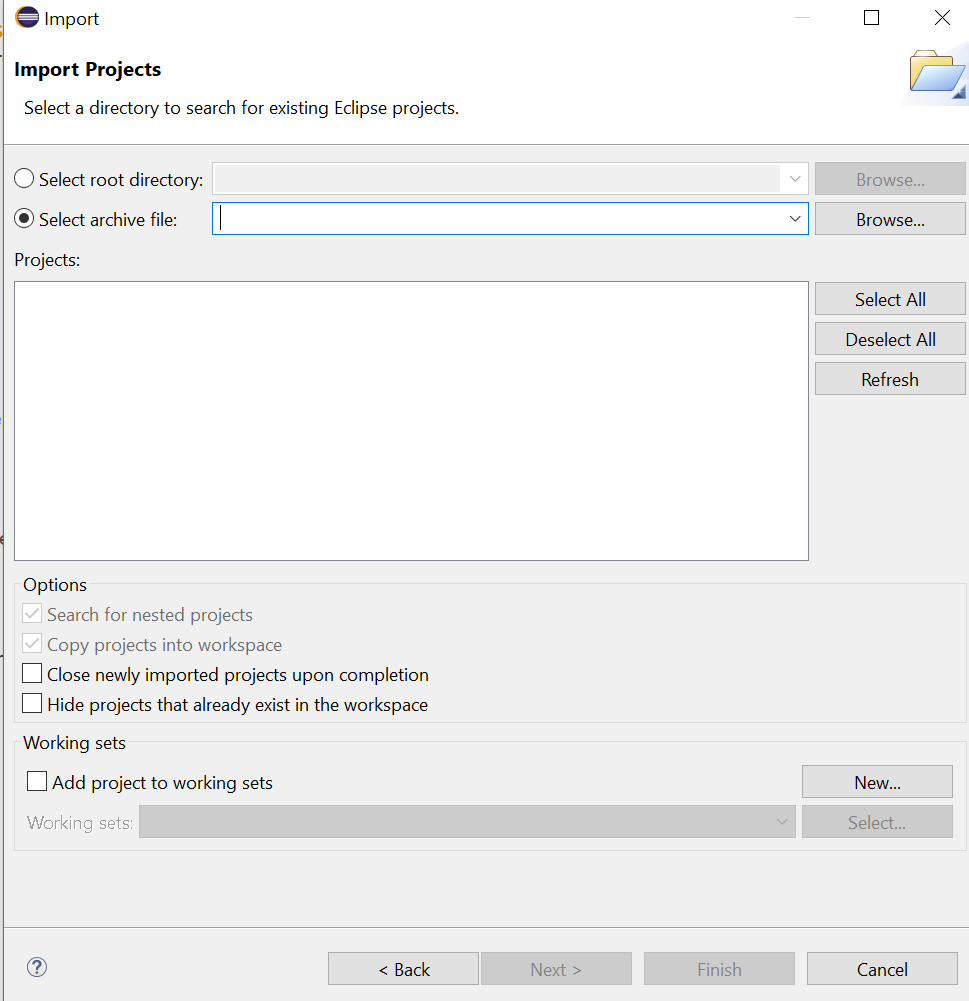
4. MySQL

1. Go to <https://dev.mysql.com/downloads/installer/> to download MySQL
2. Run the program to install MySQL
3. Make the root password “root”

5. XAMPP

1. Go to <https://www.apachefriends.org/download.html> and download the appropriate installer for your computer
2. Run the installer to install XAMPP

6. Import code to eclipse

1. Download the sfarch file from <https://github.com/xinhechen/KWIC> as zip
2. Launch Eclipse select File -> Import ->General->Existing Project into Workspace -> Next
3. Choose Select archive file, select the zip file downloaded and click finish 
4. username =root password= root url:
5. Run XAMPP to turn on MySQL
6. Run As Tomcat server at localhost

7. Go to the webpage: <https://localhost:8080/sfarch/input.jsp> to see the working program