ENG 3050 – Software Engineering Design I

Winter 2015

Artificial Intelligence Literature Searching

Assignment 3 – Software Design

Group #3

Contributing Members:

John Simko, Brandon Stanley, Shahood Mirza

Date Submitted: 3/2/2015

Approval

This document has been read and approved by the following team members responsible for its implementation:

Print Name	Signature	Comments
John Simko		

Print Name	Signature	Comments
Brandon Stanley		

Print Name	Signature	Comments
Shahood Mirza		

Revision History

Date	Author	Comments
Feb 15	Brandon Stanley	Created design document framework, added server class diagram.
Feb 16	Brandon Stanley	Added updated client class diagram, added login sequence diagram
Feb 27	John Simko	Updated CRC cards
Feb 28	Brandon Stanley	Added to the GUI Design section
Mar 1	John Simko	Wrote System Design algorithms for Server
Mar 2	Brandon Stanley	Added new GUI mock ups and modified client diagram, copied over CRC cards, added the introduction.
Mar 2	John Simko	Created and added Interconnection Diagram
Mar 2	Shahood Mirza	Added pseudo code and descriptions for client side portions Added sequence diagram for new user creations

Table of Contents

1. Introduction

1.1 Design Overview

This is the design document for the SAIL product. This document is split into three major headings: the Introduction, the System Architecture, and the System Design.

The system architecture is an overview of the system as a whole. It is described in detail using CRC cards. How everything connects together is shown with a class interconnection diagram. The overall architecture of the two programs are shown using UML Class diagrams. How it is envisioned to look is shown with several GUI mockups. Finally some important communications are mapped using sequence diagrams.

The GUI design section is split into three parts, the server design, the client design, and the GUI design. In each section, the owner of each respective system will cover some of the more important functions in depth. For each such function an explanation is given as to what is happening in the function, and some pseudocode is shown to give an outline of how it will be implemented.

The design and implementation of this product is split into three sections, with the server as John's responsibility, the client as Shahood's responsibility, and the GUI as Brandon's responsibility. Each section will be designed and implemented by their respective owners until completion, at which point they will aid whoever needs the most assistance to obtain completion in a timely manner.

1.2 Changes to Specification Documentation

Date	Author	Comments
Feb 15	Brandon Stanley	Updated Intermediate COCOMO, made mention of threading to Login Use Case, fixed confusing class names, and mis-typed multiplicity, added the word “program” to all references to Client and Server that appeared in use case diagram to clear up confusion, made mention of input of IP and port number to connect to the server.
Feb 16	Brandon Stanley	Updated GUI in Client Class Diagram, added admin-user based client/server communication codes, added Login Sequence Diagram
Feb 27	John Simko	Updated Introduction and CRC Cards to reflect changes to class diagram. Reorganized sections to be more understandable. Corrected typos.
Mar 1	John Simko	added Server side function response code for indicating successful admin login and for log out
Mar 2	Brandon Stanley	Re-did GUI to more accurately reflect product, minor changes to client diagram., added to CRC cards

2. System Architecture

2.1 CRC Cards

2.1.1 Server Side CRC Cards

CLASS **Server_System**
RESPONSIBILITY 1. Listen for new connections from listen socket 2. Create **Server_Side_Connection_To_Client** in its own thread. 3. Import resources table 4. Import user logins table 5. Search resources vector for search matches 6. Add or Remove **Resource** 7. Modify existing **Resource** 8. Send login information to **Login** 9. Add and remove valid logins 10. Mutex lock resources/logins when being managed
COLLABORATION 1. **Server_Side_Connection_To_Client** 2. **Resource** 3. **Login** 4. **Client_System**

CLASS **Server_Side_Connection_To_Client**
RESPONSIBILITY 1. Hold connection to **Client_System** running in its own thread. 2. Store vector of **Resource** search results 3. Send search/sort/filter results over socket to **Client_System** 4. Parse requests from **Client_System** 5. Send search requests to **Server_System** 6. Create vector subset on filtered keyword(s) 7. Send login information to **Server_System** 8. Send add/remove resource requests to **Server_System** 9. Send add/remove login requests to **Server_System**
COLLABORATION 1. **Server_System** 2. **Client_System** 3. **Resource**

CLASS **Login**
RESPONSIBILITY 1. Verify login credentials are valid 2. Return login request results 3. Identify client as user or admin
COLLABORATION 1. **Server_System**

CLASS **Resource**
RESPONSIBILITY 1. Store a resource 2. Return list of current resource attributes 3. Return resource's specified attribute 4. Return ID of selected resource
COLLABORATION 1. **Server_System** 2. **Server_Side_Connection_To_Client** 3. **GUI Controller**

2.1.2 Client Side CRC Cards

CLASS **Client_System**
RESPONSIBILITY 1. Connect to the **Server_System** 2. Send login information over server socket to **Server_System** 3. Send search information over server socket 4. Send filter information over server socket 5. Send sorting information over server socket 6. Send add/remove resource information over server socket 7. Send modify resource information over server socket 8. Send add/remove attribute information over server socket 9. Send add/remove login information over server socket 10. Send export database request over server socket 11. Send request for a list of usernames over server socket
COLLABORATION 1. **Server_System** 2. **GUI_Controller**

CLASS **GUI_Controller**
RESPONSIBILITY 1. Parse **GUI window** inputs 2. Send input data with relevant function code to **Client_System** 3. Display **Login Screen** on startup 4. After successful login, display **Main Screen** 5. If isAdmin true, display button to access **Admin Screen**
COLLABORATION 1. **Client_System** 2. **GUI_Window** **3. Resource**

CLASS **Main_Window_Frame**
RESPONSIBILITY 1. Spawn **Login_Pane**, **Tutorial_Pane**, and **Search_Pane** 2. If user is admin, also spawn **Admin_User_Pane**, **Admin_Attribute_Pane**, and **Admin_Resource_Pane** 3. Send user input to GUI Controller
COLLABORATION 1. **GUI_Controller** 2. **Login_Pane** 3. **Search_Pane** 4. **Tutorial_Pane** 5. **Admin_User_Pane** 6. **Admin_Attribute_Pane** 7. **Admin_Resource_Pane**

CLASS **Tutorial_Pane**
RESPONSIBILITY 1. Display simple text instructions for how to use the client to search, sort, filter, get other pages, etc.
COLLABORATION 1. **Main_Window_Frame**

CLASS **Search_Pane**
RESPONSIBILITY 1. Show user input box for search string. 2. Show selectable options for what field to search by 3. Show page of search results 4. Show buttons for next page and previous page 5. Show list of valid keywords to filter results by 6. Indicate which field search results are being sorted by
COLLABORATION 1. **Main_Window_Frame**

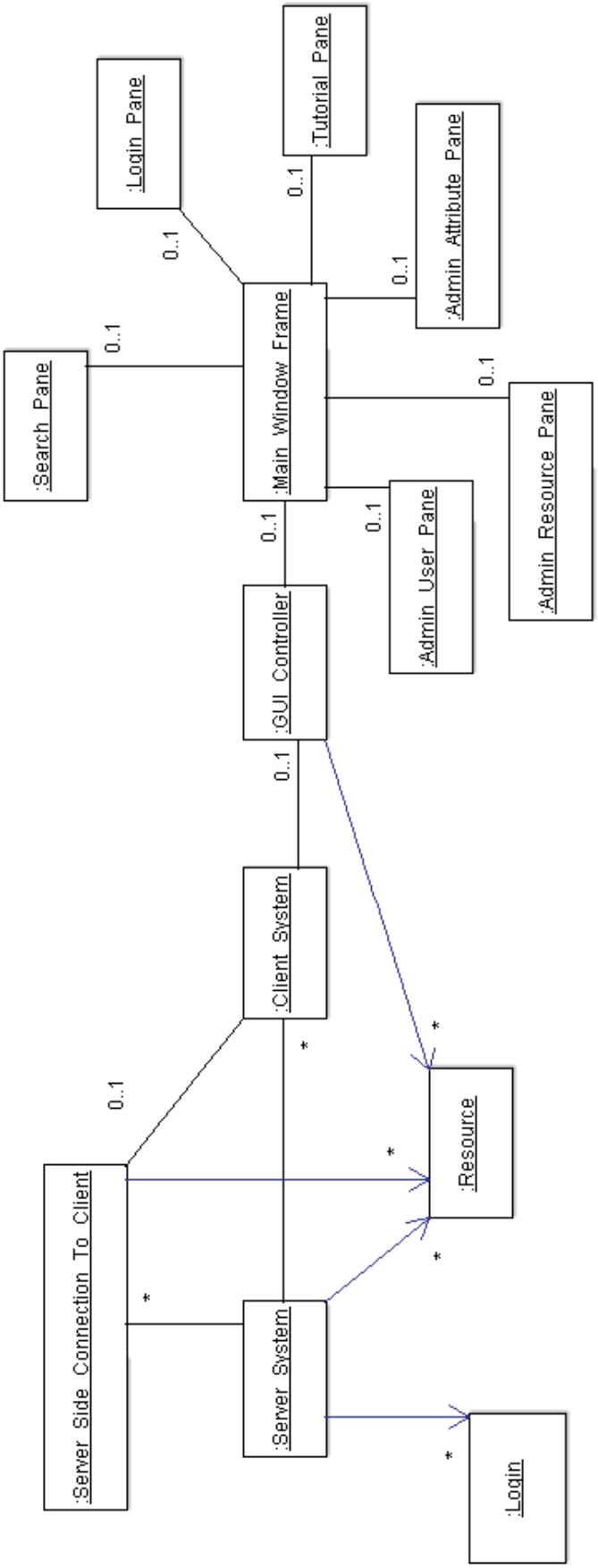
CLASS **Login_Pane**
RESPONSIBILITY 1. Show user input box for username, password, and the desired servers IP address and port
COLLABORATION 1. **Main_Window_Frame**

CLASS **Admin_User_Pane**
RESPONSIBILITY 1. Add new users to the database 2. Remove users from the database
COLLABORATION **1. GUI_Controller** **2. Search_Pane** **3. Main_Window_Frame**

CLASS **Admin_Resource_Pane**
RESPONSIBILITY 1. Add new resources to the database 2. Modify existing resources. 3. Delete existing resources.
COLLABORATION **1. GUI_Controller** **2. Main_Window_Frame**

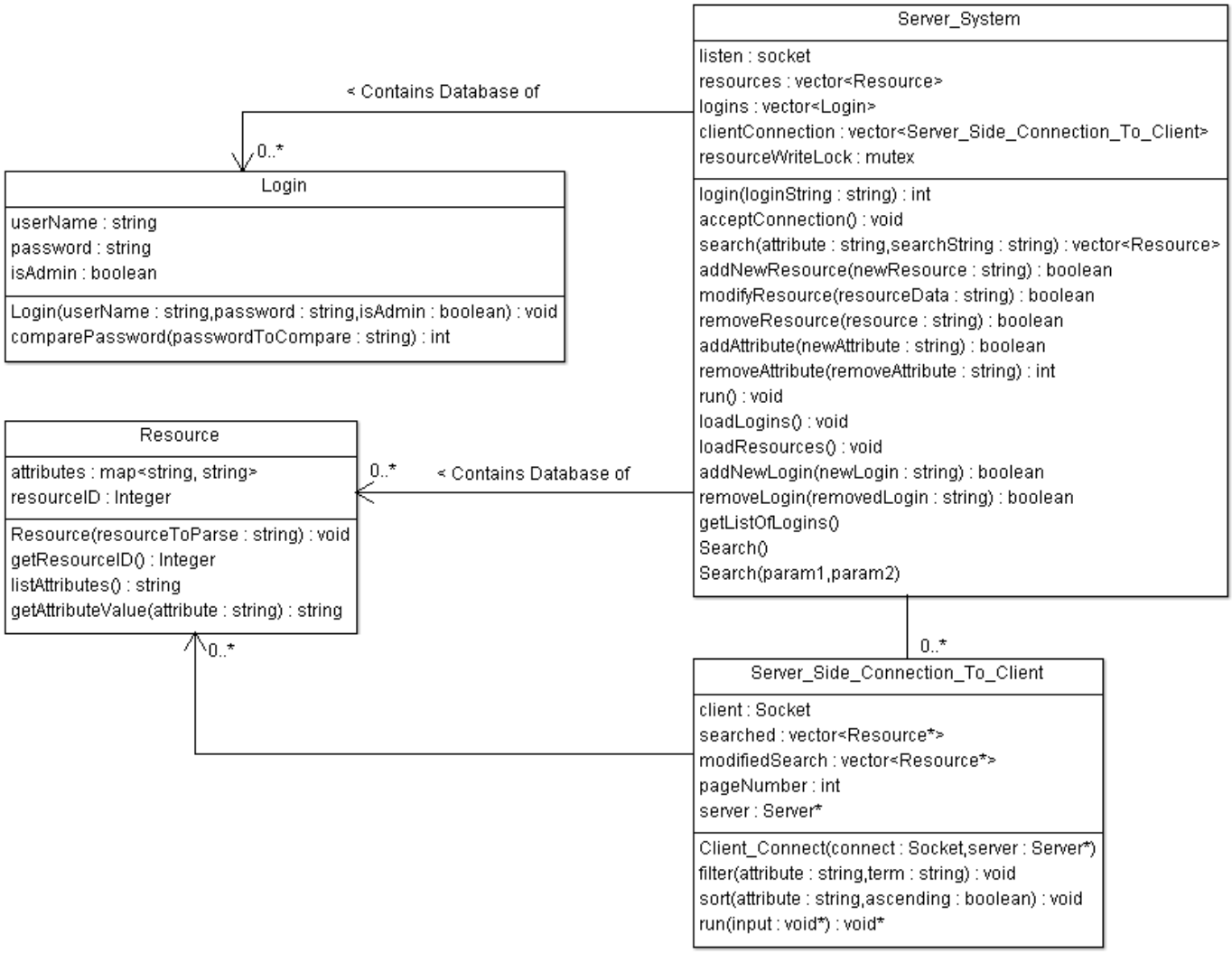
CLASS **Admin_Attribute_Pane**
RESPONSIBILITY 1. Add new attributes to the database 2. Remove attributes from the database
COLLABORATION 1. **GUI_Controller** **2. Main_Window_Frame**

2.2 Class Interconnections

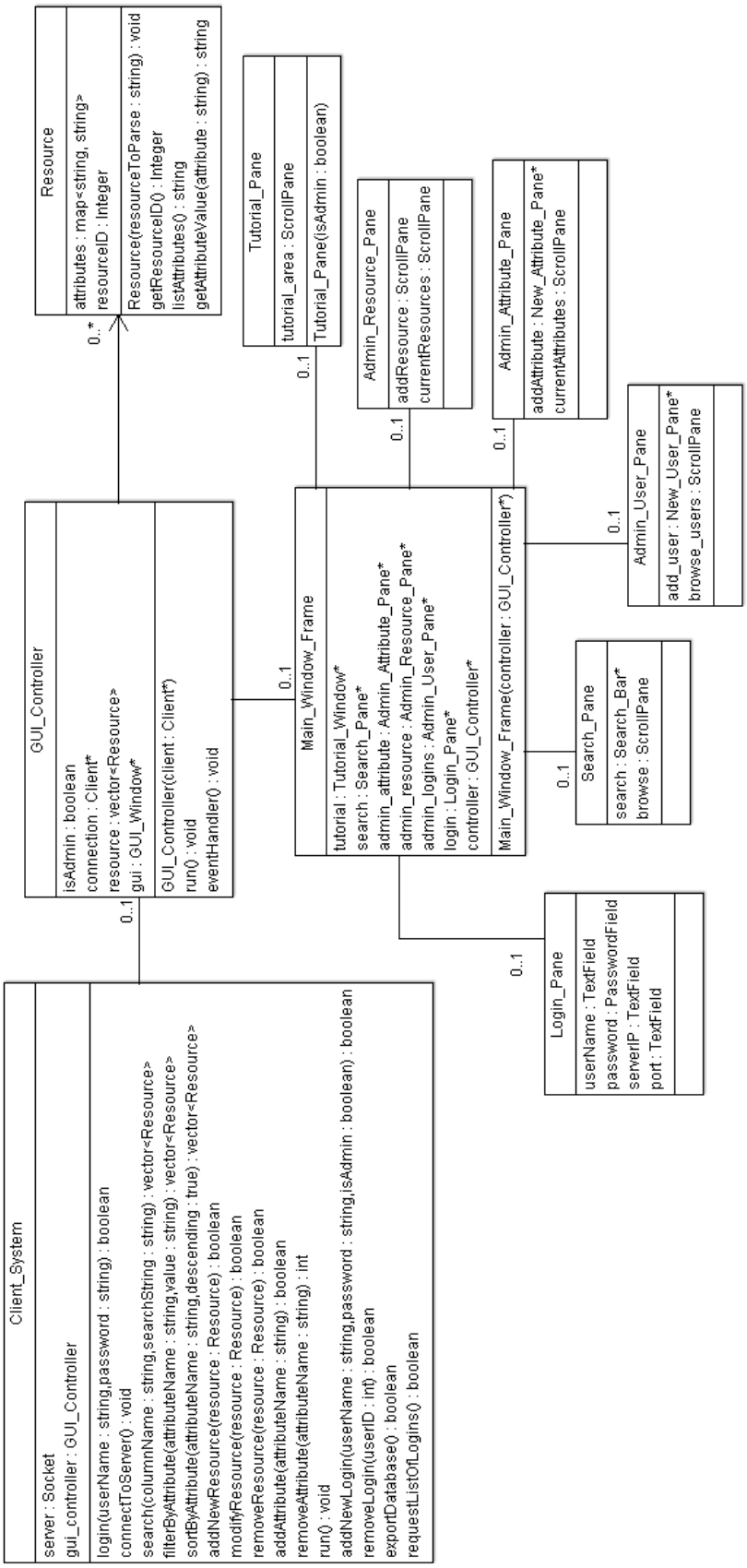


2.3 UML Class Diagrams

2.3.1 Server Class Diagram



2.3.2 Client Class Diagram



2.4 Graphical User Interface

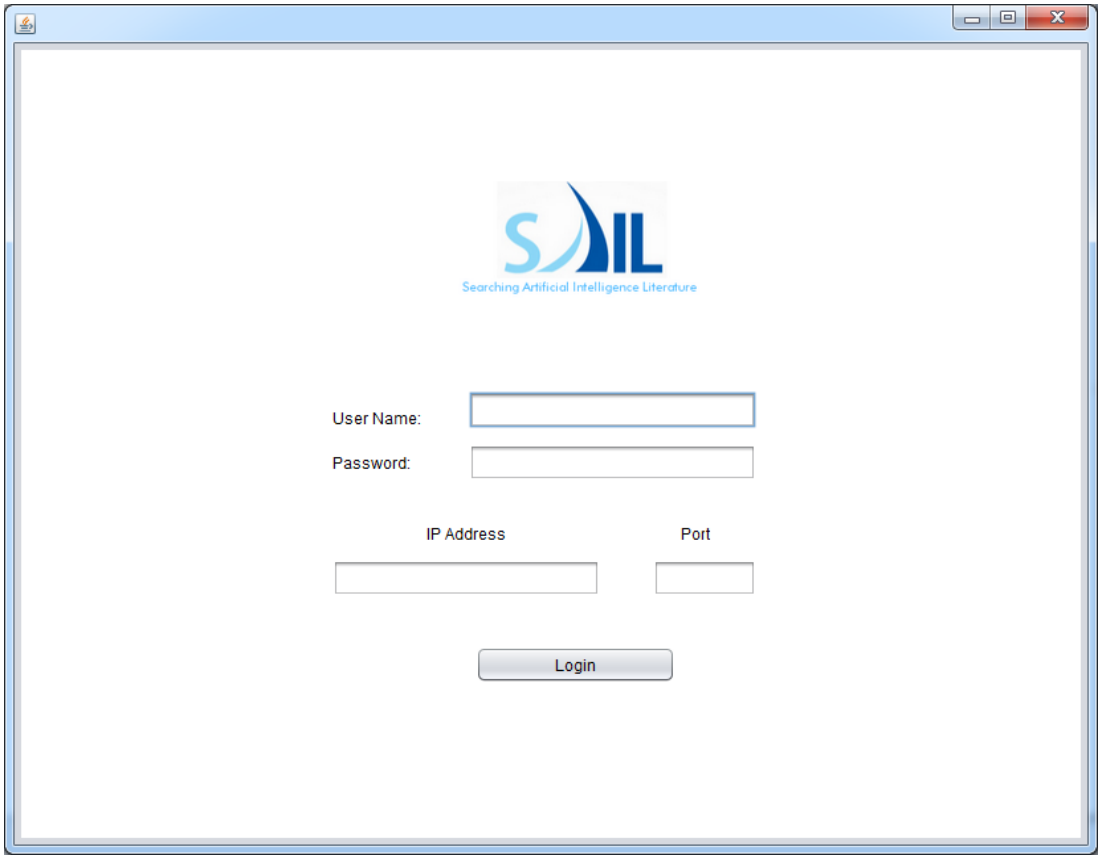


Figure1: Login screen

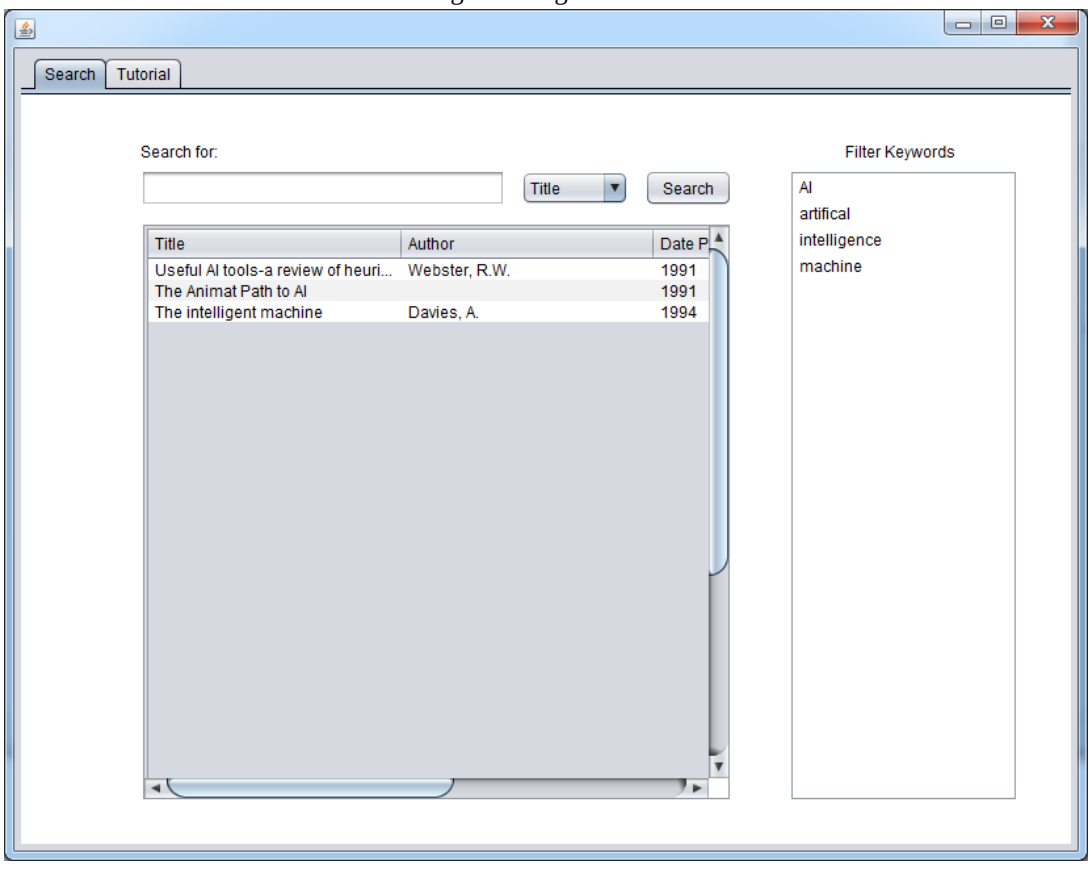


Figure 2: Search Window (Common to both users)

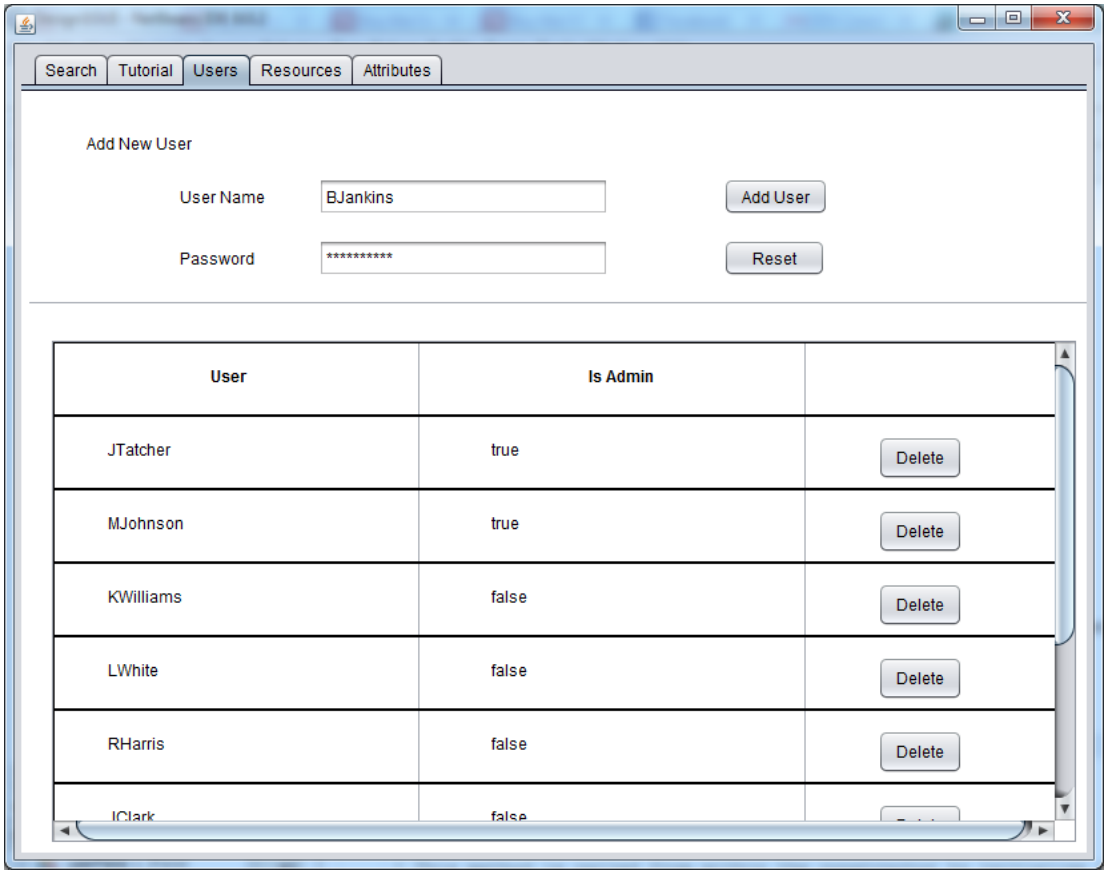


Figure 3: Administration panel for adding and removing users

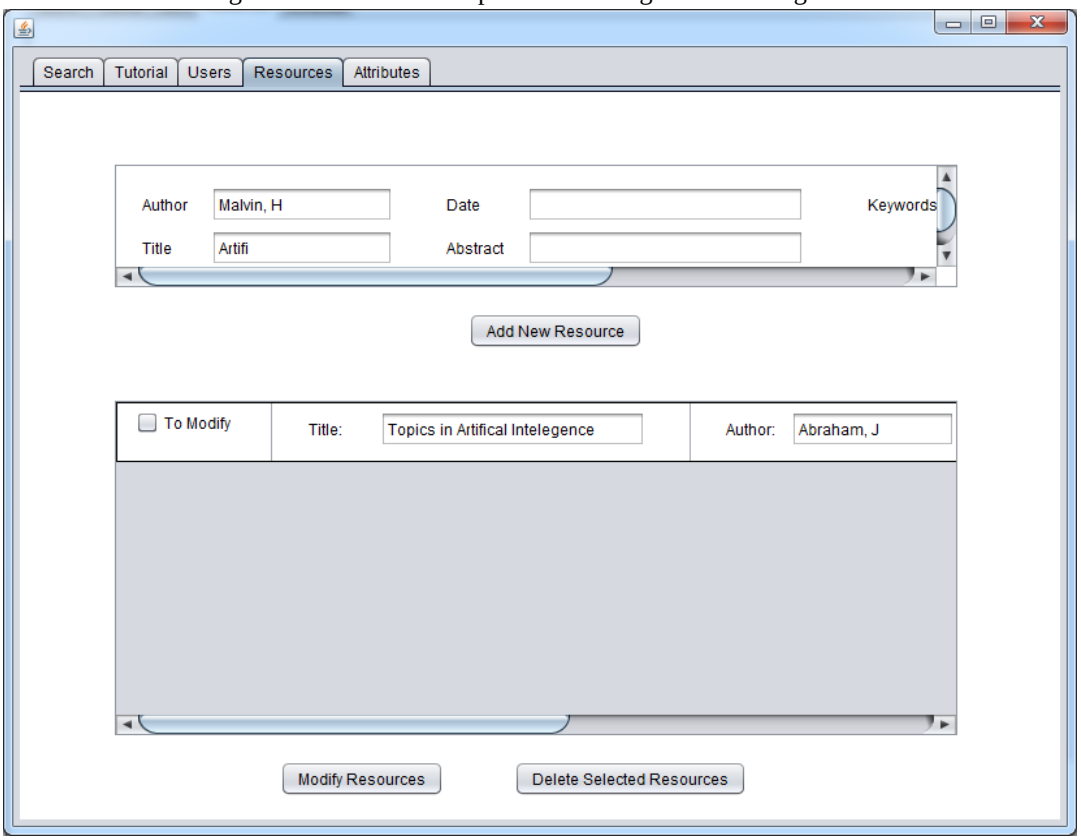


Figure 4: Administration panel for adding, modifying, and removing resources

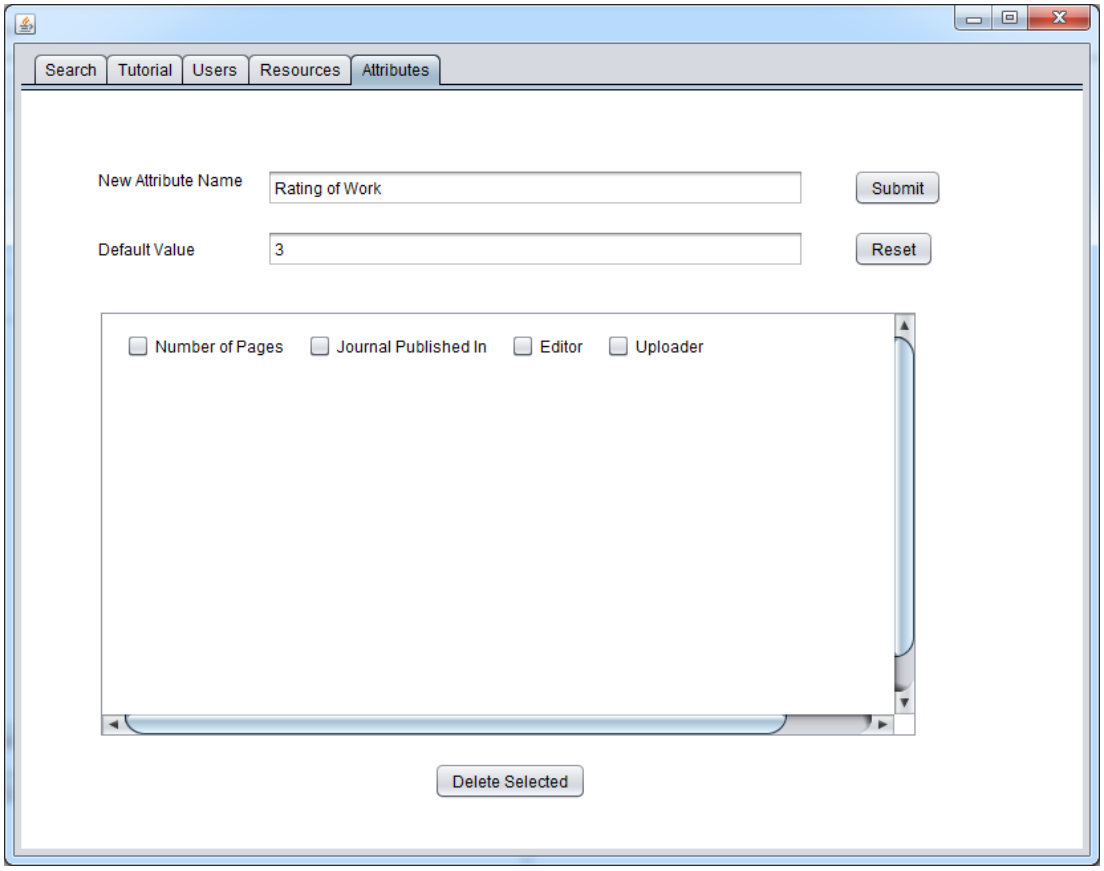
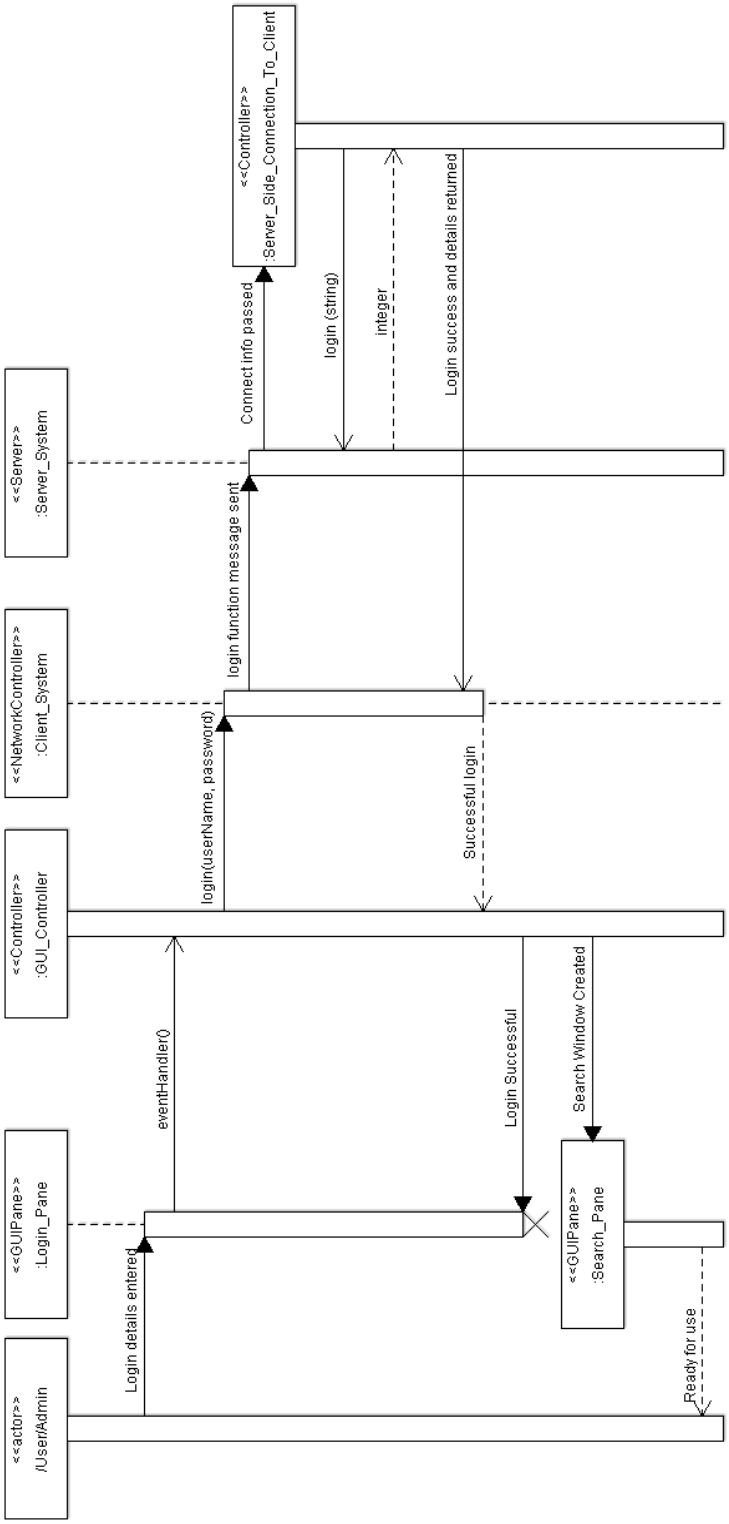


Figure 5: Administration panel for adding, and removing attributes

2.5 Sequence Diagrams

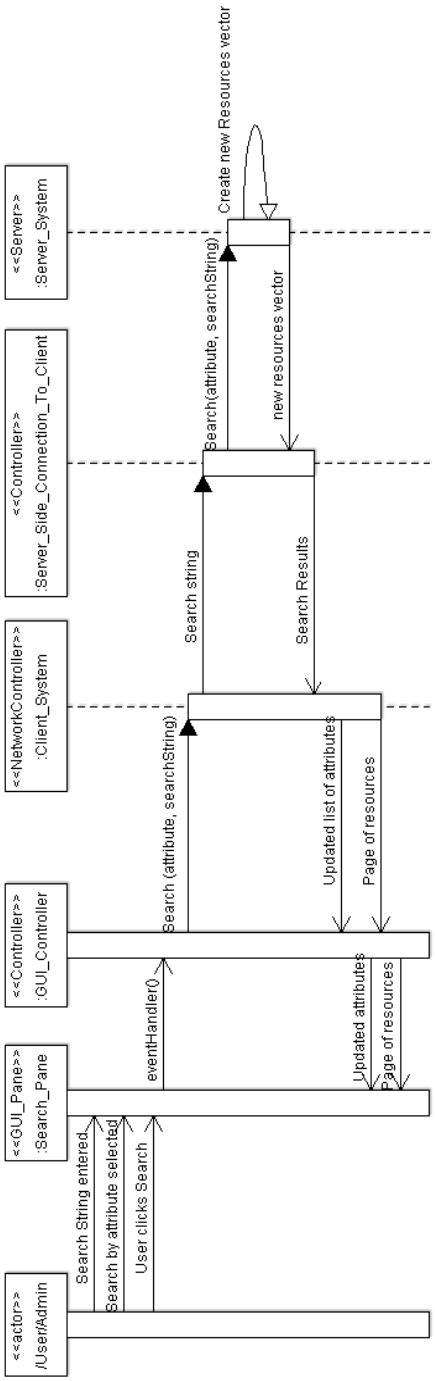
2.5.1 Login Sequence Diagram

This diagram shows the login process of a user to gain access to the database. A user types their login information into the client program's GUI, presses the submit button, and that data is sent to the server program. The server spawns a Server_Connection_To_Client object, and logs the user in. The server program then sends a confirmation and all relevant login information to the client program.



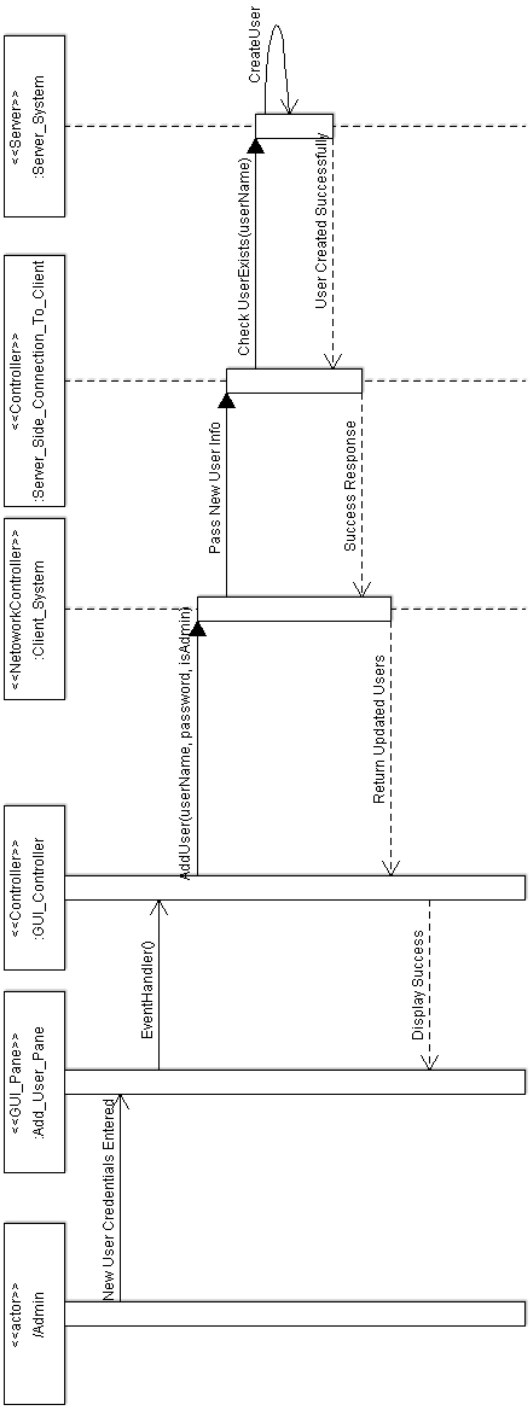
2.5.2 Search Sequence Diagram

This diagram shows a logged in user performing a search by some attribute. The user types in their search string, selects the attribute they want to search by, and clicks the search button. This gets caught by the GUI_Controller's event handler, and sends the search string and attribute to the Client_System. Client_System converts the search information to match the communication protocol, appending the search control code to the front. The Server_Side_Connection_To_Client checks the control code, and calls the Server_System Search function with the search string. A new vector of resources is created and returned to Server_Side_Connection_To_Client, which sends the first page of results back to the Client_System along with the current list of attributes. Client_System restores data into resources vector and sends to GUI_Controller to display.



2.5.3 Create User Sequence Diagram

The diagram shows the creation process of a new user. As the credentials are added and submitted to the server via the GUI, a connection is created with the server and the data is submitted. The server checks to ensure the user does not already exist, and if not creates the new user. Finally, it sends a success response to the client and updates the user table.



3. System Design

3.1 Server Design

3.1.1 Server_System run method

3.1.1.1 Description

When the server host starts up our client/server program, main method calls Server_System's run method, which will do all the initializing of the database and logins, then listen for new client connections and spawn a new thread for each connection. each thread will be running Server_Side_Connection_To_Client's run method.

3.1.1.2 Pseudo Code

```
CALL loadLogins
CALL loadResources
WHILE TRUE
        WAIT for incoming connection request
        CALL acceptConnection
ENDWHILE
```

3.1.2 Server_Side_Connection_To_Client run method

3.1.2.1 Description

When the Server_Side_Connection_To_Client receives data from the Client_System, it parses the request to determine what the client wants, and calls the appropriate methods. If the request is a search or log in, or any of the admin-specific functions, it calls the appropriate Server_System method. Otherwise, Server_Side_Connection_To_Client calls its own relevant methods.

3.1.2.2 Pseudo Code

```
WHILE TRUE
        WAIT for Client request
        READ first Byte of request to int Request
        CASE Request OF
                0x00:   send current list of attributes
                0x01:   INCREMENT pageNumber
                        send current list of attributes
                        send page(pageNumber)
                        break
                0x02:   DECREMENT pageNumber
                        send current list of attributes
                        send page(pageNumber)
                        break
                0x03:   SET Searched vector = (CALL Server_System's search method)
```

send current list of attributes
Generate Filter Keywords(searched)
set pageNumber to 1
send page(pageNumber)
break

0x04: CALL Sort method
send current list of attributes
set pageNumber to 1
send page(pageNumber)
break

0x05: CALL Filter method
send current list of attributes
Generate Filter Keywords(modified_search)
set pageNumber to 1
send page(pageNumber)
break

0x06: CALL Server_System's addAttribute method
send current list of attributes
break

0x07: CALL Server_System's removeAttribute method
send boolean pass/fail
send current list of attributes
break

0x08: CALL Server_System's addNewResource method
send boolean pass/fail
break

0x09: CALL Server_System's modifyResource method
send boolean pass/fail
break

0x0A: CALL Server_System's removeResource method
send boolean pass/fail
break

0x0B: send (CALL Server_System's getListOfLogins method)
break

0x0C: CALL Server_System's addNewLogin method
send boolean pass/fail
send (CALL Server_System's getListOfLogins method)
break

0x0D: CALL Server_System's removeLogin method
send boolean pass/fail
send (CALL Server_System's getListOfLogins method)
break

0x0E: Call Server_System's login method

```
                    IF login result is 9 THEN
                        close connection, exit loop.
                    ENDIF
                    send int login result
                    IF login result not fail_code THEN
                        send current list of attributes
                    ENDIF
                    break
                ENDCASE
ENDWHILE
```

3.1.3 Generate Filter Keywords (Server_Side_Connection_To_Client)

3.1.3.1. Description

Whenever the Client_System requests a new search (or filters already given search results) the Server_Side_Connection_To_Client creates a paired list of keywords and the number of times they are used from the resources and sends this list to the Client_System for the user to potentially filter their search results by.

3.1.3.2. Pseudo Code

```
SET keyword_count as a paired list that consists of a keyword and a number of occurrences
FOR each resource
    SET words to the extraction of all keywords in a resource
    FOR each keyword in words
        IF keyword is in keyword_count THEN
            INCREMENT occurrences of that keyword
        ELSE
            INSERT keyword to keyword_count
            SET occurrences of keyword in keyword_count to 1
        ENDIF
    ENDFOR
ENDFOR
SEND keywords to client program
```

3.1.4 Search Method (Server_System)

3.1.4.1. Description

Once a client's search request is forwarded from Server_Side_Connection_To_Client to the Server_System, then Server_System goes through each entry in its resources vector and creates a new vector of resources with a pointer to each match in the database. It then returns this new vector back to Server_Side_Connection_To_Client.

3.1.4.2. Pseudo Code

```
SET Search_Vector as an empty vector of resources
SET Current_Resource as pointer to head of resources vector
WHILE current resource not null
        IF (resource title matches searched title) THEN
                add resource to Search_Vector
        ENDIF
        SET Current_Resource to next resource
ENDWHILE
RETURN Search_Vector
```

3.1.5 Filter Method (Server_Side_Connection_To_Client)

3.1.5.1 Description

When Server_Side_Connection_To_Client receives a request to filter their results by given keyword, Server_Side_Connection_To_Client finds all resources already in the searched vector that have the keyword, and stores them all in the modified_search vector. It then sends the first page of this new modified search vector to the Client_System.

3.1.5.2 Pseudo Code

```
IF Modified_Search vector already has values THEN
        INIT Temp_Resources vector
        SET output pointer to point to Temp_Resources vector
        SET input pointer to point to Modified_Search vector
ELSE
        SET output pointer to point to Modified_Search vector
        SET input pointer to point to Searched vector
ENDIF
WHILE input pointer not null
        IF input resource has matching keyword THEN
                add input resource to output vector
        ENDIF
        INCREMENT input pointer
ENDWHILE
IF Temp_Resources not null THEN
        SET Modified_Search to Temp_Resources
        DELETE Temp_Resources
ENDIF
SET pageNumber to 1
SEND page to Client_System
```

3.2 Client Design

3.2.1 Client-Side Login

3.2.1.1. Description

Initially loading the application will show the main login screen. As the user enters their credentials and submits, a secure connection is made with the server. The entered id and password are then sent to the server to be compared with the existing user table.

If the user exists and entered their correct password, the server will send a response to the client allowing access. Otherwise, the server will respond to the client saying the credentials provided are invalid, at which point the user will be requested to enter the correct details.

Since the application has two modes, user and administrator, when an admin enters their credentials the server will respond accordingly. In turn, the client side window will show the correct graphical interface to the user depending on their level of access.

3.2.1.2. Pseudo Code

```
//Upon clicking login button
ACCEPT userid, password
SEND connection request to server
WAIT for server connection success

SEND userid, password to server
WAIT for server validation of credentials

IF credentials are valid THEN
        IF user is admin THEN
                Invoke GUI controller for admin interface
        ELSE
                Invoke GUI controller for user interfaces
        ENDIF
ELSE
        DISPLAY message to user for incorrect credentials
ENDIF
```

3.2.2 User/Admin Mode - Search

3.2.2.1. Description

Users are presented with a search interface where they can select the field to search by. Once the user has entered their criteria, the client will send the information to the server. The server side will then retrieve any found resources and return them to the client.

The client interface will display any found resources to the user along with relevant keywords they will be able to use for filtering the data.

3.2.2.2. Pseudo Code

```
//Upon clicking ‘Search’ button
ACCEPT search_criteria, search_field
SEND search_critria, search_field to server

WAIT for server to retrieve any resources

READ returned resources

IF resources found THEN
        DISPLAY resources
        DISPLAY relevant keywords for filtration
ELSE
        DISPLAY message to user, no resources found
ENDIF
```

3.2.3 Admin Mode - Add/Remove User

3.2.3.1. Description

Administrators have the added function to be able to add or remove users from the system. With this interface, users can enter their chosen credentials and be granted access to the system.

Another function is removing users who no longer require access to the application. This can be done by clicking the ‘Delete’ button for the specific user. The server will then remove the selected user from the system.

In both cases, an updated user list will be presented to the admin for review.

3.2.3.2. Pseudo Code

```
//Upon clicking ‘Add User’ button
ACCEPT userid, password, isAdmin
SEND userid, password, isAdmin to server

WAIT for server response

IF server response successful THEN
        DISPLAY message user was added

        WAIT for server to send updated user list
        READ updated user list
        DISPLAY updated user list in table
ELSE
        DISPLAY user not added //error case (ie. User exists)
ENDIF

//Upon clicking ‘Delete’ in user table
DISPLAY confirmation dialog to admin

IF admin confirms deletion THEN
        READ userid of row
        SEND userid to server for deletion

        WAIT for server confirmation
        DISPLAY message user was deleted

        WAIT for server to send updated user list
        READ updated user list
        DISPLAY updated user list in table
ELSE
        Take no action
ENDIF
```

3.2.4 Admin Mode - Add/Modify Resource

3.2.4.1. Description

Admins can add or modify resources from the client side using the provided interface. New resources are added by inputting the required data. The data is then sent to the server which will create a new field and populate it with the acquired information.

Modifications can be done in a similar way. Fields that need to be modified can be selected and changed. Once complete, all selected fields will be iterated and sent to the server for updating.

Updated resource lists will be presented after completion for review.

3.2.4.2. Pseudo Code

```
//Upon clicking ‘Add New Resource’
ACCEPT author, title, date, keywords etc. //any additional fields
SEND field data to server (indicate new resource)

WAIT for server response

IF server response successful THEN
        DISPLAY message resource was added
ELSE
        DISPLAY resource not added //error case
ENDIF

//Upon clicking ‘Modify Resources’
FOR each resource in admin table
        IF ‘ToModify’ is True THEN
                ACCEPT author, title, date, keywords etc. //any additional fields
                SEND field data to server (indicate existing resources)

                WAIT for server response

                IF server response successful THEN
                        INCREMENT successfully modified number
                        SET ‘ToModify’ to false //unsuccessful modifications will remain checked
                ELSE
                        INCREMENT unsuccessful modifications //error case
                ENDIF
        ENDIF
ENDFOR
DISPLAY message number modified, number unable to be modified
```

WAIT for server to send updated resource list
READ updated resource list
DISPLAY updated resources in admin table

3.3 GUI Design

Here is some sample functionality the GUI and GUI Controller provides, as well as some Pseudo code of how it is performed.

3.3.1 Initial Startup Process

3.3.1.1. Description

When the client program is started, a GUI Controller is created. The GUI Controller spawns a GUI Frame, in which a Login window is displayed. The user enters their login name, password, the IP of the server, and what port number they are connecting to. When the user presses the enter button, the GUI Controller calls the login method from the Client object. If the response from the server is a success, the GUI Controller will destroy the login window, and then create the user interface for using the database program. If the login is a fail, a message is displayed saying there was an issue.

3.3.1.2. Pseudo Code of startup from GUI Controller perspective

```
CREATE GUI window frame
CREATE login pane inside window frame
        WAIT for login submission event

Submit button is pressed
CALL client login function
IF login is successful
        DESTROY login pane
        CREATE search pane
        CREATE tutorial pane
        IF user is “admin”
        THEN
                CREATE user pane
                CREATE attribute pane
                CREATE resource pane
        ENDIF
ENDIF

WAIT for user driven event
```

3.3.2 User uses search functionality

3.3.2.1. Description

The user selects an attribute to search by from the drop down lists of search attributes, enters the information they want to search for that is relevant to that attribute selected, and press the submit button. The GUI Controller receives the submit event, and calls the client's search method, which returns either a page of results as well as a list of filter keywords, or a fail message. If a success, display the page of results, otherwise display a fail message.

3.3.2.2. Pseudo Code of search function from GUI Controller perspective

```
Submit button event triggered
GUI Controller calls Search method from client, passing the search information

Client search method returns results
IF results not empty
THEN
        DISPLAY each result to the browse pane
        DISPLAY list of filter by keywords
ELSE
        DISPLAY "Search results not found" in the browse pane
ENDIF

WAIT for user driven event
```

3.3.2 Admin uses add resource functionality

3.3.2.1. Description

The admin adds the details to create a new resource, clicks submit. The GUI Controller calls the client's add resource method, passes it the new resource. The client returns a pass/fail message to the GUI Controller. The GUI Controller then tells the GUI to print the message.

3.3.2.2. Pseudo Code of add resource functionality from GUI Controller perspective

```
Submit button event triggered
GUI Controller calls the add resource method
Client returns a boolean value
IF returned value is TRUE
THEN
        DISPLAY "New Resource Added Successfully"
ELSE
        DISPLAY "There was an error adding the new resource."
ENDIF
WAIT for user driven event
```