**CyberMiner - Web Search Engine**

**Phase 2 Final Project Document**

Fredrick Horn

Parker Tate

Collin Matz

Pranit Konda

Conner Carraher

Joshua Brown

Jacob C Danel

Website URL: <https://cs4376-team-a.github.io/Project/>

**Team meetings**

|  |  |  |  |
| --- | --- | --- | --- |
| Meeting # | Meeting Date | Discuss notes | Members present |
| 1 | 06/08/2023 | Distribution of work for overall project: design and documentation: Collin Matz, Joshua Brown, Parker Tate, Jacob Danel; System implementation: Frederick Horn and Conner Carraher  System Testing: Joshua Brown, Jacob Danel  User Manual: Conner Carraher  \*\*note\*\*: Pranit Konda was not present for this meeting as he was not in the class yet. | Conner Carraher, Collin Matz, Joshua Brown, Frederick Horn, Parker Tate, Jacob Danel |
| 2 | 06/30/2023 | Discussion of how to go about work for phase 2 of the project. Same work assignments. | Conner Carraher, Collin Matz, Joshua Brown, Frederick Horn, Parker Tate, Jacob Danel, Pranit Konda |
| 3 | 7/28/2023 | Discussion of determining finalization of existing documentation and the system for final submission.  \*\*note\*\*: Pranit and Conner were not present for this meeting due to work obligations. | Collin Matz, Joshua Brown, Frederick Horn, Parker Tate, Jacob Danel |

**Revision History**:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Revision Date | Description | Author |
| 1.0 | 06/23/2023 | Initial Draft, layout, formatting. | Jacob C Danel |
| 1.1 | 06/27/2023 | Rough Draft of Document: Writing of introduction, adding in various design documents along with descriptions, etc. | Jacob C Danel |
| 1.2 | 6/28/2023 | Updated some diagrams, proofread sections, further fleshed out the implementation section and added associated diagrams as well as a textual and video demonstration of the current iteration of the CyberMiner system being used. | Jacob C Danel, Collin Mitz, Joshua Brown |
| 2.0 | 7/29/23 | Proofreading and updating some graphs to better reflect the current version of the CyberMiner system. Added new diagrams. | Jacob Danel |
| 2.1 | 7/30/23 | Added some more sections and updated MVC. | Jacob Danel |
| 2.2 | 7/31/23 | Finalizing document, added test cases with results, etc. | Jacob Danel |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

1. **Introduction**
   1. **Purpose………………………………………………………………………….6**
   2. **Scope of the System…………………………………………………………..6**
   3. **Objectives……………………………………………………………………….7**
   4. **References………………………………………………………………………7**
2. **Domain Model**
   1. **Description……………………………………………………………………...8**
   2. **Model…………………………………………………………………………….9**
   3. **Class Models……………………………………………………………………9**
3. **Requirements Specifications**
   1. **Description……………..……………………………………………………..12**
   2. **Functional Requirements…………………………………………………...12**
   3. **Non-Functional Requirements……………………………………………..13**
   4. **Traceability Matrix……………………………………………………………14**
   5. **Model…………………………………………………………………………...15**
      1. **Use Cases**
      2. **Sequence Diagrams**
4. **(Architectural) Design Specifications**
   1. **Description…………………………………………………………………….18**
   2. **Model…………………………………………………………………………...18**
      1. **Class Diagram**
      2. **Class Models**
      3. **Sequence Diagrams**
   3. **Architecture Design………………………………………………………….21**
      1. **Activity Diagrams**
      2. **State Transition Diagrams**
   4. **Component Design…………………………………………………………..23**
5. **Implementation**
   1. **Description…………………………………………………………………….25**
   2. **Model – Implementation…………………………………………………….25**
   3. **Model – Test Cases…………………………………………………………..26**
   4. **Demonstration………………………………………………………………..28**
      1. **Creating a New Index**
      2. **Modifying an Existing Index**
      3. **Deleting an Index**
      4. **Logging in as an Admin**
      5. **Searching the Database for Websites**
         1. **OR conditional**
         2. **AND conditional**
         3. **NOT conditional**
      6. **Connecting to a Website**
      7. **Video Demonstration**
      8. **Test Cases Table with Results**
6. **Glossary……………………………………………………………………………40**
7. **Introduction**
   1. **Purpose**

The purpose of this search engine, CyberMiner, is to allow users of the system to be able to search for websites using keywords they submit as input as quickly and efficiently as is possible by the CyberMiner system. This is done primarily by breaking up the input obtained from the user and kept as keywords to be compared to other keywords associated with websites within our website list database. Once relevant matches are found using these keywords CyberMiner would then return to the user a list of related websites that the user may choose from. The primary goal for implementing the system this way is to generate and return said list of websites to the user as soon as they hit the enter button.

Response time is a particularly important factor in the development of CyberMiner. Typically search engine/web browser users have come to expect near instantaneous results when utilizing such technologies, so maintaining this expectation with CyberMiner is an important aspect to achieve. In order to effectively do so, the system must be able to promptly obtain input, extract keywords from input, compare the keywords obtained from users to the websites’ own keywords in the indices table, return relevant websites, generate an easily navigable webpage with a list of the relevant websites, then connect the user to the website whose list entry they interact with. This is done for all indices in the table and all relevant websites are stored in a list as this process is happening. Once the relevant website list has been generated it is then displayed on a navigable web page generated by CyberMiner for the user. The user can then connect to whichever website they choose to click on.

* 1. **Scope of the System**

The scope of the CyberMiner system, in relation to a user’s experience, is as stated ahead. A typical user will open the CyberMiner application on their personal computer which should then display a page similar to other search engines/web browsers where there is a text field where the user can input whatever search queries they may want to make. Once input is given by the user to CyberMiner, CyberMiner would then use that input to generate a page of related websites that the user may then be able to connect to through the CyberMiner application. Once the connection is made the page is updated to be that of the chosen website’s layout and other related information. Should the connection fail however, CyberMiner will instead update the page to display an error message that gives the user a brief description of the error (i.e., connection failure due to no internet connection, a database connection error, etc.)

* 1. **Objectives**

Current objectives:

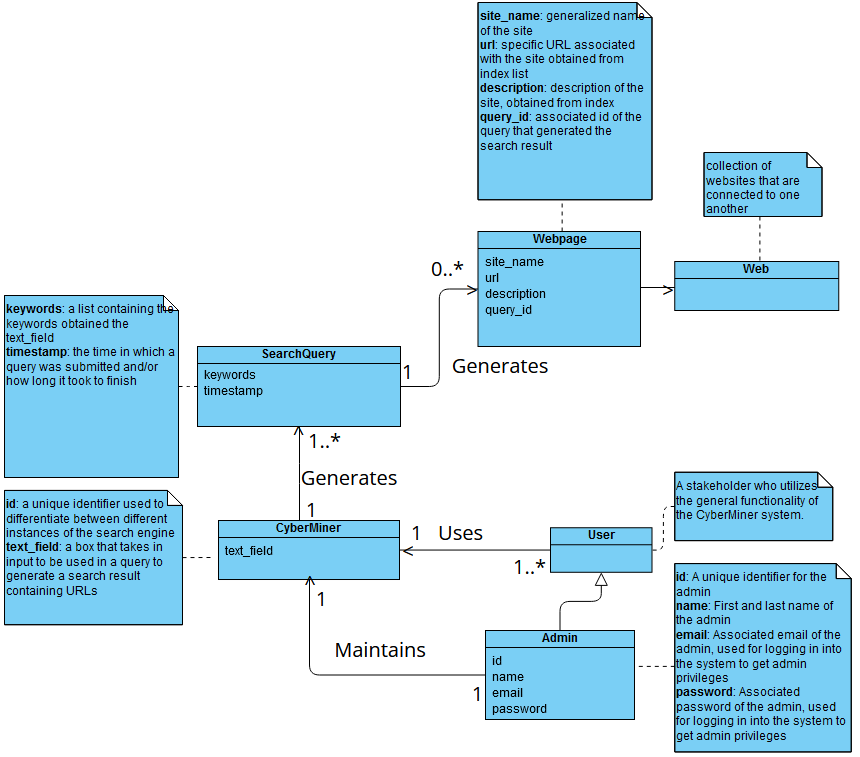
1. Keep the system as simple as possible for the User, Users do not need to be able to interact with the various components of CyberMiner, just its search functionality.
2. Keep index table search speeds as fast as possible through whatever means necessary.
3. Build upon existing version of CyberMiner with added functionality, currently CyberMiner allows for case sensitive searches and hyperlink enforcement.  
   1. **References**

* Sample system design document found on Prof. Chung’s website:
* https://personal.utdallas.edu/~chung/CS6354/CS6354\_U07\_source/Team\_1/CAD\_Design\_\_4.doc
* Another Sample system design document found on Prof. Chung’s website: <https://personal.utdallas.edu/~chung/CS6354/CS6354_U07_source/Team_2/deliverable_1_Req_Specification.doc>
* Prof. Chung’s personal webpage where slides and such are found: <https://personal.utdallas.edu/~chung/OOD/syllabus.htm>
* Phase 1 assignment document: https://personal.utdallas.edu/~chung/OOD/Project1.pdf

1. **Domain Model**
   1. **Description**

The domain model is meant to be a more general overview of the system for users to be able to easily understand and to be used as a basis for further design documents and diagrams. In its current iteration the domain model shows the general sequence of events that occur when a user, or admin, interacts with the CyberMiner system. For general users, the functionality needed in order for them to use CyberMiner effectively is primarily giving input and receiving web results that they can then connect to with a click of the mouse. Additional functionality could include such things as allowing users to create bookmarks for websites they frequent often, cataloging and displaying their browsing history, and other basic functionality one would find in other search engines/browsers. However, those added functions are not entirely necessary to be able to use CyberMiner effectively, more so they are just quality of life additions.

For Administrators of the CyberMiner system the functionality they require is more robust compared to general users of the system. On top of the basic functionality needed in order to use the system, admins would need additional functionality that would help them in properly maintaining the system and troubleshooting any problems that may arise during use. For instance, such functionality would be having access to access logs to see where or why a connection was not able to be made with a particular website for a user. Another may be having access to the index table, so that they may make changes to it as needed (creating, updating, fixing entries in the table, etc.).

* 1. **Model** ****
  2. **Class Models**

1. **User (stakeholder)**

Someone that interacts with and utilizes the CyberMiner system in order to search for websites. Primary functionality needed would be what was mentioned before, searching for desired websites based on provided input. To stay on track keeping the system “as simple as possible” for the users their functionality will consist of only being able to search for websites using the system. Adding additional functionality for them does not facilitate the goals of “simplicity” and “speed” that we are aiming for with this system. Typically, additional functionality bloats and slows systems down.

1. **Admin (stakeholder)**

A user that has more permissions/functionality with the CyberMiner system than a typical user would have. Examples of additional permissions/functionality would be anything that would help facilitate their ability to maintain the system. For instance, having access to and the ability to change or manipulate the index table, as well as individual indices in the index table. Displayed in the model an Admin has three attributes: “id”, “name”, and “email” and “password”. Their “id” attribute would be used in the authentication process when an admin would attempt to make changes to the index table. The “name” attribute is mostly there for display purposes. The “email” and “password” attributes are used by the admin to log into the system as an admin and utilize those additional functionalities.

1. **CyberMiner**

This is the primary system that both users and admins will be interacting with. In this version of the domain model, it has one attribute field called “text\_field”. Text\_field represents the text box in which input will be received, the attribute itself will store the input taken in by the system that is then later broken down into keywords.

1. **SearchQuery**

A search query is generated after the CyberMiner system has been given some input by a user or admin. It has two attributes called “keywords” and “timestamp”. The keywords attribute is a list that is created from the input given to CyberMiner that was stored in the text\_field attribute, CyberMiner breaks up the input based on delimiters in the input and stores the values in between those delimiters into the keywords list. The timestamp attribute is a value which stores the time when the query is generated. This attribute would primarily be used for troubleshooting reasons by an admin.

1. **WebPage**

A webpage in the system is a visual that a user or admin sees which displays the list of websites obtained from the index table that are relevant to the input given by the user or admin. It has a number of attributes, those of which are: “site\_name”, “URL”, “description”, “query\_id”. The first three would be visible by any user of the system while the query\_id attribute is primarily for admins for troubleshooting purposes. “Site\_name'' is as the name suggests, the generalized name of the website, “URL” is the web address link that is used to make connections to said website, “description” is a basic description of what it is the website is about or what the website offers.

1. **Web**

As stated in the diagram above it is the collection of websites that are connected to one another.

1. **Requirements Specifications**
   1. **Description**

The Requirements section, as well as the various models and diagrams utilized within it, provides a visual or conceptual representation of how the system would be used by any of the stakeholders (those being users and admins). In the provided use cases, there is an instance of both an Admin and a User utilizing the system. For an admin, a typical use case is managing the index table. Whether that be by creating a new index to be inserted into the table or by modifying an existing index within the table. Modification of an index would include changing information within an index to better reflect the index as it is currently known or to outright delete an index from the index table. For a user, a typical use case is where they search for a website using the CyberMiner system.

* 1. **Functional Requirements**

The functional requirements primarily deal with what the system’s inputs and outputs typically would be and how they are each handled.

Inputs:

* What
  + information received for a user that is to be used by the system to perform a search on the databases and return websites that are relevant to what the user submitted.
* How is it obtained
  + Input comes from a user that submits information in an interactable text field.
* How is it used
  + the input is processed through the system to be used to compare to indices keywords within the index table
* What is the process of making it useable by the system
  + The system takes the input as a string, that string is then broken down into individual strings for each word, those words are stored in a keyword list, then each of those input keywords is compared against the keywords

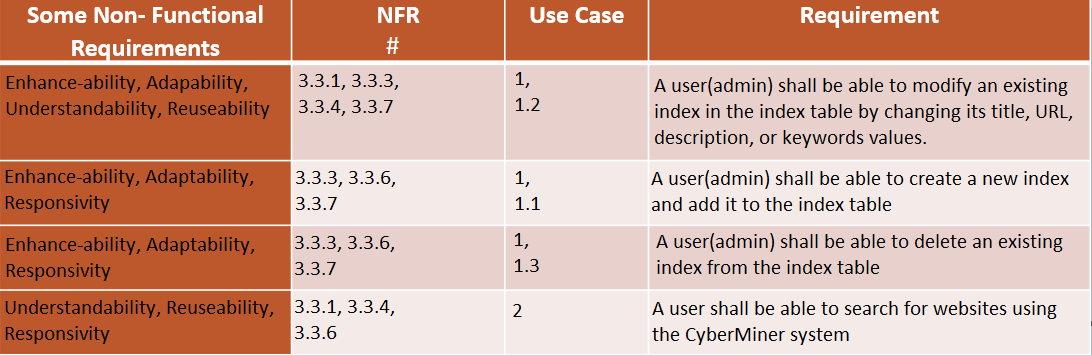
Outputs:

* What
  + A list of websites to be presented to the user after a search query is completed. More specifically a list of URLs and their associated website titles (potentially descriptions as well) that is obtained after the input keywords are compared to all index keywords within the indices table.
* How is it obtained
  + It is obtained after the input keywords are compared to all index keywords within the indices table and a list of websites is generated.
* How is it used
  + It is used to display the list of websites to the user in the GUI and allow them to connect to a website of their choosing

Manipulating the index table

* What
  + The ability to create, delete, or modify indices within the index table
* Where does it come from
  + Inputs given by an admin through the GUI screen for index table modification. Those inputs being the add and delete buttons primarily. Modification of existing indices is done by changing the specific field values and saving those change by hitting “enter” on the keyboard afterwards.
* Where does it go
  + All added, deleted, or modified indices are kept within the index table, which is presented to admins through the modification GUI.
* How is it done
  + Through the use of the aforementioned buttons and interactables. To get to this window an admin must hit the “modify” button at the top right corner of the home screen and login in with their credentials.
  1. **Non-Functional Requirements**

1. Understandability: The level at which the system is able to easily be used by a user. The goal being to have a UI that is user friendly and easily understandable at a glance.
2. Portability: The system shall be platform independent, making it easily deployable on multiple machines.
3. Enhance-ability: The CyberMiner system shall be enhance-able and support changes over the course of time.
4. Reusability: Code components shall be reusable to allow for streamlining of implementation.
5. Performance: The CyberMiner system shall provide reliable performance from the user’s perspective by returning search results and quickly and cleanly as possible.
6. Responsivity: Similarly, to performance, the system shall be quick to respond to a user’s input.
7. Adaptability: The CyberMiner system shall be adaptable in the sense that certain changes can be made to the system by an admin as needed.  
   1. **Traceability Matrix**

****

* 1. **Model**   
     A picture containing text, diagram, plan, technical drawing

     Description automatically generated
     1. **Uses Cases and Associated Use Case Texts** A picture containing diagram, text, circle, sketch

        Description automatically generatedA picture containing text, receipt, screenshot, parallel

        Description automatically generated
* For admins, their input is whatever new information they aim to provide to the index table through the functions createIndex and modifyIndex and their associated output is thus the additional indices or modifications made to existing indices.
* For users, their input is a string that is received by the system and their output is a list of websites that are related to the input they gave.
* Admins are a specialized type of user and thus also have the same functionality that is allowed to users on top of their added functionality
* For traceability purposes the specific use case instances within the above use cases will be listed as:
  + 1: Manage Index
    - 1.1 Create Index
    - 1.2 Modify Index
    - 1.3 Delete Index
  + 2: Perform Search Query
    1. **Sequence Diagrams** A picture containing text, diagram, screenshot, parallel

       Description automatically generated

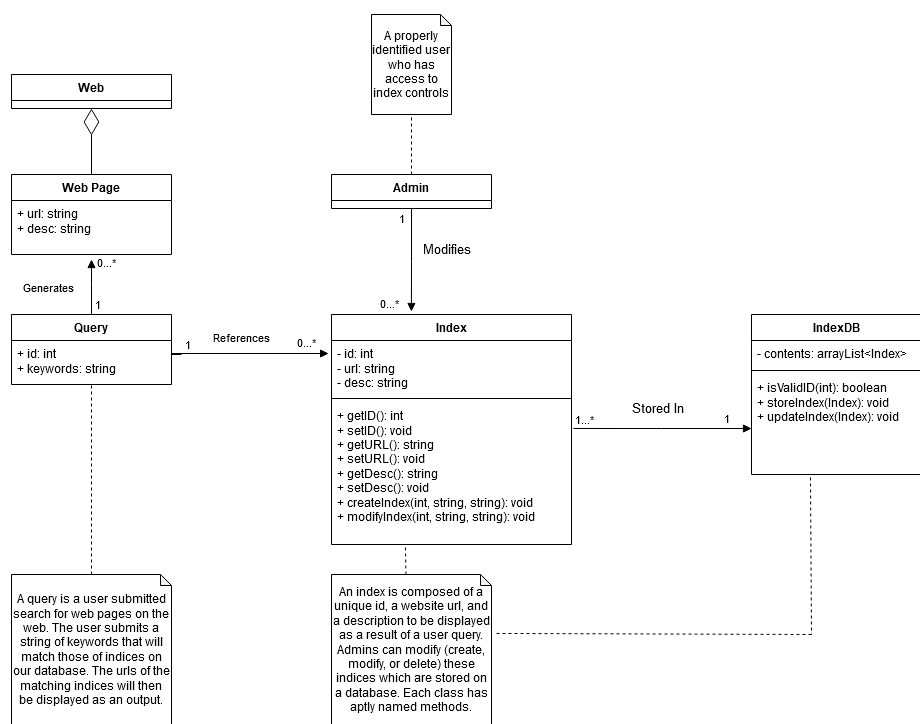
A screenshot of a computer

Description automatically generated with low confidence

* The sequence of events in the sequence diagrams are easy enough to follow in the diagrams themselves.

1. **(Architectural) Design Specifications**
   1. **Description**

The design specifications give a more robust representation of how the system is going to be implemented. In the class diagram it shows a more detailed representation of the class diagram of the system seen in the Domain Model. How CyberMiner’s queries interact with the indices table is shown, where the query will reference a specific index (or indices) from the index table (referred to as the IndexDB in the diagram). The diagram further shows the relationship between admins and the index table as well as some of the functionality they are able to utilize in order to effectively monitor and manipulate the index table as needed (through the functions createIndex and modifyIndex). Furthermore, the sequence diagrams from the Requirements section are also shown with more detail. Specifically, two different instances of where an admin is manipulating the index table through createIndex and modifyIndex.

* 1. **Models**
     1. **Class Diagram**
* The flow of events is, a query is generated by a user, which references an index or indices from the index table (aka IndexDB)
* Using the indices obtained from the IndexDB, the query then generates a web page that displays a list of all sites that are relevant to the users search query
* The user is then able to connect to whatever website they choose from the list generated by clicking on them
* The flow of events for an admin is the same as a user when submitting a search query, for making modifications to the index table it is slightly different, those will be discussed more in the sequence diagrams below
  + 1. **Class models**

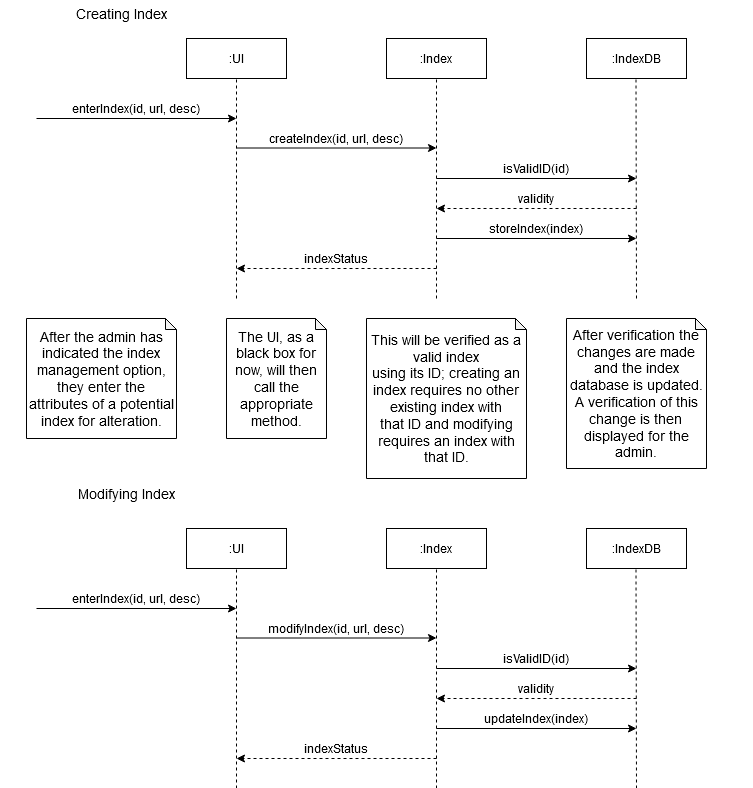
(Those that were not already described in the Domain Model section)

1. **Index**

An entry within the indices table. Contains all the information about an individual website that is stored within the table. Has the attributes “id”, “URL”, and “desc”. The “id” attribute refers to a unique identifier value that is used by an admin for troubleshooting reasons, “URL” is just the website’s uniform resource locator link used to connect to a website, and “desc” is a short description of the website which details what kind of information it has or what kind of services it offers. The index class has a number of functions that facilitate the admin being able to properly maintain the system. Included are some basic getter and setter functions as well as the createIndex and modifyIndex functions which allow the admin to make changes to the IndexDB

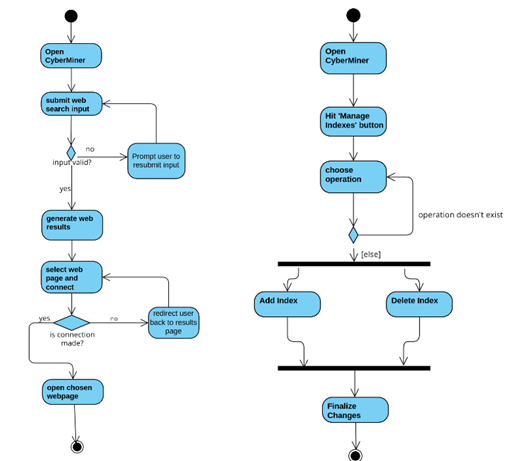
1. **IndexDB**

The indexDB is responsible for containing the list of all indices. Its attribute “contents” is the list that was previously mentioned. The IndexDB class also has functions that facilitate the admin being able to properly maintain the system. Those being “isValid()” which checks the validity of an individual index within the table, the “storeIndex()” function appends a new entry into the table, then the “updateIndex()” function allows an admin to modify an existing index within the table.

* + 1. **Sequence Diagram**
* Two different instances of an admin making modifications to the indices table is shown above
* The first sequence diagram shows an instance where an admin creates a new index to be stored within indexDB. Once the admin uses the createIndex function the IndexDB checks the validity of the new index using isValid, in this case the created index is valid, so the IndexDB returns a message denoting that and then the new index is thus stored within the IndexDB. After which the Index class sends an IndexStatus message to the admin’s UI that confirms the successful creation.
* The second sequence diagram shows an instance where an admin modifies an existing index within the IndexDB. Once the modification has been submitted by the admin the IndexDB checks the validity of this change, in this case the modified index is still valid and is thus stored within the indexDB. After which the Index class sends an IndexStatus message to the admin’s UI that confirms the successful modification.

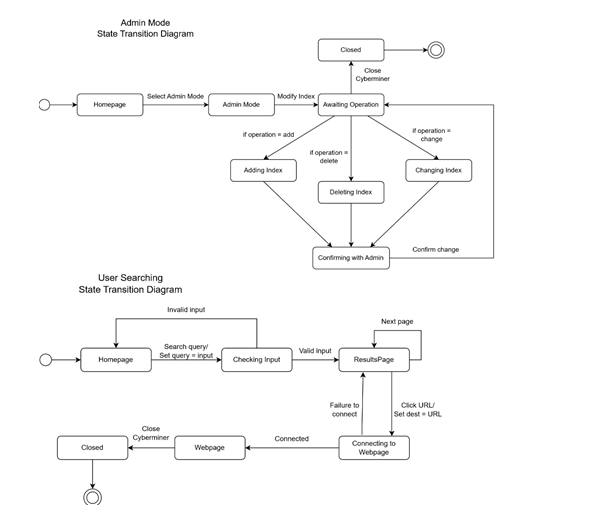
**4.3 Architecture Design**

**4.3.1 Activity Diagrams**

****

Above are some example activity diagrams of the expected series of events that will happen for a typical interaction between a user and the system (the image on the left) or one particular instance of an administrator making edits to the indices table within the system, i.e. clicking the manage indexes button to then go to the indices table where there are further options to add, delete, or modify an index or indices. These were examples taken from the early Requirements specification document (some use case examples) remade using activity diagrams.

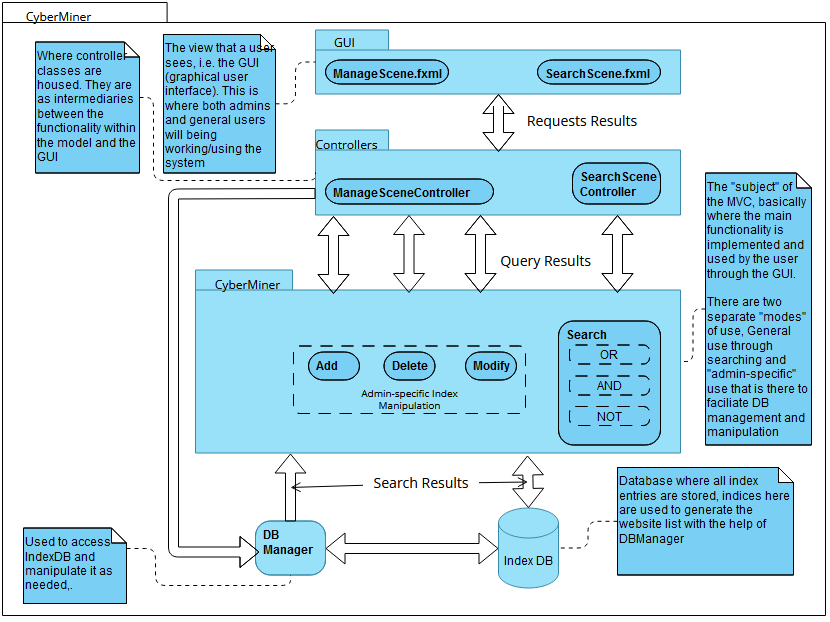
**4.3.2 State Transition Diagrams**

****

Above are some example state transition diagrams showing some interactions between the system and a user or administrator. It is the same series of events that occurred in our use case diagrams within the requirements specification, just with more detail on what specifically happens from action to action.

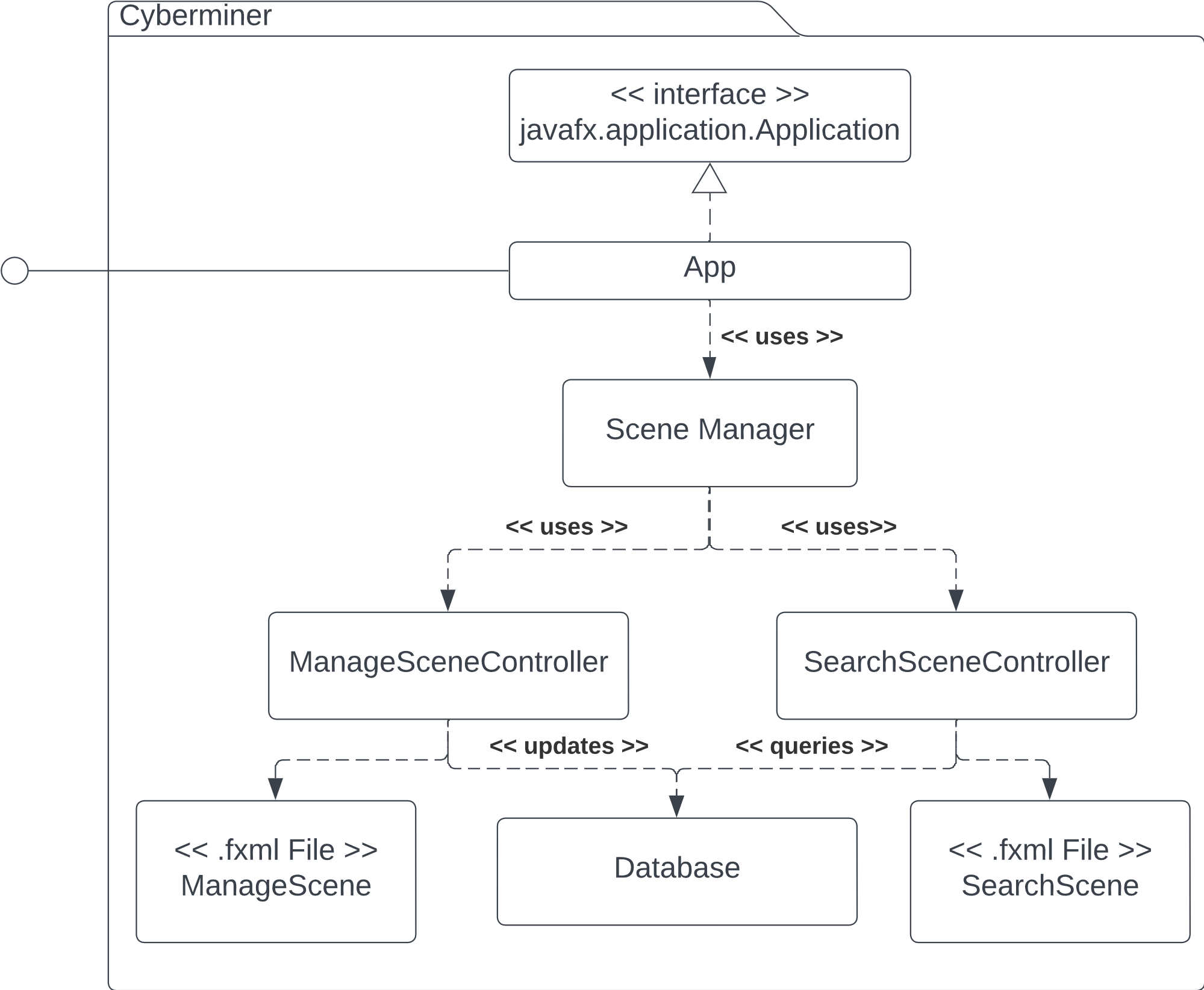
**4.4 Component Design**

**4.4.1 Model-View-Controller Representation**

****

Above shows a simplified representation of how the different components within the system interact with one another. This is primarily represented using a MVC model which utilizes the observer design pattern. The general sequence is that a user would interact with the view model (i.e., the GUI), which would in turn interact with the control model in order to perform the necessary functions the user is trying to do. Like a search query being made and the system returning a list of websites relevant to their query. This query pulls the data from the index DB (database) through the DB manager. Once this is obtained the query results are run back up to the view component and displayed to the user through the GUI.

**4.4.2 Facade Design Pattern Representation**

****

* Above shows another representation of how the different components within the system interact with each other. This time using the façade design pattern. The same sequence of events occurs where a user would interact with the App (CyberMiner in this case), which in turn utilizes the Scene Manager to indirectly interact with the two different controllers to perform the necessary functionalities. Where the ManageSceneController is related to the functionality needed by Admins to modify the index table. While the SearchSceneController is related to the basic functionality of the system, that being performing search queries. Both of these controllers interact directly with the Database.

1. **Implementation**
   1. **Description**

The CyberMiner system is to be implemented using Java and Javafx. The indices will be stored within a database using MySQL. The code for our system has been primarily written by Fredrick Horn and Conner Carraher.

* 1. **Model – Implementation**A screenshot of a computer screen

     Description automatically generated with medium confidence
  2. **Model – Test cases**A picture containing text, diagram, font, parallel

     Description automatically generated
     1. **Test Suite LoginA screenshot of a computer program

        Description automatically generated with low confidence-** Instances of how a login test case would happen, just input email and password to be granted access, if fields do not match what is on file, then deny access. The “name” string is the associated display name to the admin’s account. Demonstrated in the second video in section 5.4.
     2. **Test Suite Generate QueryA picture containing text, receipt, line, font

        Description automatically generated  
        -** Instances of how generation of a query may be handled in the system, essentially the system would read the keywords string list and parse through the index database for each keyword and populate another list containing websites whose known/stored keywords match with the user’s input. Demonstrated in the second video in section 5.4.
     3. **Test Suite Create/Delete IndexA picture containing text, font, line, number

        Description automatically generated-** Demonstrated in the first video in section 5.4
     4. **Test Suite Modify IndexA picture containing text, receipt, screenshot, line

        Description automatically generated  
        -** Demonstrated in the first video in section 5.4
  3. **Demonstration**

Below will be a demonstration of the current capabilities of the CyberMiner system. Currently the System has functionality to support manipulation of the index table. There will be a text-based representation with images as well as a video representation of these test cases in case that is easier to digest.

* + 1. **Creating a New Index**

A screenshot of a computer

Description automatically generated with medium confidence

* Navigate to the Index Table, this can be done by hitting the button in the top left corner called “Manage Indexes”

A screenshot of a computer

Description automatically generated with medium confidence

* The fields are filled in with the necessary information, “test” for title, “thisisatest.com” for URL, “this is a test” for description, and “test” for keywords. Once these fields are filled it an admin can hit the “Add” button to append the entry as an index to the index table at the bottom.
* Navigating to the bottom of the list shows that the entry has been made into A screenshot of a computer

  Description automatically generated with medium confidencean index and appended to the end of the index table.
  + 1. **Modifying an Existing Index**
* Modification of indices is simple within the CyberMiner system
* A screenshot of a computer

  Description automatically generated
* By double clicking on a field (not including ID, Created Date, and delete) for any of the indices an admin is able to alter the information in that field. Once the change has been made the admin can hit enter to save the change for that field. A screenshot of a computer

  Description automatically generated with medium confidence
* After hitting enter the change is made, in the case above the description field for the index we created before went from “this is a test” to “this is a modification”
  + 1. **Deleting an Index**

**A screenshot of a computer

Description automatically generated with medium confidence**

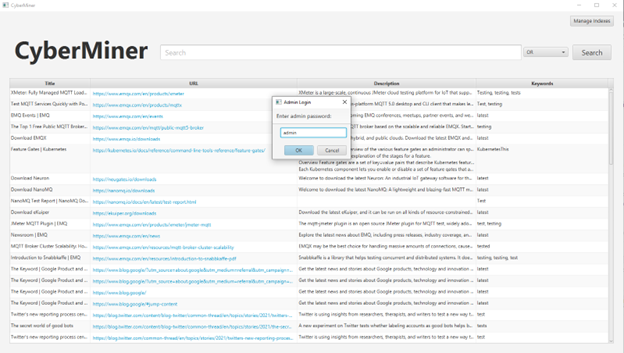
* Deletion of an index occurs when an admin hits a delete button in the row for the index they intend to delete. (Continued on next page).

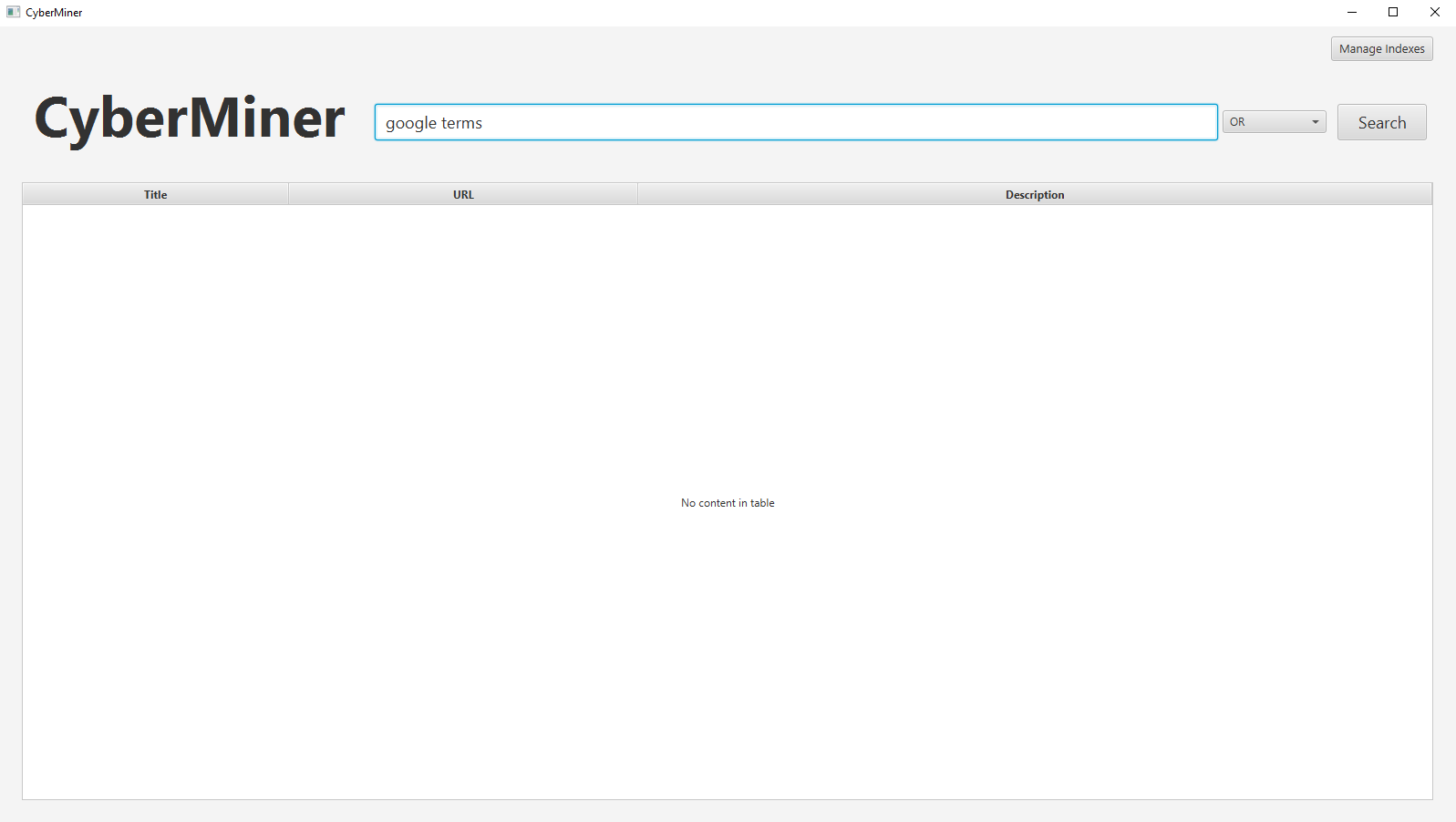
A screenshot of a computer

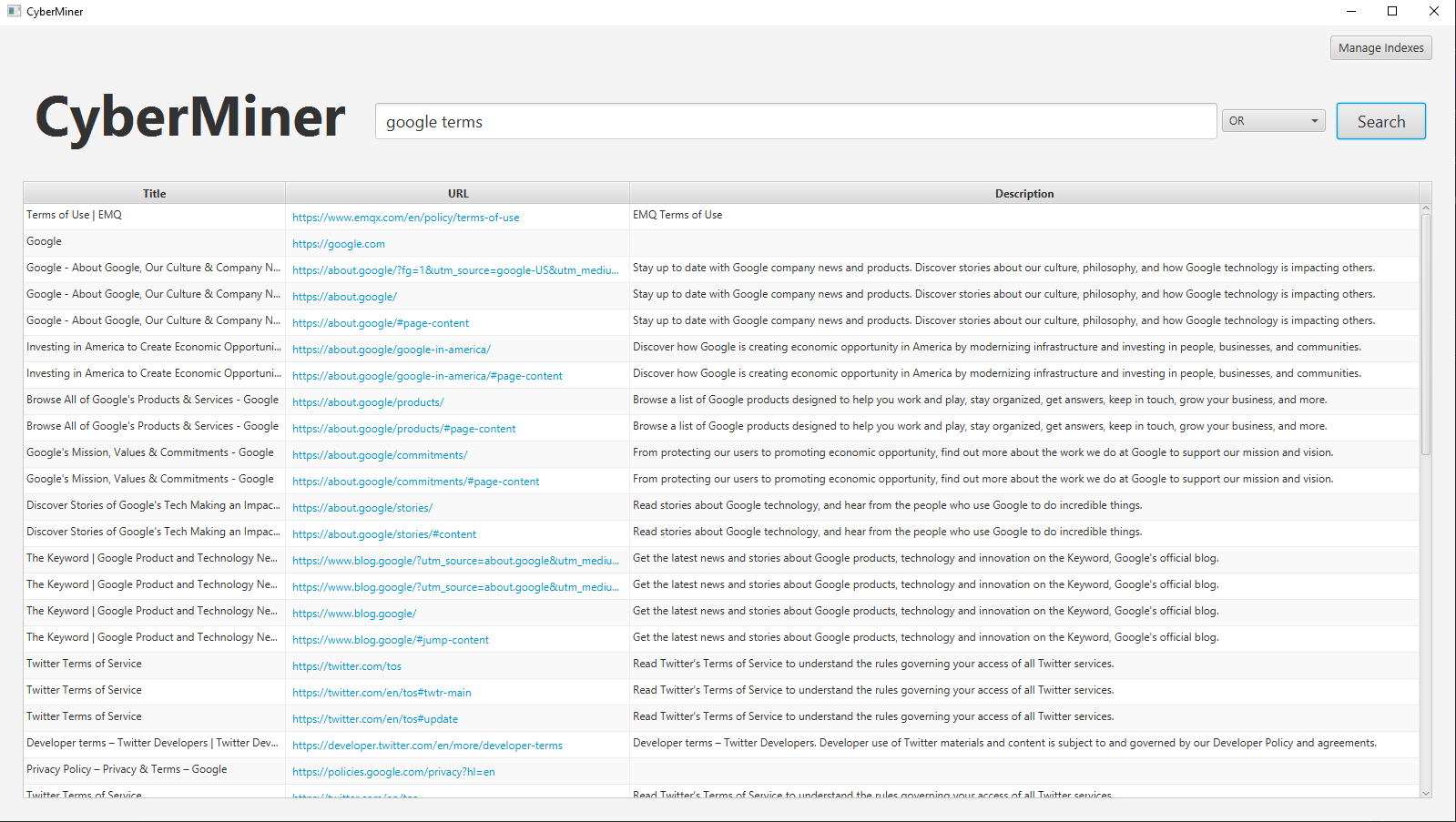
Description automatically generated with medium confidence

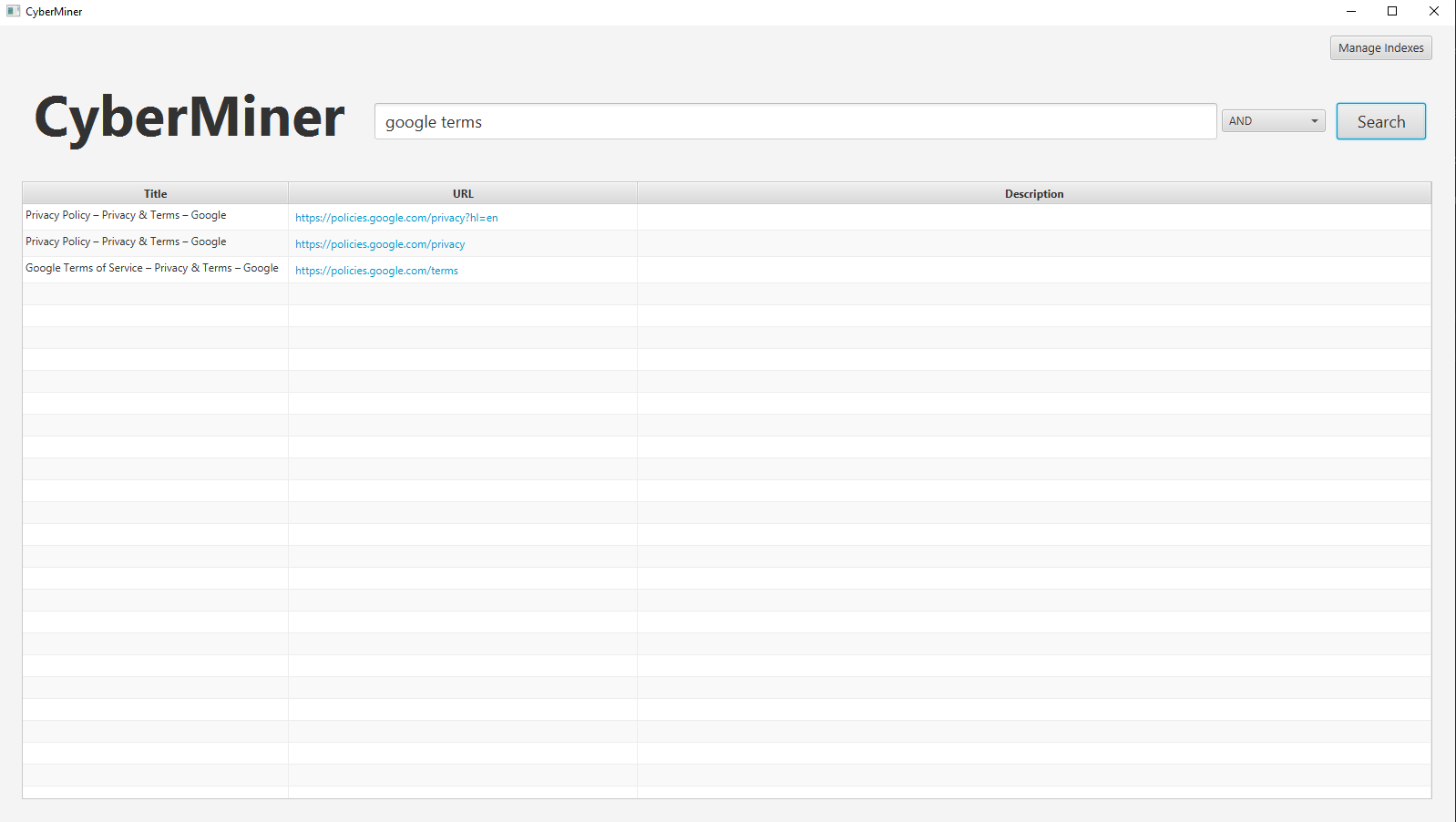
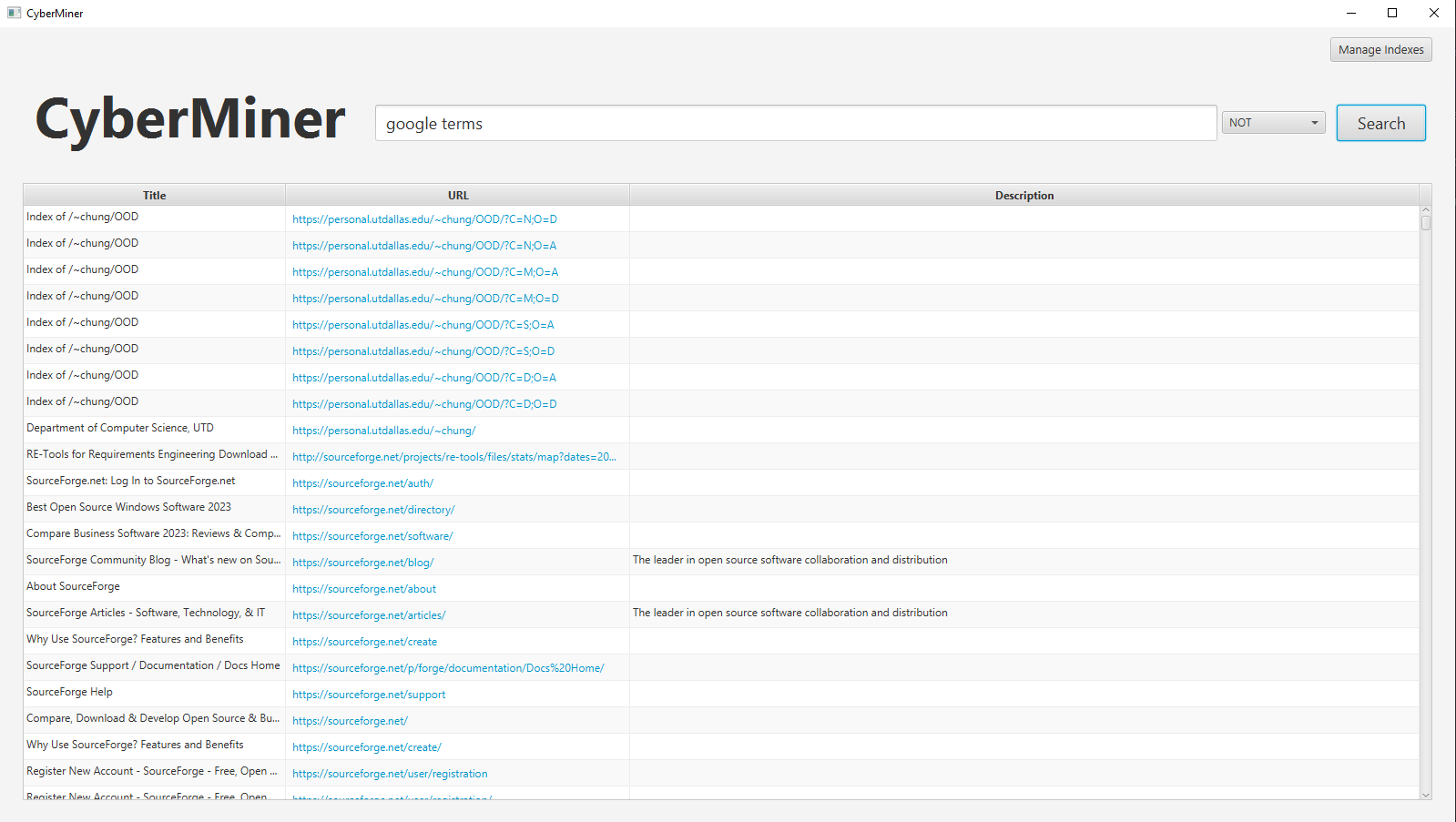
- The system after having hit the delete button for the index that was created before, the table is updated automatically and it is shown that it is not there anymore as the last entry in the index table is now 3884, where it was 3891 before.

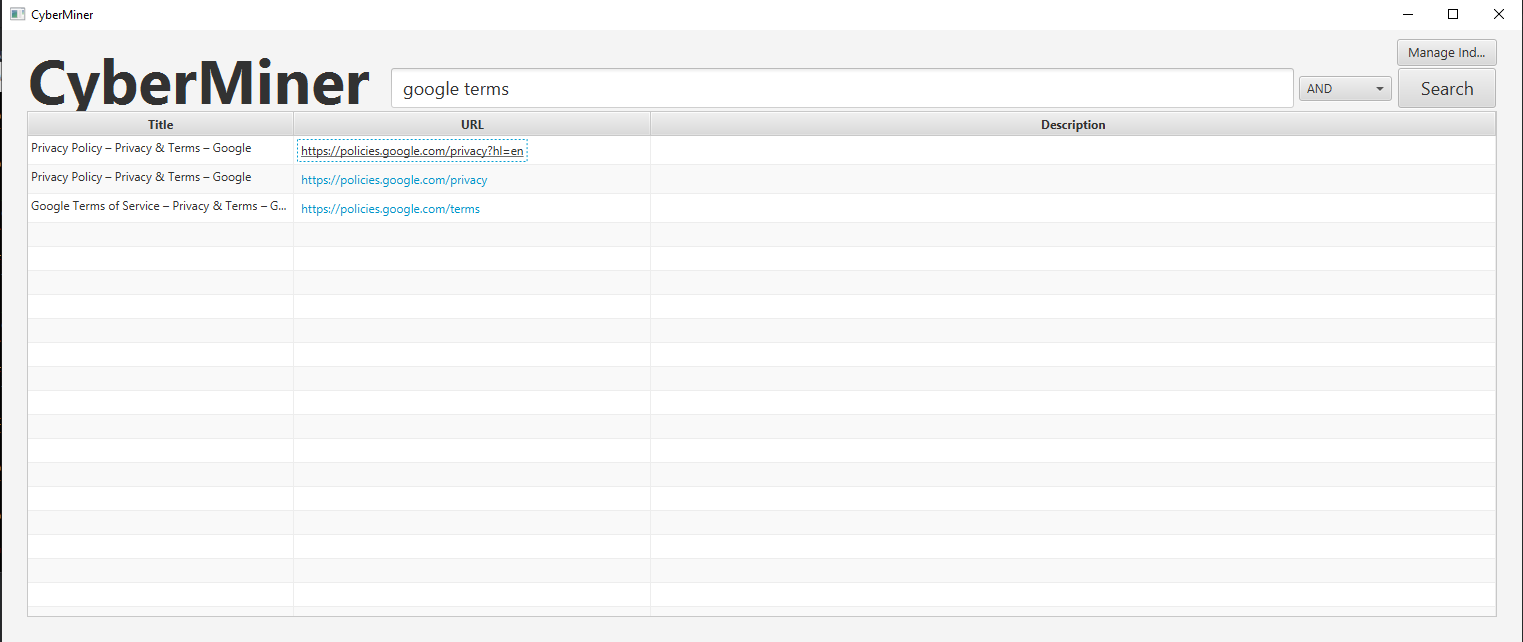
* + 1. **Logging in as an Admin**

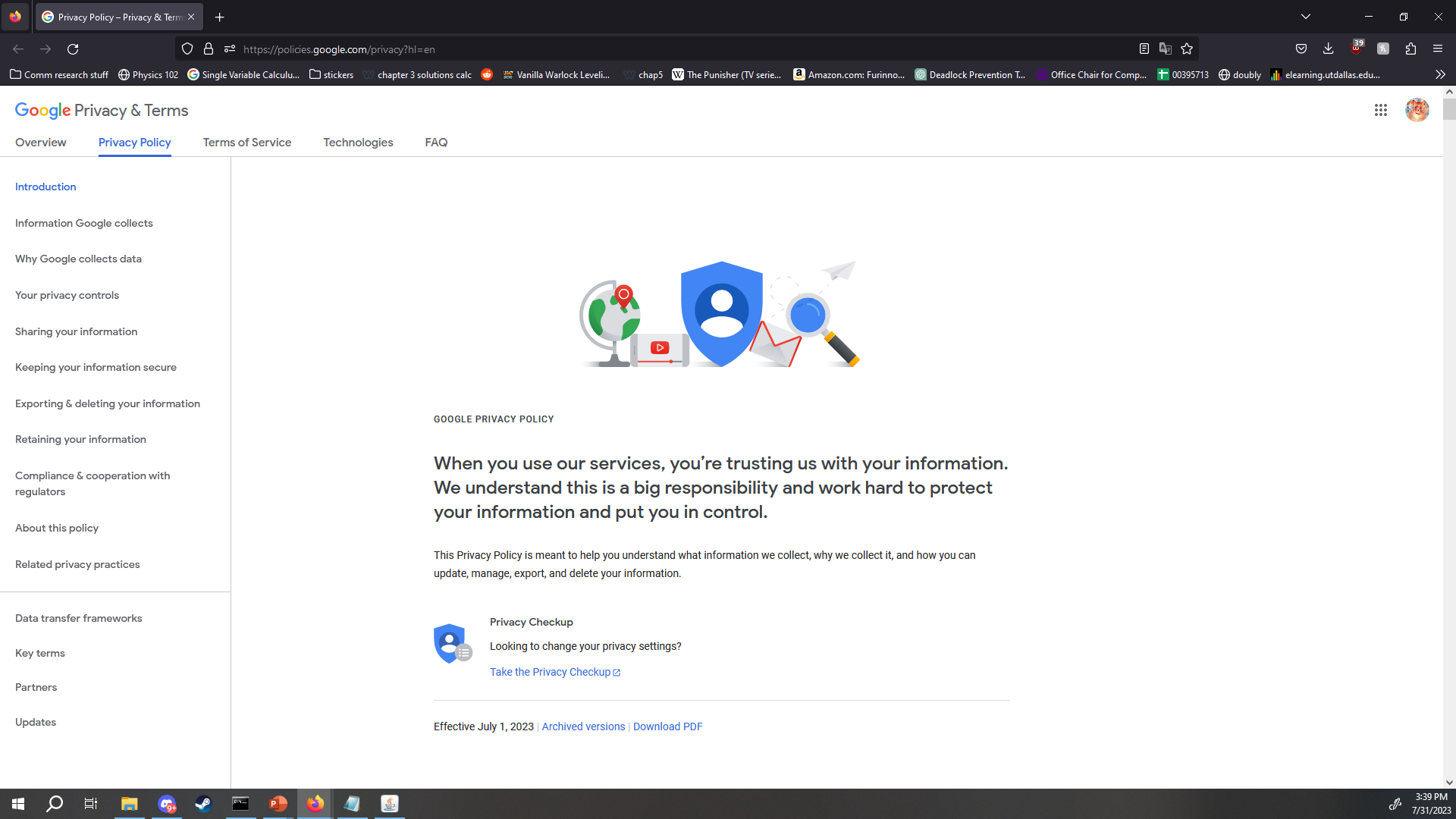
****

* Logging in is required when hitting the Manage Index button in the top right. In this instance we are logging in with the generic Admin account, whose credentials are just a password currently (the password being admin).  
  + 1. **Searching the Database for Websites**
* **OR conditional**  
  



* Above is an instance where we see the CyberMiner system before querying a search then immediately following that is a search with the OR conditional selected.
* **AND conditional**  
  
  + Above is an instance of a search query being done with the AND conditional, doing the same search terms as before. Can see that the results are much less than what was seen with the OR conditional.
* **NOT conditional**  
  
  + Above is an instance of a search query being done with the NOT conditional which returns all indices in the index table minus anything containing either of the search terms.
    1. **Connecting to a website**

****

****

* Current functionality has clicking the URL to connect to the website by opening up the selected website in your default web browser. Above this can be seen by the user clicking the first URL within the presented search results then that website being opened in the default browser.
  + 1. **Video Demonstration**

Link to video: <https://youtu.be/hQtH4G20Tcg> (this was demo for phase 1, does not have logging in/searching)

Link to video 2: https://youtu.be/-izgOs\_Ku-o (this is to be watched after the first video, covers logging in/searching)

* + 1. **Test Cases Table with Results (results in videos above)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case | Description | Test Step | Expected Result | Actual Result |
| Create Index | Makes a new index that is appended to the end of the index table. | Fill in interactable fields near the top of the program with values {“test”, “test.com”, “this is a test”, “test, testing” then hit the “Add” button. | A new index is created and appended to the end of the index table within the CyberMiner system. | A new index is created and appended to the end of the index table within the CyberMiner system. |
| Delete Index | Pressing the “delete” button next to an index entry within the index table removes that specific entry and updates the table to reflect this. | Press the “delete” button on a test index to see if said index is properly removed and table updated. | The index table is removed from the index table and the table is updated to reflect the new number of indices present within the index table. | Index is indeed removed on pressing delete and the table is properly updated to reflect this change. |
| Modify Index | Double-clicking any of the indices’ field values within the index table allows an admin to change the value of that field after making the change and hitting “enter”. | Double click on any one of the four fields for any given index and change the contents. | The field value is changed after hitting enter. | Able to properly change any of the four values for any index within the index table by double-clicking the field, typing in any changes, then hitting enter. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case | Description | Test Step | Expected Result | Actual Result |
| Login as Admin | Pressing the “Manage Indexes” buttons will prompt a user to sign in as an admin to gain access to the index table and associated functionality. | Hit the “Manage Indexes” button and provide a password. | Login and transition to index table upon successful login or notify user of incorrect login credentials with error message upon unsuccessful login. | Both cases work in testing, successful login does transition users to index table and notifies users of unsuccessful login with incorrect passwords. |
| Search OR | Submitting search keywords will return results that contain any of the keywords (some or all). | Submitted the keywords “Google terms”. | Will return results that contain at least one of the keywords provided. | Results are indeed ones that contain both of the provided keywords, which in this instance ended up being three results. |
| Search AND | Submitting search keywords will return results that contain all of the keywords (but not just those keywords, at minimum all keywords submitted). | Submitted the keywords “Google terms”. | Will return results that contain all of the keywords provided. | Results are indeed ones that contain both of the provided keywords, which in this instance ended up being three results. |
| Search NOT | Submitting search keywords will return results that do not contain the keywords provided. | Submitted the keywords “Google terms”. | Will return results that contain none of the keywords provided. | Results are indeed ones that contain neither of the provided keywords, in this case it was all indices in the index table minus any that have the keywords “Google” or “terms” associated with them. |
| Connect to Website | System should be able to connect the user to chosen website from the URL link shown after a search query has finished. | Click on the URL of a resultant index after making a search. | A connection will be made to the chosen URL and will be opened up in the default web browser. | A connection was successfully made to the chosen URL and the web page was opened within the user’s default web browser. |

1. **Glossary**

Search engine: A tool on the internet that helps one find information. A user can type in what they are looking for and it will show a list of websites and other sources that are related to the user’s search.

User: A person or entity that uses a service or tool in order to perform some task or tasks. In the case of CyberMiner a user is someone who uses the CyberMiner tool in order to search the web for specific websites.

Admin: An admin is a user with additional benefits and permissions that allow them to be able to perform maintenance and update the CyberMiner system.

Query: It is a question or request for information that a user asks a search engine. This question or request is then used by the search engine to look for relevant results that match the question or request.

Delimiter: A character or sequence of characters used to separate various parts of a text or data.

MVC: Model-View-Control design pattern, which is itself another representation of the Observer design pattern. Used to show the relationships between a controller entity and various observer entities that the controller manipulates.

GUI: Graphical User Interface, the visual aspect of the system that users primarily interact with.