

A WEB INFORMATION ORGANIZATION AND
MANAGEMENT SYSTEM

By

TYNAN D. GRAYSON

Bachelor of Science

Langston University

Langston, Oklahoma

1996

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 1999

A WEB INFORMATION ORGANIZATION AND
MANAGEMENT SYSTEM

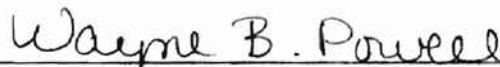
Thesis Approved:



Thesis Adviser



Jacques E. LaFrance



Wayne B. Powell
Dean of the Graduate College

ACKNOWLEDGEMENTS

I want to thank my husband Ralph A. Grayson for his love, patience, and profound friendship. He inspires me daily with his wisdom, strength and will to succeed. His patience and encouraging words gave me the determination to see the writing of this thesis through.

I also want express my sincere gratitude to Dr. K. M. George, my principal adviser, for giving me invaluable advice throughout my graduate study. His guidance and generous aid helped make this work possible.

I am grateful to Dr. G. E. Hedrick and Dr. Jacques LaFrance who gave me support and advice to guide me through the thesis writing process. They helped me shape and organize my work.

My brother, Milton L. Anderson, Jr., and his wife Lynn, deserve my deep gratitude for being a constant source of support in my life. I am truly indebted to my parents, Milton L. Anderson, Sr., and Carolyn, for cheering me on in pursuit of my dreams.

Last, but certainly not least, I want to thank God for giving his only begotten Son, Jesus Christ, to die for my sin, so that I may have abundant life. With God I have nothing to lose and everything to gain.

TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION	1
II. REVIEW OF THE LITERATURE	4
III. DEVELOPMENT TOOLS	13
3.1 HyperTextMarkup Language (HTML)	13
3.2 JavaScript	14
IV. DESIGN AND IMPLEMENTATION	17
4.1 Current Functionality	17
4.2 Improved and Customized Functionality	19
4.3 Event Driven Programming in the Client/Server Environment	20
4.4 Implementation	25
4.4.1 Window Heirarchy	25
4.4.2 Driver Frame	26
4.4.3 Viewer Frame	27
4.4.4 Navigation	27
4.4.5 Document Description	27
4.4.6 User Assigned Priority/Category	28
4.4.7 Storing Entries	29
4.4.8 Viewing Lists	31
4.4.9 Save, Retrieve, and Delete List	32
4.4.10 Keyword Search	34
4.4.11 WWW Search	35
4.4.12 Help Menu	36
4.4.13 History	38
V. SUMMARY, CONCLUSION AND FUTURE WORK	40
5.1 Summary	40
5.2 Contributions and Conclusions	40
5.3 Future Work	42
BIBLIOGRAPHY	43

APPENDIXES	46
APPENDIX I--IMPLEMENTATION SOURCE CODE	46
APPENDIX II--GLOSSARY	69
APPENDIX III--LIST OF ACRONYMS	70

LIST OF TABLES

Table	Page
I. WIOMS vs. Similar Tools	42

LIST OF FIGURES

Figure	Page
1. WWW Resource Discovery System's architecture	7
2. <i>NaviGuidor</i> functionality	10
3. PowerBookmarks architecture	12
4. The current history view	18
5. The WIOMS model based on event-driven programming	22
6. Event Loop	23
7. JavaScript Object Event Handlers	24
8. The WIOMS interface	25
9. WIOMS interface with New Window in background	26
10. Example of document description	27
11. Expanded view of 'Store Entry' select object	30
12. Store Entry and Keyword Search prompts	31
13. Results of selecting 'Show List' option	32
14. Save, retrieve, and delete list prompts	34
15. WWW search interface	35
16. Result of WWW search using AltaVista for 'car'	36
17. WIOMS help menu	37
18. WIOMS help menu with 'Show/Modify Entry selected'	38
19. Result of clicking 'Show History'	39

20. JavaScript permission request alert	19
---	----

CHAPTER I

INTRODUCTION

Just over a decade ago the World Wide Web (WWW) was a distributed internal network of computers sharing information via a text-based browser [SS97]. It has grown to become a massive loosely configured web of several networks of computers located at sites all around the world including, but not limited to, schools, organizations, corporations, and individual homes [Cap98]. The browser is a software application used to display HyperText Markup Language and other formatted documents. Previously the browser interface was text based, but it has developed into a tool with a Graphical User Interface (GUI) that enables users to explore the Web and view multimedia documents that not only contain text, but also pictures, video clips and sound, on widely varied topics. In addition to accomplishing these tasks, the browser maintains ease of movement between sites via hypertext technology.

There are several browsers and Netscape Navigator is one of the most popular. The Netscape browser includes many convenient features including a history feature that maintains a list with the Uniform Resource Locator (URL), which is the unique address of a web page or file, of each site the user visits [Cap98]. The URLs are added to the list without regard to any criteria even though the user may judge the sites with varying degrees of personal interest, importance, or relevance. Therefore the history file potentially could grow to contain a list of documents that is completely meaningless to

the user. In this case the user would face the time-consuming task of searching through the list to find the site where he/she wants to return.

The history feature also lists the title of the document associated with a particular URL. HyperText Markup Language (HTML) is the language that web pages are often written in and the title is an HTML feature intended to provide a brief description of the document to help the user locate the site in order to revisit it. The title is both optional and completely under the control of the writer of the document. If the title is omitted or, in the opinion of the viewer, ill suited to the document it will be hard if not impossible for the user to relocate it in the list.

Among the additional features of the Netscape browser is a bookmarking feature which allows users to save the URL in a folder they name. The problem with this feature is that the user must determine when a bookmark is no longer valid and delete it manually. Also the user cannot add adequate descriptions to each entry.

Studies have revealed problems associated with navigating the WWW [BEA99] which make it clear that it is beneficial to the user to have the capability of filtering to retain only the relevant URLs and to add what he/she feels are more appropriate descriptions when necessary. Users need to be able to locate and return to previously visited URLs. They need the ability to organize collected information. To achieve this I propose the design of a Web Information Organization and Management System (WIOMS) to eliminate the possibility of having a useless history list or a list without accurate document descriptions and to improve user ability to organize URLs and add content descriptions. WIOMS will allow assignment of priority to a site based on user defined criteria at the time of the visit and it will allow the user to decide whether or not a

site should be added to the history list at all. In addition the WIOMS will allow the user to include a description for documents that exclude or have unclear or inaccurate titles. These features will increase overall user control of history while ensuring that the list it maintains is personalized and contains only those documents of importance or relevance to the user.

CHAPTER II

REVIEW OF THE LITERATURE

The rapid expansion of the World Wide Web has fueled growth in the number of applications created to improve its efficiency, security and usefulness [IP96][FGR96]. Although browser technology has grown to include more features there remain three predominant challenges in producing applications for the web. First, the needs and practical expectations of the user must be determined. Second, any software solution must sufficiently address these needs and be easy to use [IP96]. Third, to facilitate ease of use, the user interface of any applications should be well designed and neither confusing nor inefficient [Gal97].

Vast resources have been devoted to understanding and developing technologies for the World Wide Web while relatively little research has been devoted to the needs of the user [KP96]. To develop effective tools we must find the answers to many questions. What do people actually do with their browser? What would they like to be able to do? Since it can be difficult for users to find the information they want in the first place, what kinds of methods are available for users to relocate that information at a later time [KP96]? How likely are people to use features of the application? Can this likelihood be improved?

Hypertext technology has driven growth in the number of resources available on the World Wide Web [Wit96]. The largest problem facing users with this growth is

accessing and managing this information [Wit96]. Once access is gained another problem is that a lot of the information a user finds when searching for a particular topic may not be relevant to his interest [LEA95]. Also, to fully evaluate the relevance of the information, the user would have to read through the actual document, which is time consuming [LEA95]. Even if the user were able to take the time to read through the information they may not be able to digest all of the information presented [LEA95]. In this case an efficient method for relocating the information would be useful.

According to Instone and Pemberton [IP96], the number of users on the World Wide Web doubles annually. Thus, it is useful for any web tool to analyze user trends to build a design that has value for users. Kehoe and Pitkow [KP96], state that these trends are discovered by analyzing user response to “widely distributed, heterogeneous, ... WWW-based” surveys. These analyses made clear the fact that most people, 78.4%, who navigate the web do so using a browser, and do so almost daily [KP96]. More than a quarter of respondents reported that not being able to organize the information they found as a problem and it was revealed that 23.7% of users reported that not being able to relocate a page once visited was also a problem [KP96]. Some users even reported that web browsers were “either poorly designed or did not work well” [KP96].

Several authors have addressed the problem of resource discovery. Lam et al. developed a method of keeping track of new information as it became available despite the large number of servers and pages on the web. The most popular method of alleviating the ‘resource discovery problem’ involves allowing the user to do a keyword search to locate URLs [LEA95]. This provides the user with information that may or may not be relevant [LEA95]. This strategy does not effectively keep track of the ever-

changing information available. The best tool would not only be able to track 'non-coordinated changes', but also "detect any inconsistencies caused by independent modifications, and organize information into a structured index which enables efficient search" [LEA95]. Lam et al., presents a resource discovery tool that uses a web indexer robot to build a database for keyword searching. This is good given the dynamic Web environment [LEA95]. According to [BEA99] 27.8% of users responding to a survey of 11,700 people report organizing collected information among their problems and 12.16 report finding pages already visited as a problem. This problem is significant since according to [BEA99] a study on user revisitation patterns to WWW pages found that 88% of individuals pages are revisits.

The tool developed by Lam et al. also addresses the fact that the potentially germane 'hits' returned by the search might not be relevant to the user. They alleviate this concern by providing a "relevance feedback mechanism which is supported by the user interface" [LEA95]. The mechanism refines a search until a desired set of documents is obtained [LEA95]. To judge this relevance the user interface presents a hierarchical map of the links within 'hits' and the links within neighboring pages [LEA95]. The tool also facilitates sharing of information between users by allowing the user to save his query so that someone else can re-use it at a later time [LEA95]. This is all clearly illustrated in figure 1. The implementation also makes clear the importance of the user interface. The developers found that users should not be made to set too many parameters and the process to use a tool should already be somewhat familiar to the user [LEA95].

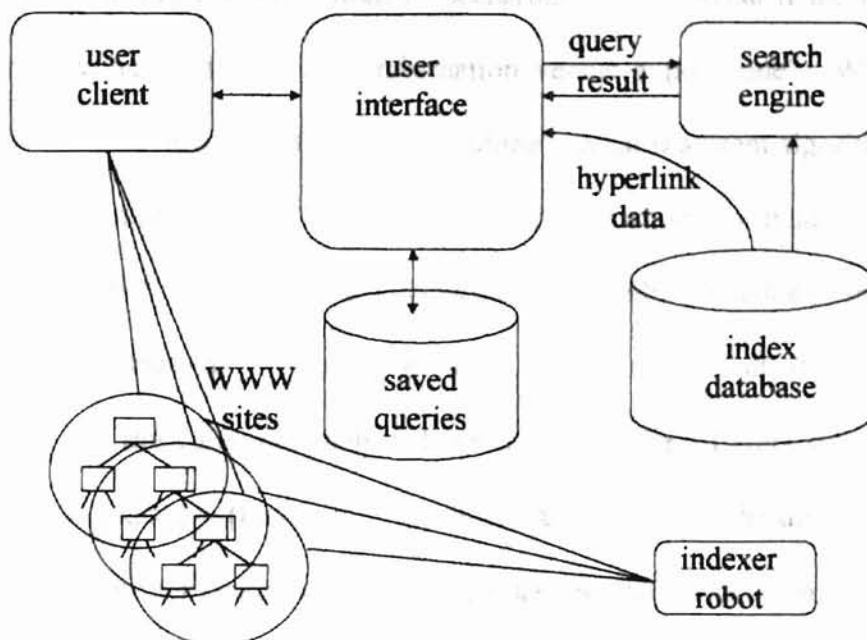


Figure 1. WWW Resource Discovery System's architecture adopted from ILEA95I.

Accessing and managing all the information available on the web is a problem analyzed by Kent Wittenburg [Wit96]. There is a lack of research beyond html-based tables of contents. The problems extend to commercial Intranets, which can grow in complexity at a rate similar to that of the World Wide Web [Wit96]. He suggests that the amount of information available is greater than the user's ability to absorb it, and that there is a need for arranging and filtering search results interactively [Wit96]. Wittenburg recommends that this problem be addressed by new human-computer interfaces that enable users to filter in conjunction with search and navigation instead of using these things in isolation.

Resulting from the fast expansion of the web are two major problems: slow access and "failure to find information that is known to be available [Pir98]". Peter Pirolli states that designers of browsers for such a rapidly growing medium should be concerned with alleviating both of these problems [Pir98]. According to Peter Pirolli [Pir98], designers

should go through several 'what if' scenarios such as; What if the user is faced with a repository rich with relevant information versus a poor one? What if the user has unlimited time as opposed to a hard deadline? What if system algorithms are made faster rather than more accurate? Finally, what if presentations are made more informative but slower to read? In the technique suggested by Pirolli, the states of interaction, including the changes that can be made from state to state as a result of user actions should be delineated. Basically, the analyst should represent the different states abstractly such as the browser display [Pir98]. He/she should also represent the different changes that can be made between states including the changes that result as a result of user actions [Pir98]. "This defines an abstract state space representing the possible paths that human-computer interactions may take" [Pir98]. Values are specified for the expected number of relevant documents that will be found and costs are associated with the time it takes the user to find them [Pir98]. Pirolli then evaluated the different browser interfaces using *dynamic programming*. Dynamic programming was used to search through all possible paths of human-computer interaction, evaluating their costs and values, and determining the best paths [Pir98].

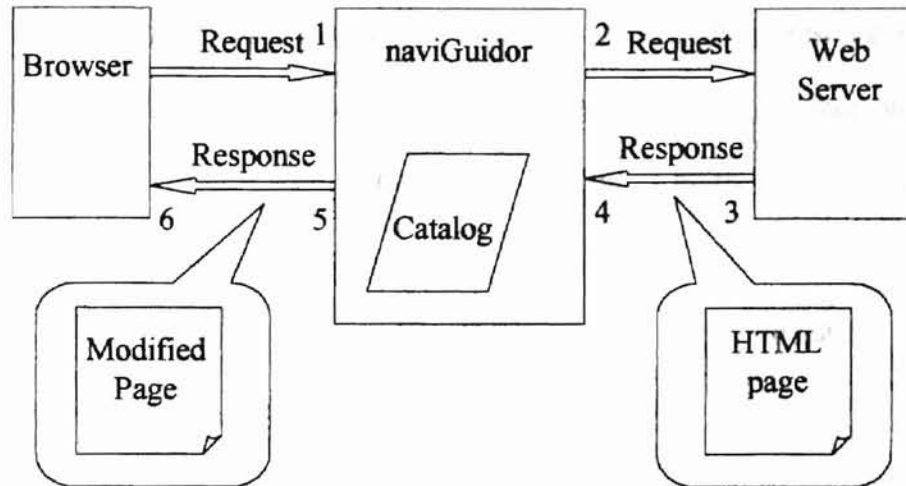
Norm Archer, Milena Head and Yufei Yuan developed a tool called the Memory Extender Mechanism for Online Searching (MEMOS) for Netscape. This tool allows the user to store the navigational history in a model that more closely resembles the user's own mental picture than traditional history storage methods including pushdown stacks [AHY99]. The stack is not an accurate trace of the user's movements since it may pop several pages from the stack or add duplicate information to the stack. MEMOS also allows the user to access the history in the form of a frequency list that ranks the sites in

order of the number of visits and to save history of navigation sessions for future use. Empirical studies conducted by the developers found that it is important not only to aid users in finding new information but also in recovering old information [AHY99]. It was also found that “tools that create smaller, more manageable information subsets, can reduce the negative consequences associated with information overload” [AHY99]. This reduction effects faster and more precise information gathering [AHY99].

Yossi Matias and Godi Wolfman developed a tool called *naviGuidor* that is intended to add value to the Web navigation experience of users through the addition of features that improve referencing, filtering, and various annotations. Although methods for directing the navigation of users have been explored before, Matias and Wolfman believe that these features should address the individuality of users and the diversity of their skill levels. *NaviGuidor* controls and modifies the pages accessed by a user based upon their needs. The tool gives the user the opportunity to specify these needs in advance [MW99].

The ultimate goal of *naviGuidor* is to utilize the power of the web and create the ideal navigation situation by offering convenient references to relevant resources while adding annotations of terms and more guidelines to each resource. *NaviGuidor* acts as a go-between for the browser and the web itself. The user can create specifications for the navigation-guiding features they want to use. These specifications “are written as *catalogs* in XML syntax, and are encapsulated in specific HTML pages” [MW99]. The user created catalogs can be maintained and easily shared because they can be imported into *naviGuidor* [MW99]. Using the specifications *naviGuidor* dynamically adds annotations to terms in HTML pages.

Figure 2, shows clearly the way the tool modifies the pages accessed by the user with explanations of what happens when a page is accessed.



Reading an HTML page. HTTP messages between the browser and the Web are transmitted through the naviGuidor: (1) An HTTP request from the browser is accepted and parsed by the naviGuidor. (2) The naviGuidor decides, depending on the current catalog, if the HTTP request is to be transmitted as is, modified or blocked. (3) A Web server accepting a forwarded request is responding with an HTTP message. (4) HTTP response is accepted and parsed by the naviGuidor. (5) The naviGuidor decides, depending on the current catalog, if the HTTP request is to be transmitted as is, modified or blocked. (6) The HTTP response is accepted at the browser.

Figure 2. NaviGuidor functionality adopted from [MW99].

Currently Ralph Grayson is developing a new tool called Persistent History Tool (PHT) that helps users have better navigation sessions by giving them access to past and present history lists. Grayson states that the user will benefit from the ability to use previous sessions as a starting point for the current session [Gra99]. The history list is a vital feature that allows users to go back to previous sites and retrieve resources that they have already gone through the trouble of finding. Sometimes users seek information about very specific topics and the entire session can be tedious. If the user has other sessions looking for other kinds of information the prior lists they have compiled could

either be lost if the history limit is too small or it can be hard to distinguish which sites match which topic [Gra99].

Persistence of the history list allows users to establish lists that are topic specific and it makes it easier to relocate and take advantage of resources they have found [Gra99]. The history lists stored in cookies easily can be manipulated via the user interface. Access to prior history lists is facilitated by the interface that allows the user enter the name or creation date of the list [Gra99]. The list is then loaded and the back and forward buttons of the interface allow the user to navigate to resources quickly [Gra99].

PowerBookmarks was developed to address resource relocation and organization problems. Designed by Corey Bufi, et al, PowerBookmarks utilizes query, modeling and navigation capabilities provided by WebDB, which is a hypermedia database system built on top of an Object-Oriented Database Management System [BEA99]. Additionally, PowerBookmarks supports information sharing with access control, automated bookmark classification based on document contents through an external classifier, and gives users the benefit of automated dead link and inactive link removal. Also, it monitors their behavior to provide automatic bookmarking for frequently accessed URLs. The architecture of PowerBookmarks is shown in figure 3.

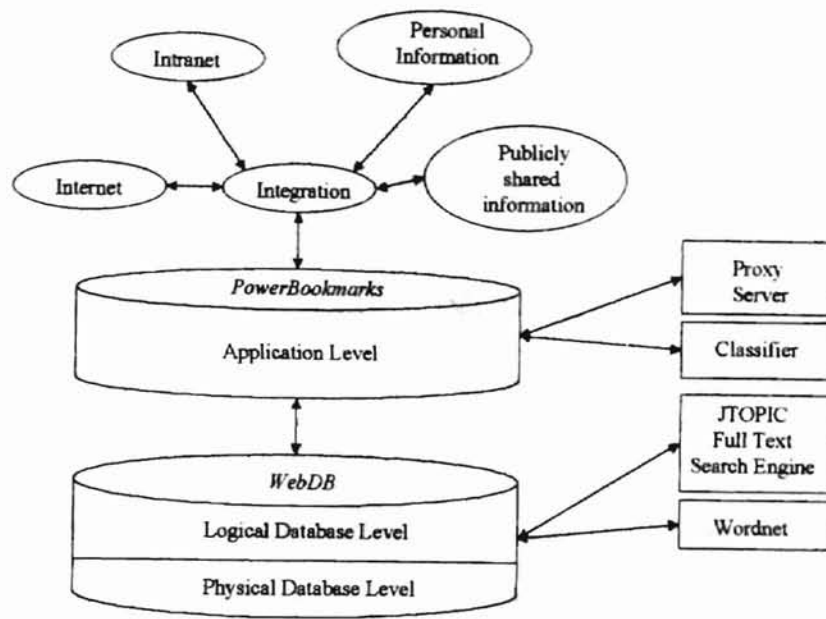


Figure 3. PowerBookmarks architecture adopted from [BEA99].

CHAPTER III

DEVELOPMENT TOOLS

3.1 HyperText Markup Language (HTML)

The browser is a client-side software application that allows the user to navigate the World Wide Web. The browser interprets HTML commands to format documents for the viewer. It gives users the ability to follow links in and see hypertext information. The browser interacting with the server accomplishes this. The user initiates a request for information or action and the server interprets the request and takes some action. Among the most popular graphical browsers is Netscape Navigator.

There are several tools that can be used to create browser enhancements that make the users navigation session more useful including Hypertext Markup Language (HTML) and JavaScript. HTML code uses a set of tags that tell the browser how to format, load and align text and graphics. Tags are commands that define the overall form of the HTML document and give basic structure to the way a page appears. Tags are not visible on the browser, but their effects are. A tag might note that a line should be a title or a heading, for example.

Each tag is enclosed in angled brackets. Paired tags are different in that the last tag has a forward slash just before the command. Commands are not case sensitive, but are usually written in uppercase to promote clarity by making commands easier to spot

when reading an HTML file. Hypertext links are special tags that link one page to another page or resource. When a mouse is placed over a link and clicked, the browser jumps to the link's destination.

3.2 JavaScript

JavaScript is a new technology that was developed initially by Netscape under the name LiveScript. It is intended to extend the capabilities of basic HTML. JavaScript usually resides between the `<SCRIPT>...</SCRIPT>` tags in ordinary HTML documents. It gives developers the ability to write scripts that interact with objects within a web page, such as forms, frames, and background color. In its current state it is more closely linked to Java which is why the name was eventually changed. It is designed to allow logic to exist on the client side to perform tasks such as data validation.

JavaScript is different from Java in that it is not as strict or sophisticated as Java. Java is an object-oriented programming language and JavaScript is object-based. Java has "strong typing (all variable data types must be declared), static binding (object references must exist at compile time), and is compiled into bytecode. The bytecode is then interpreted. In contrast, JavaScript has loose-typing and dynamic binding [FS98]. JavaScript is strictly interpreted not compiled even though the term "JavaScript compiler" is commonly used to refer to the built-in browser mechanism that reads the code and executes it or produces an error message. Both JavaScript and Java can be used to make web pages more sophisticated and exciting by executing the 'local code' [Way97]. The biggest difference is that JavaScript will *only* run on a browser. It is

tightly integrated into HTML whereas Java is simply connected to an HTML document through the <APPLET> tag and is stored in another file.

Two of the buzzwords associated with JavaScript are *object-oriented* and *interpreted*. The use of the term object-oriented can be misleading. The more accurate term of object-based should be used instead because although the single large hierarchy is just the basic data about the browser, its window, the document displayed, the forms, and the links within the document [Way97]. There is no greater class structure and no provision for inheritance.

JavaScript is interpreted [Way97]. Variables and functions can be defined on the fly and used several lines later. There is no compiler or preprocessor [Way97]. The disadvantage of being interpreted is that it takes longer for the code to execute because the browser compiles the instructions at runtime just before executing them [WEA97]. The advantage is that it is easier to update the source code. When you change the script in the source HTML file the new code is executed the next time the user accesses the document [Way97].

Other characteristics of JavaScript include its being event-driven [WEA97]. Most JavaScript code is written to respond to events generated by the user or the system. HTML objects, such as buttons, or text fields are enhanced to support event handlers [WEA97].

Finally JavaScript is a good multipurpose tool that allows developers to accomplish many goals. For example, it helps enhance static HTML pages, through special effects, animation, and banners [WEA97]. It permits validation of data without passing everything to the server and is a building block for client/server Web

applications. JavaScript serves as a bond between HTML objects, Java applets, and Netscape Plug-ins while providing connectivity without using a Common Gateway Interface [WEA97].

CHAPTER IV

DESIGN AND IMPLEMENTATION

4.1 Current Functionality

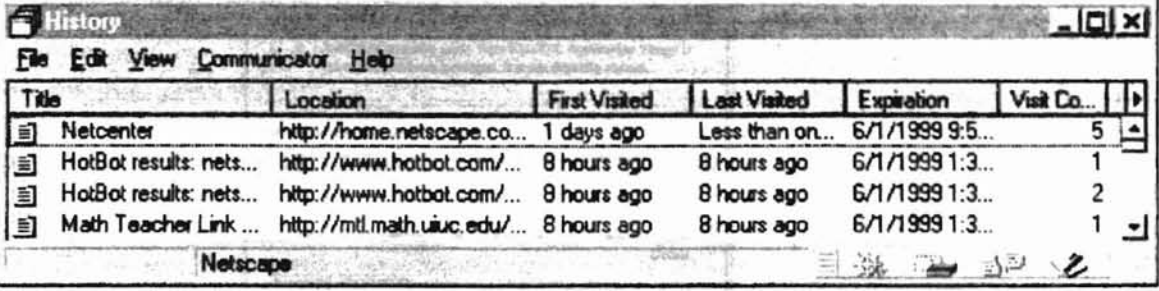
As a user surfs the web, the sites that he/she visits are logged in the history file. The history file includes the URL, the title of the web document, the date of the first and most recent visits, the expiration date, and the number of visits as shown in Figure 4. The user can view this file and double click on the URL of the site they wish to revisit. This information is compiled without consideration of the personal preferences of the user.

Efforts have been made to increase history and bookmarking functionality, and more features have been included from one version of the browser to the next.

Current user/history interaction is limited. A user cannot control which documents are stored in the list. The user can search the titles of the documents in the list, but he/she can only delete the sites the search turns up. He/she can look through the list manually and delete individual entries. Also the entire contents of history can be deleted at once either by the user or by the browser after a date which the user specifies. While these functions do serve a purpose, history should be more personalized to increase its overall usefulness.

Another browser feature is bookmarking, which allows document URLs to be stored in files that the user can name. One problem with this feature is that the user

cannot add document descriptions and the files have to be deleted manually. Another problem is that the process of creating files can be tedious.



The screenshot shows the 'History' window in Netscape. The window title is 'History' and it has a menu bar with 'File', 'Edit', 'View', 'Communicator', and 'Help'. Below the menu bar is a table with the following columns: 'Title', 'Location', 'First Visited', 'Last Visited', 'Expiration', and 'Visit Co...'. The table contains four entries:

Title	Location	First Visited	Last Visited	Expiration	Visit Co...
Netcenter	http://home.netscape.co...	1 days ago	Less than on...	6/1/1999 9:5...	5
HotBot results: nets...	http://www.hotbot.com/...	8 hours ago	8 hours ago	6/1/1999 1:3...	1
HotBot results: nets...	http://www.hotbot.com/...	8 hours ago	8 hours ago	6/1/1999 1:3...	2
Math Teacher Link ...	http://mtl.math.uiuc.edu/...	8 hours ago	8 hours ago	6/1/1999 1:3...	1

At the bottom of the window, the 'Netscape' logo is visible.

Figure 4 The current history view

Security features of the most recent versions of browser software make it difficult to access and manipulate the history file. The developers of Netscape found it necessary to enforce more strict limits on access to history information due to concerns about invasion of privacy [WEA97], and also because of malicious programmers who use the information for bad reasons [WEA97]. The WIOMS issues a request for the user to enable the Netscape Permission: Universal Browser Access, via a prompt initiated by the first user/WIOMS interaction. As shown in figure 20, this prompt alerts the user that a JavaScript is requesting permission to read or modify browser data. Although the user may choose to deny this permission, he/she should click the button labeled 'GRANT' to allow the WIOMS to access this private information.

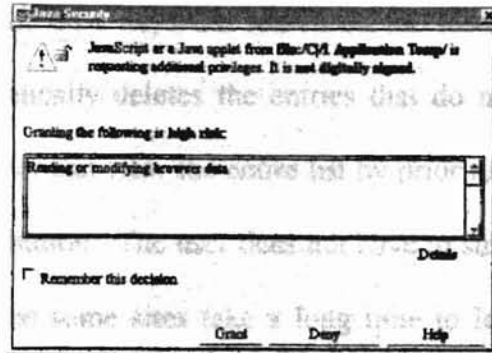


Figure 20. JavaScript permission request alert.

To modify or enhance the power of history and bookmark function, one needs to modify the browser, which is labor intensive and very time consuming. The alternative is to develop tools on top of the browser that incorporate both features and extend user control while ensuring that the retained information is relevant to them.

4.2 Improved and Customized Functionality

The WIOMS focuses on the improvement of the functionality of the navigation, bookmark organization, and document management while increasing the level of user control. It utilizes storage, event driven programming, Netscape security, Object-based JavaScript programming and HTML document generation. The user maintains the current history and bookmarking functionality provided by the browser while increasing the level of control and personal usefulness. In addition to the uses mentioned earlier, the WIOMS allows the user to make snap judgments about the worthiness of documents as it

pertains to inclusion in the list. He/she is able to assign a priority number or category name based on any criterion he/she chooses to those documents that are to be included. Not only can he/she do a keyword search on the history list, but he/she can also do a search that automatically deletes the entries that do not contain the search string in a single step. The user can view the entire list by priority/category or simply delete it with a single click of a button. The user does not have to surf to a site to include it in the list. This is useful, since some sites take a long time to load and the user may already be familiar with the contents and want to store the entry as part of a category the resource pertains to. WIOMS also incorporates WWW search capability and allows the user to view history with a single button click.

4.3 Event Driven Programming in the Client/Server Environment

Software development is addressed by several computing models among them is the client/server model. Event-driven programming, currently emphasized in the marketplace according to Dann and Woodworth, makes effective use of this model. Event-driven programming supports the development of a more sophisticated user interface in the client/server computing model [Eas91]. The WIOMS is developed using event-driven programming in a client server environment. In event-driven programming developers are concerned with assigning responses to events rather than execution sequences as in traditional application programming, where execution begins at the top of the program, then flows through function calls and control flow statements in a fairly predictable manner; the program is in control.

The WIOMS is an event-driven program in which the user controls and decides which portion of the program gets executed. Clicking the mouse generates a mouse event, pressing a key generates a keystroke event. The user clicks the buttons or uses the keyboard to select menu options, tab through controls and so on. Each mouse click and key press generates an event to which the program must respond. For example, the WIOMS contains the code for the 'Clicked' event of the 'View Description' button, but only when the user actually clicks on the button will this code be executed.

The Web Information Organization and Management System (WIOMS) is constructed using the development tools discussed in chapter 3. The key elements of the WIOMS are:

- GUI components
- Layout
- Listening state
- Event processing

The GUI components, such as text fields and buttons, are the screen elements that are manipulated by the user with the mouse and keyboard. The layout governs how the components appear on the screen. The listening state is one in which the program waits for events to be triggered by the user. Events signal important user actions like a mouse click.

The WIOMS responds to events, generated by GUI components that indicate that specific actions have occurred. The program waits in the listening state for these events. The entire program consists of the code that presents the GUI to the user and the specific code that is executed when events occur. Upon loading, the program immediately enters

the listening state. When an event occurs, the program enters the active state in which events are handled. After the event is handled the WIOMS modifies the GUI, if necessary, and returns to the listening state (See figures 5 and 6).

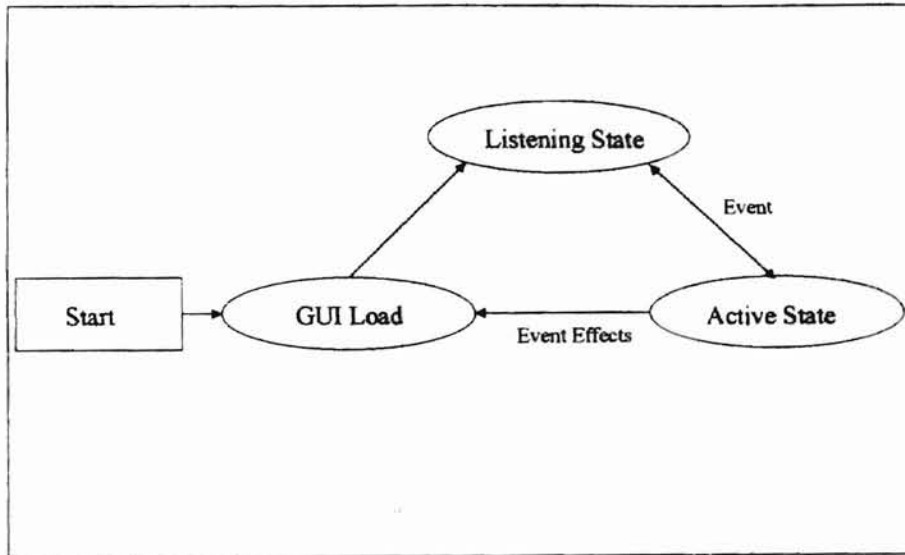


Figure 5. The WIOMS model based on event-driven programming.

```
Start(PHNT)

Event = NULL
WHILE (Event != QUIT)
    WHILE (Event == NULL)
    {
        // listen for event
    }
    IF (Event != QUIT)
    {
        // handle event
        IF (Effect_GUI == TRUE)
            // modify GUI
    }
    Event = NULL
}
```

Figure 6. Event Loop

Events involving GUI components such as buttons, textboxes and select objects are recognized and interpreted by JavaScript event handlers (See figure 7). These handlers automatically respond when an event occurs and transfer control to custom methods, which are executed. Components can have multiple event handlers for example: the Select object has onChange, onFocus, and onBlur event handlers.

<i>Object</i>	onClick	onSubmit	onChange	onFocus	onBlur	onSelect
button	■					
text			■	■	■	■
textarea			■	■	■	■
select			■	■	■	

Figure 7. JavaScript Object Event Handlers. Adapted from [WEA97].

Figure 8 shows the WIOMS interface that consists of windows, frames, buttons, textboxes, and prompts, tables and labels. The figure also illustrates the point and click nature of the design. The frames organize the WIOMS into two sections, a top (driver) frame and a bottom (viewer) frame, that reside within a parent frame. The viewer frame is where the results of all user actions are displayed with the exception of viewing multimedia documents. This takes place in a separate window, which is below the main window in the hierarchy (See figure 9). The driver frame houses the JavaScript that runs the WIOMS and provides interconnection for the set of functions that the user has access to. The buttons, which are located in the driver frame, provide the user with some actions whether it be searching, viewing, or deleting the entire history list, navigating to another document, assigning a priority or category name document, or deleting the part of history which do not meet the search criteria.

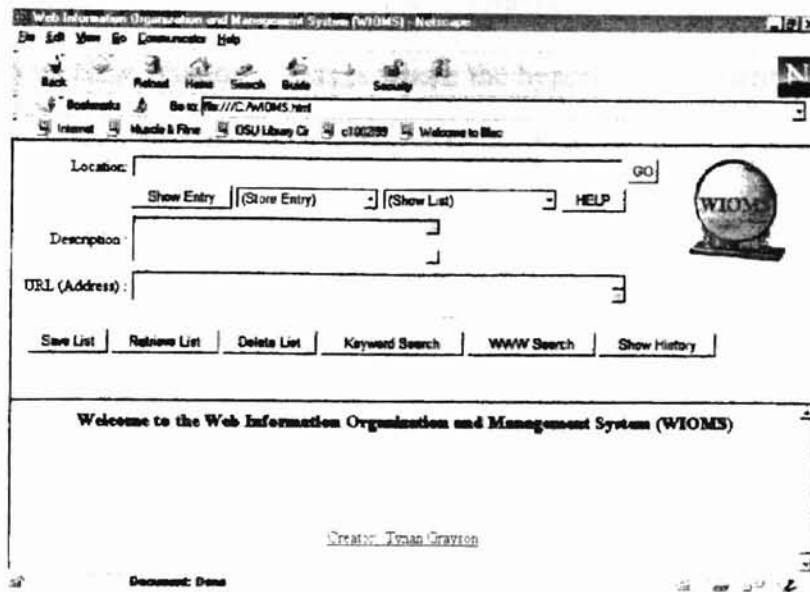


Figure 8. The WIOMS interface.

4.4 Implementation

In order to effect this implementation the WIOMS, written using HTML and JavaScript, maintains each new list temporarily in an array when the WIOMS is in use and, for a length of time specified by the user, in a cookie otherwise. The WIOMS resides entirely on the client side and uses a frame oriented interface to enhance ease of use and user communication with the lists as shown in figure 8. This facilitates the point and click nature of the tool.

4.4.1 Window Hierarchy

When the WIOMS is loaded the main window contains two frames one above the other on the display. The top frame is called the Driver Frame and contains the code for the GUI and the code that is executed in response to user generated events. The lower frame is called the Viewer Frame and is used to show documents generated by the program in

the Driver Frame. An extra window is generated upon loading the WIOMS, referred to simply as New Window. This is where the hypertext documents to which the user surfs are displayed. Initially this window is blurred which means it does not gain focus and is placed on the desktop behind the main window. To focus the window the user can click in the New Window. The New Window has all the functionality of a normal browser window and the user can load any Web document or program that the browser can interpret (see figure 9).

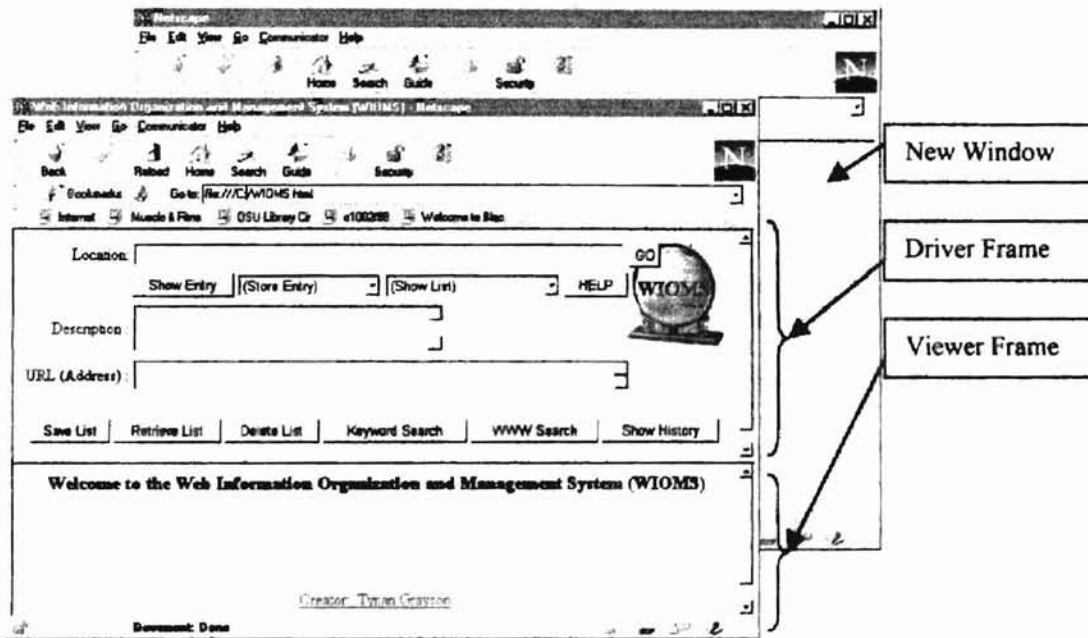


Figure 9. WIOMS interface with New Window in background.

4.4.2 Driver Frame

The most prominent part of the GUI is the Driver Frame. This frame allows the user to control the application. The buttons, textboxes, and select objects allow the user

to initiate events such as viewing the URL and description of the document loaded in the New Window. Figure 9 clearly shows the driver frame in its initial state.

4.4.3 Viewer Frame

The bottom portion of the main window is dedicated to the viewer frame. In this frame HTML documents generated in response to user initiated events are displayed. Initially this frame displays an HTML document with a welcome message. The code for this initial document is located in a file named `WIOMS_viewer.html` and must be stored in the same directory as the code for the driver frame and the main window.

4.4.4 Navigation

Upon loading in the main window, the WIOMS generates a new browser window behind the main window. The main window remains in focus. As the user navigates the WWW the documents are displayed here. There are several methods by which the user can navigate. He/she can enter a URL in the textbox labeled 'Location' in the Driver Frame and press ENTER or click the button labeled 'GO' immediately following the textbox. He/she can also navigate by using the location bar of the New Window, following a link embedded in the current document or clicking the 'GO' button following an entry in the Viewer Frame as part of a document generated by the WIOMS.

4.4.5 Document Description

The title and URL of the current document can be displayed by clicking the 'Show Description' button in the Driver Frame. As shown in figure 10, the title, if one

exists, is displayed in the textarea labeled 'Description' and the document URL is displayed in the textarea labeled 'URL (Address)':.

The user can choose to save the entry as is, with the current information, or change it prior to saving. If the user chooses a new description it should be entered in the textarea. No limit is placed on the length.

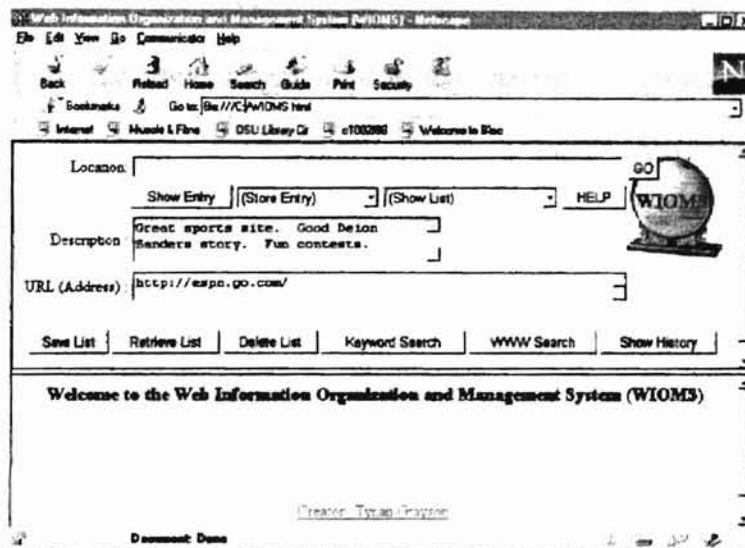


Figure 10. Example of document description.

4.4.6 User Assigned Priority/Category

The implementation allows the user to assign each entry to one of five lists. The lists can be used to divide entries by up to five categories or priority level.

Prioritization

A user can assign these in the manner he/she feels best suits his/her personal need. For example, a user who is shopping online for Star Wars memorabilia can store "must-

have” items in the list assigned a priority of 1, while storing items that he/she will buy only if the others are not available in a list assigned a priority of 5.

Categorization

The user could decide to just use category names of Must Have and Might Buy One Day to be more clear. Another user might want to use this capability to aid in dividing research findings among up to five different sub-categories. For example a user researching Algorithms might want to use the sub-categories of (1) Graph Algorithms, (2) Search Algorithms, (3) Sorting Algorithms, and (4) Amortized Analysis. Notice that the user is not required to use all five categories.

4.4.7 Storing Entries

Once the user determines that an entry is satisfactory he/she can store it in one of the five category/priority lists. Initially the lists are named (1) Store Category/Priority 1, (2) Store Category/Priority 2, (3) Store Category/Priority 3, (4) Store Category/Priority 3, and (5) Store Category/Priority 5. Clicking the small triangle to the right of the Select object labeled ‘Store Entry’ can access each list. The selection options appear as shown in figure 11.

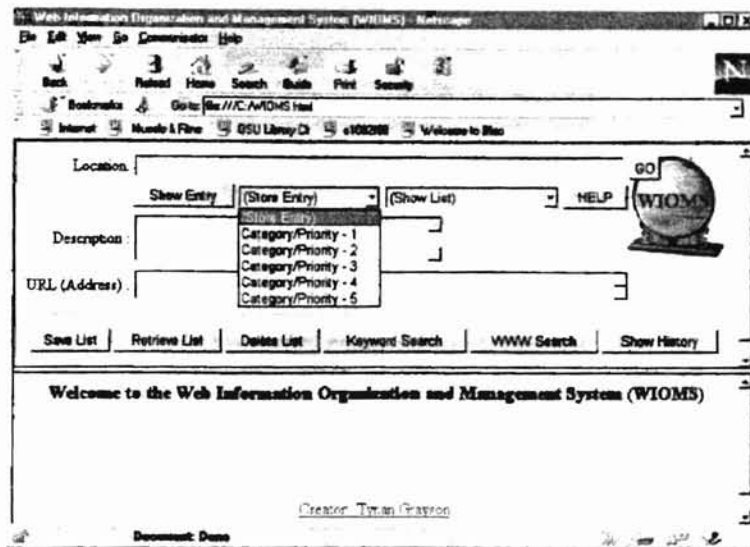


Figure 11. Expanded view of 'Store Entry' select object.

Upon choosing a category/priority list the user is prompted to choose a new name for the category if it is the first time during the current session that the list has been accessed. See figure 12a. The default name will become the permanent name unless the user enters a new name here. After choosing the list name, the entry—consisting of the description and URL of the hypertext document—is added to the list.

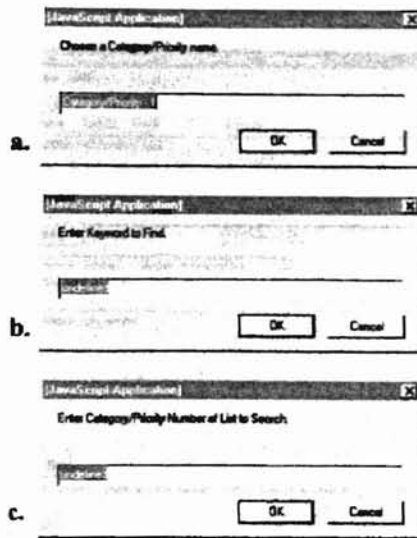


Figure 12. Store Entry and Keyword Search prompts.
a. Category/Priority list name prompt.
b. Keyword search string prompt
c. Keyword search list prompt

4.4.8 Showing Lists

To print a list in the Viewer Frame, the user should choose the name of the list from the select object labeled 'Show List'. Upon clicking the down arrow to the right of the selection object the list names are shown. The list names will match those given in the 'Store Entry' select object options. When the user selects the name of the list to be viewed. The contents of the list are displayed in the viewer frame. The result of the "show list" operation is shown in figure 13.

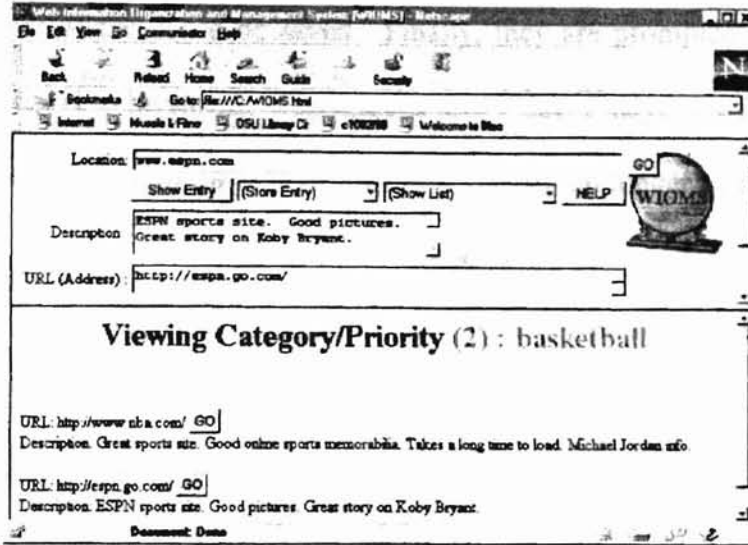


Figure 13. Result of selecting 'Show List' option.

Each entry is displayed in the following format:

Description	
URL	<input type="button" value="GO"/>

Clicking the 'GO' button following the URL results in the desired document, described in the entry, being loaded in the New Window.

4.4.9 Save, Retrieve, and Delete List

Figure 14 shows the prompts generated by the methods used to save, retrieve and delete lists. The WIOMS allows the user to save lists between browser sessions in *persistent client state HTTP cookies* which allow information to be stored on the client browser's computer for retrieval at a later date. Cookies are currently the most powerful technique available for maintaining a state between sessions. Clicking the button labeled 'Save List' triggers an `onClick` event which is handled by the `getSetCookieValues` method. First, the

user is prompted for the name they want to give the list. Second, he/she is prompted for the number of the list to be saved. Finally, they are prompted to choose the length of time the list is to be stored. The options are: 1 day, 3 days, 1 week, and 2 weeks. After the specified length of time, the list expires and can no longer be retrieved.

To remove a list before the designated expiration date, the user can initiate the `clearCookie` method by clicking the 'Delete List' button on the interface. At the prompt he/she must enter a valid name of a saved list. The expiration date of the list is set to three days prior to the current date, which effectively deletes the cookie.

The `onClick` event corresponding to the 'Retrieve List' button is handled by the `getRetrieveCookieValues` method, which prompts the user for the name of the saved list. Next, the user is prompted for the category/priority number they want to use for the list during the current browser session. The `getRetrieveCookieValues` method invokes the `getCookie` method, which retrieves the list. If the category/priority number the user selects is already associated with a list in the current session not empty, then the retrieved list is added after the current list contents.

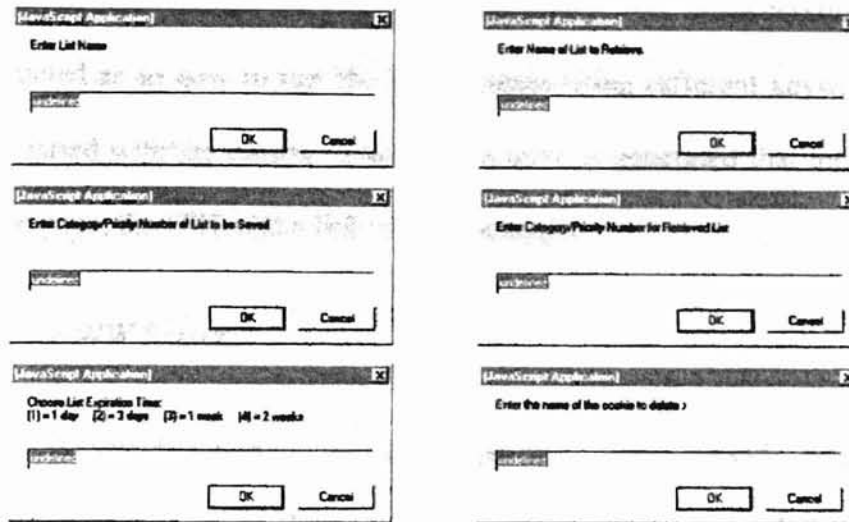


Figure 14. Save, retrieve and delete list prompts.

4.4.10 Keyword Search

Searching the history takes place via a JavaScript embedded in the Driver Frame and the JavaScript frame object prompt (message, response) is used. When the user clicks the search button and a dialog box with a text entry field is displayed prompting the user to enter keywords for the search as shown in figure 12b. The user response in the text entry field is returned as a string and the search is not case sensitive. Next, the user will be prompted for the number of the list they want to search. This number corresponds to the list position in the select object of the Driver Frame as shown in Figure 12c. The descriptions of the documents in the list are searched for matches to the keyword string.

An HTML document is then generated by the script, which displays the results of the search. If no entries were found which contained the keywords, then a message is

displayed that informs the user that the search was unsuccessful. The user will be instructed as to how to run the search again using different keywords in case they are unsatisfied with the results. Otherwise a table is generated that includes the description of the page, the URL and a link to the document.

4.4.11 WWW Search

A search engine is software that lets a user specify search terms and finds documents that include those terms on the WWW. The WIOMS includes a WWW search feature, which allows the user to conduct a search using one of two search engines, AltaVista and Yahoo. This is accomplished via a second JavaScript program within the Driver Frame. Clicking the button labeled 'WWW Search' calls the webSearch method which opens a document in the Viewer Frame as shown in figure 15.

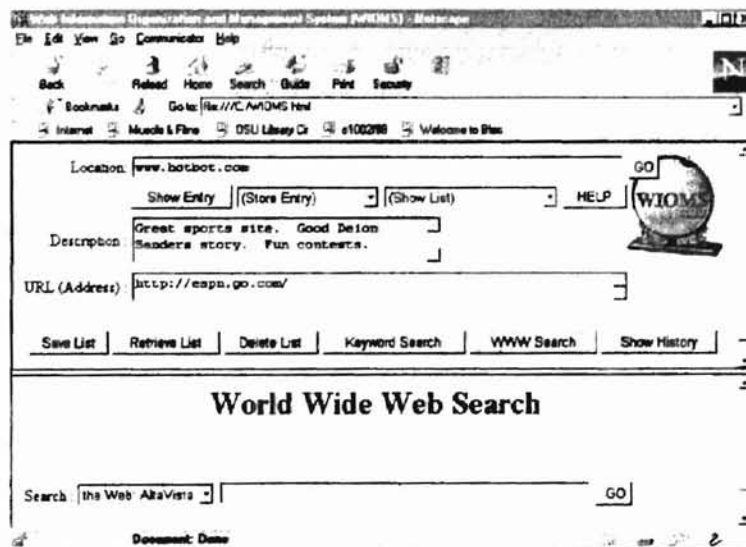


Figure 15. WWW search interface.

The document includes a select object, a text object and a button. The selection box options are the names of the search engines available to the user via the WIOMS. Upon selecting a search engine the user should type the search terms in the text box and press ENTER or click the button labeled 'GO'. The chosen search engine is accessed with the search terms and the results are displayed in the Viewer Frame. Figure 16 shows the result of conducting a search using AltaVista with the search term 'car'.

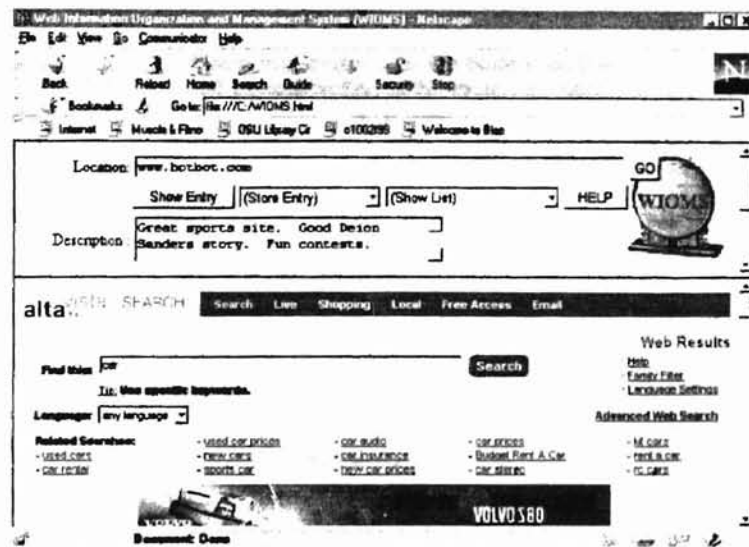


Figure 16. Result of WWW search using AltaVista for 'car'.

4.4.12 Help Menu

The help menu is accessed by clicking the button labeled 'HELP' on the Driver Frame. This action causes the HTML file containing the help menu to open. It is displayed in a new window and is designed to help the user understand how to use the features of the WIOMS (refer to figure 17). Each help category is assigned a button as

part of the help menu interface. Placing the mouse over any of these buttons triggers an onmouseover event which highlights the button and displays the help information indicated by the button label. Figure 18 shows the interface with the 'Show/Modify Entry' button selected.

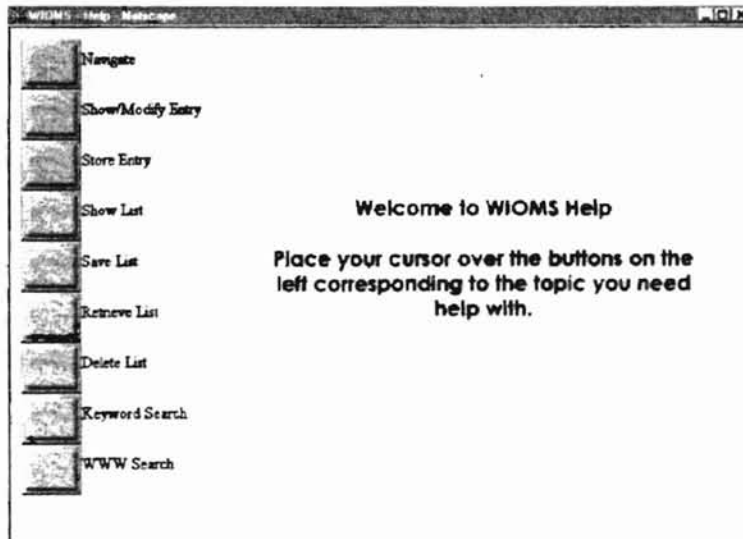


Figure 17. WIOMS help menu.

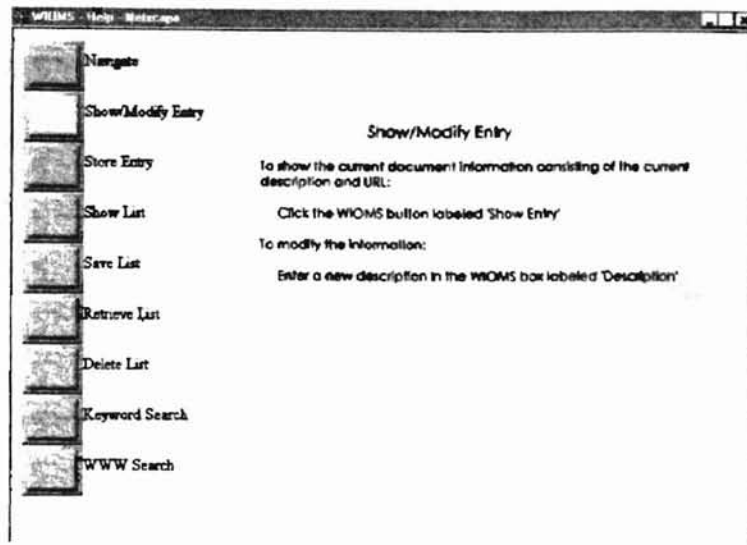


Figure 18. WIOMS help menu with 'Show/Modify Entry' selected.

4.4.13 History

The history list of the current browser session can be accessed by initiating the onClick event of the 'Show History' button in the Driver Frame. The showHistory method handles the event and produces an alert box containing the Uniform Resource Locator of each document in the history list. An example of the alert box produced by the showHistory method is shown in figure 19.

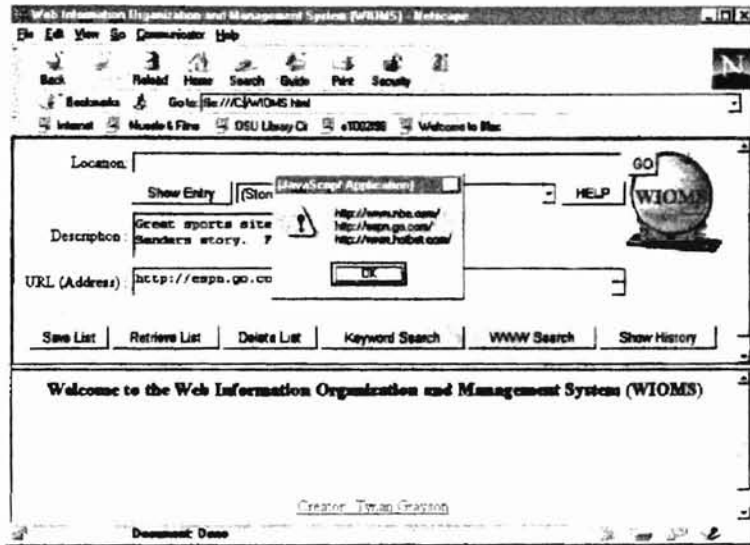


Figure 19. Result of clicking 'Show History'.

CHAPTER V

SUMMARY, CONCLUSION, AND FUTURE WORK

5.1 Summary

The World Wide Web is growing at a rapid pace. Trends in web navigation patterns have been studied by several authors, and recent surveys reveal that among the primary needs of the expanding user base are resource relocation, document relevance determination, and organization of found information. The history and bookmarking features of the Netscape Navigator browser address some of these needs. However, these features are inadequate. As a more complete answer this thesis proposes a Web Information Organization and Management System (WIOMS). The proposed WIOMS incorporates both current browser features, and additional capabilities in order to address user needs more completely.

5.2 Contributions and Conclusions

The major contribution of this thesis is a Web Information Organization and Management System (WIOMS). The WIOMS uses event-driven programming in a client/server environment. It is a tool that works on top of existing browser software. It gives the user the ability to view document descriptions and Uniform Resource Locators (URL) with a single button click, modify the descriptions to suit their need, add entries

(consisting of the description and URL) to lists based on user determined category/priority criteria. In addition, WIOMS includes functions that allow users to navigate the web, view category/priority lists, save lists in *persistent client-state HTTP cookies*, retrieve lists for use in future browser sessions, or delete a list all together. The WIOMS also incorporates the ability to perform a keyword search on any of the user-compiled lists as well as to search the World Wide Web per the AltaVista or Yahoo search engines. There is a button that, when clicked, displays a history consisting of all URLs visited during the current browser session, a help menu to aid user concerning WIOMS utilization, and all WIOMS features may be accessed via a user-friendly interface created using the principles of Graphical User Interface design.

Table 1 provides a comparison of WIOMS with similar tools. It shows that WIOMS is most complete and provides users with more combined features. WIOMS answers user concerns while ensuring ease of use through an interface design that incorporate familiar elements.

Feature	WIOMS	Navi-Guidor [MW99]	MEMOS [AHY99]	Power-Bookmarks [BEA99]	WWW Resource Discovery Tool [LEA95]	PHT [Gra99]
WWW Navigation	X	X		X		X
Bookmarking	X			X		
History	X		X			X
Model History after User Logic			X			X
Modify Document Description	X					
Priority/Category Assignment	X					
Relevance Determination (User)	X					
Relevance Determination (Automatic)		X			X	
Resource Relocation	X					
Document Data Extraction	X			X		
Familiar GUI Features	X			X	X	X
Keyword Search	X			X	X	
WWW Search	X			X		
Information Sharing	X	X		X	X	

Table 1. WIOMS vs. Similar Tools.

5.3 Future Work

Future work may include adding an automatic bookmarking feature to monitor user navigation patterns and add entries to the category/priority lists based on visit frequency. This thesis implements the WIOMS on top of the Netscape Navigator browser; another implementation could extend WIOMS capabilities to others.

REFERENCES

- [AEA96] Andrews, Keith, et al, Visualizing the Internet: Putting the User in the Driver's Seat, in: CHI '96 Conference Proceedings on Human Factors in Computing Systems, ACM, 1996, pp. 163.
- [AHY99] Archer, Norm, Milena Head, Yufei Yuan, World Wide Web Information Support Through User Histories, in: Toronto '99 8th International World Wide Web Conference Poster Proceedings, Foretec Seminars, Inc., 1999, pp. 58-59.
- [BEA99] Bufi, Corey, et al, PowerBookmarks: A System for Personalizable Web Information Organization, Sharing, and Management, in: Proceedings of the International Conference on Management of Data, ACM SIGMOD, 1999, pp. 565-567.
- [Bro98] Broadhead, Glenn J., Using HTML frames for Institutional Websites, in: Proceedings on the Sixteenth Annual International Conference on Computer Documentation, ACM, 1998, pp. 278-285.
- [Cap98] Capron, H. L., Computers: Tools for an Information Age, 5th ed., Addison-Wesley Longman Publishing Company, Inc., 1998.
- [CLV99] Chang, Edward, Li, Wen-Syan and Quoc Vu, On Constructing Personalized Navigation Trees for Web Documents, in: Toronto '99 8th International World Wide Web Conference Poster Proceedings, Foretec Seminars, Inc., 1999, pp. 94-95.
- [DEA92] De Baar, Dennis J. M. J., Coupling Application Design and User Interface Design, in: CHI '92 Conference Proceedings on Human Factors in Computing Systems, ACM, 1992, pp.657-660.
- [DOL90] Don, Abbe, Oren, Tim and Brenda Laurel, Issues in Multimedia Interface Design: Media Integration and Interface Agents, in: CHI '90 Conference Proceedings on Human Factors in Computing Systems, ACM, 1990, pp. 133-139.
- [Eas91] Eastwood, David S., Programming for Events, in: Proceedings of the International Conference on APL '91, ACM, 1991, pp. 141-149.

- [FF93] Frank, Martin R. and James D. Foley, Model-Based User Interface Design by Example and by Answering Questions, in: CHI '93 Conference Proceedings on Human Factors in Computing Systems, ACM, 1993, pp. 161.
- [FGR96] Forsythe, Chris, Grose, Eric M. and Julie Ratner, Characterization and Assessment of HTML Style Guides, in: CHI '96 Conference Proceedings on Human Computer Factors in Computing Systems: Common Ground, ACM, 1996, pp. 115.
- [FS98] Frentzen, Jeff and Henry Sobotka, JavaScript Annotated Archives, Osborne/McGraw-Hill, 1998.
- [Gal97] Galitz, Wilbert O., Essential Guide to User Interface Design, John Wiley and Sons, 1997.
- [GT98] Giller, Verena and Manfred Tscheligi, Java Based User Interface Design and Development, in: CHI '98 Proceedings on Human Factors in Computing Systems, ACM, 1998, pp. 115-116.
- [Goo98] Goodman, Danny, JavaScript Bible, IDG Books Worldwide, Inc., 1998.
- [Gra99] Grayson, Ralph, The Design and Implementation of a World Wide Web Navigation History Tool, 1999.
- [GW99] Govindarajan, Jayesh and Matthew Ward, GeoViser: GeoSpatial Clustering and Visualization of Search Engine Results, in: Toronto '99 8th International World Wide Web Conference Poster Proceedings, 1999, pp. 79-83.
- [HR96] Hankin, Chris and Hanne RiisNielsen, ACM Computing Surveys, volume 28, number 2, 1996, pp. 293-294.
- [IP96] Instone, Keith and Steve Pemberton, HCI Issues of the World Wide Web, in: CHI '96 Proceedings on Human Factors in Computing Systems, ACM, 1996, pp. 423.
- [KP96] Kehoe, Colleen M. and James E. Pitkow, Emerging Trends in the WWW User Population, Communications of the ACM, June 1996, Volume 39 Number 6, pp. 106-108.
- [KP92] Kitajima, Muneo, Peter G. Polson, A Computational Model of Skilled Use of a Graphical User Interface, in: CHI '92 Conference Proceedings on Human Factors in Computing Systems, ACM, 1992, pp.241-249.
- [LEA95] Lam, Savio L. Y., et al., A World Wide Web Resource Discovery System, 1995.

- [MW99] Matias, Yossi, Gadi Wolfman, NaviGuiding Through the Many Faces of the Web, in: Toronto '99 8th International World Wide Web Conference Poster Proceedings, Foretec Seminars, Inc., 1999, pp.64-65.
- [Nat98] Nation, David A., WebTOC: a Tool to Visualize and Quantify Web Sites Using a Hierarchical Table of Contents Browser, in: CHI '98 Conference Proceedings on Human Factors in Computing Systems, ACM, 1998, pp.185-186.
- [Pir98] Pirolli, Peter, Exploring Browser Design Trade-Offs Using a Dynamical Model of Optimal Information Foraging, in: CHI '98 Conference Proceedings on Human Factors in Computing Systems, ACM, 1998, pp. 33-40.
- [SS97] Sachs, David and Henry Stair, The 7 Keys to Effective Web Sites, Prentice Hall, 1997.
- [WEA97] Wagoner, Richard, et al., JavaScript Unleashed, 2nd ed., Sams.net, 1997.
- [Way97] Wayner, Peter. Java and JavaScript Programming, Academic Press, Inc., 1997.
- [Wit96] Wittenburg, Kent, The WWW Information Glut: Implications for Next-Generation HCI Technologies, ACM Workshop on Strategic Direction in Computing Research, 1996.

APPENDIX I

IMPLEMENTATION SOURCE CODE

Frame Set

```
<HTML>
<HEAD>
<TITLE>Web Information Organization and Management System (WIOMS)</TITLE>
</HEAD>
<FRAMESET ROWS="50%,50%">
  <FRAME NAME="frame1" SRC="WIOMS_Driver.html">
  <FRAME NAME="frame2" SRC="WIOMS_Viewer.html">
</FRAMESET>
</HTML>
```

Driver_Frame

```
<HTML>
<HEAD>
<TITLE>PHNT - Driver Frame</TITLE>
<SCRIPT LANGUAGE="JavaScript">

var Description_array1 = new Array()
var Category_array1 = new Array()
var URL_array1 = new Array()

var Description_array2 = new Array()
var Category_array2 = new Array()
var URL_array2 = new Array()

var Description_array3 = new Array()
var Category_array3 = new Array()
var URL_array3 = new Array()

var Description_array4 = new Array()
```

```
var Category_array4 = new Array()
var URL_array4 = new Array()
```

```
var Description_array5 = new Array()
var Category_array5 = new Array()
var URL_array5 = new Array()
```

```
var index1 = -1
var index2 = -1
var index3 = -1
var index4 = -1
var index5 = -1
```

```
var Category_names = new Array()
var beenCalled = new Array()
```

```
for (var j = 1; j <=5; j++) {
  beenCalled[j] = -1
} // end for
```

```
var cookieName
var retrieveCookieName
var category
var retrieveCookieCategory
var expiration
```

```
var URL_value = new String()
var Description_value = new String()
var Category_value = new String()
var Cookie_value = new String()
```

```
var newWindow
newWindow = window.open("", "newWin")
newWindow.blur()
```

```
function fillBottomFrame() {
netscape.security.PrivilegeManager.enablePrivilege("UniversalBrowserAccess")
```

```
  top.frames[1].document.open()
  var newURL = entry.value
  if (newURL != null && newURL != "") {
    top.frames[1].location.href = newURL
  }
} // end function fillBottomFrame
```

```
function fillNewWindow() {
```

```

netscape.security.PrivilegeManager.enablePrivilege("UniversalBrowserAccess")
var newURL = document.forms[0].getURL.value
var temp = newURL.substring(0,7)

if (temp != "http://") {
    newURL = "http://" + document.forms[0].getURL.value
} // end if

newWindow.location.href = newURL
} // end function fillNewWindow

function showDescription() {
netscape.security.PrivilegeManager.enablePrivilege("UniversalBrowserAccess")
    document.forms[0].currDescription.value = newWindow.document.title
    document.forms[0].currURL.value = newWindow.location.href
} // end function showTitle

function storeEntry() {
netscape.security.PrivilegeManager.enablePrivilege("UniversalBrowserAccess")
    var newDescription = document.forms[0].currDescription.value
    var newCategory = document.forms[0].store.selectedIndex
    var newURL = document.forms[0].currURL.value
    var newOptionName
    var newPrintOptionName

document.forms[0].store[0].selected = "1"

if (beenCalled[newCategory] == -1) {
    if (newCategory == 1) {
        Category_names[newCategory] = prompt("Choose a Category/Priority name.",
            "Category/Priority - 1")
        if (Category_names[newCategory] != null) {
            newOptionName = new Option(Category_names[newCategory] + " - 1", 1)
            newPrintOptionName = new Option(Category_names[newCategory] + " - 1", 1)
            document.forms[0].store.options[1] = newOptionName
            document.forms[0].print.options[1] = newPrintOptionName
            beenCalled[newCategory] += 1
        } // end if
    } // end if
    if (newCategory == 2) {
        Category_names[newCategory] = prompt("Choose a Category/Priority name.",
            "Category/Priority - 2")
        if (Category_names[newCategory] != null) {
            newOptionName = new Option(Category_names[newCategory] + " - 2", 2)
            newPrintOptionName = new Option(Category_names[newCategory] + " - 2", 2)
            document.forms[0].store.options[2] = newOptionName
        } // end if
    } // end if
} // end function storeEntry

```



```

        document.forms[0].print.options[2] = newPrintOptionName
        beenCalled[newCategory] += 1
    } // end if
} // end if
if (newCategory == 3) {
    Category_names[newCategory] = prompt("Choose a Category/Priority name.",
        "Category/Priority - 3")
    if (Category_names[newCategory] != null) {
        newOptionName = new Option(Category_names[newCategory] + " - 3", 3)
        newPrintOptionName = new Option(Category_names[newCategory] + " - 3", 3)
        document.forms[0].store.options[3] = newOptionName
        document.forms[0].print.options[3] = newPrintOptionName
        beenCalled[newCategory] += 1
    } // end if
} // end if
if (newCategory == 4) {
    Category_names[newCategory] = prompt("Choose a Category/Priority name.",
        "Category/Priority - 4")
    if (Category_names[newCategory] != null) {
        newOptionName = new Option(Category_names[newCategory] + " - 4", 4)
        newPrintOptionName = new Option(Category_names[newCategory] + " - 4", 4)
        document.forms[0].store.options[4] = newOptionName
        document.forms[0].print.options[4] = newPrintOptionName
        beenCalled[newCategory] += 1
    } // end if
} // end if
if (newCategory == 5) {
    Category_names[newCategory] = prompt("Choose a Category/Priority name.",
        "Category/Priority - 5")
    if (Category_names[newCategory] != null) {
        newOptionName = new Option(Category_names[newCategory] + " - 5", 5)
        newPrintOptionName = new Option(Category_names[newCategory] + " - 5", 5)
        document.forms[0].store.options[5] = newOptionName
        document.forms[0].print.options[5] = newPrintOptionName
        beenCalled[newCategory] += 1
    } // end if
} // end if
} // end if beenCalled

if (newCategory == 1) {
    ++index1
    Description_array1[index1] = newDescription
    Category_array1[index1] = newCategory
    URL_array1[index1] = newURL
}
if (newCategory == 2) {

```

```

    ++index2
    Description_array2[index2] = newDescription
    Category_array2[index2] = newCategory
    URL_array2[index2] = newURL
  }
  if (newCategory == 3) {
    ++index3
    Description_array3[index3] = newDescription
    Category_array3[index3] = newCategory
    URL_array3[index3] = newURL
  }
  if (newCategory == 4) {
    ++index4
    Description_array4[index4] = newDescription
    Category_array4[index4] = newCategory
    URL_array4[index4] = newURL
  }
  if (newCategory == 5) {
    ++index5
    Description_array5[index5] = newDescription
    Category_array5[index5] = newCategory
    URL_array5[index5] = newURL
  }
} //end function storeEntry

function printCategory() {
netscape.security.PrivilegeManager.enablePrivilege("UniversalBrowserAccess")

var newIndex = document.forms[0].print.selectedIndex
document.forms[0].print[0].selected = "1"

top.frame2.document.clear()
top.frame2.document.write("<HTML><HEAD><TITLE>List Contents</TITLE>")
top.frame2.document.write("</HEAD><BODY><FORM>")

if (Category_names[newIndex] != null)
  top.frame2.document.write("<CENTER><H1>Viewing Category/Priority <FONT
  COLOR = #FF0000>(" + newIndex + ") : " + Category_names[newIndex] +
  "</FONT></H1></CENTER><BR>")
else
  top.frame2.document.write("<CENTER><H1>Viewing Category/Priority <FONT
  COLOR = #0000FF>(" + newIndex + ") : " + "List Name Undefined" +
  "</FONT></H1></CENTER><BR>")

```

```

if (newIndex == 1) {
  for (var i = 0; i < URL_array1.length; i++) { // max number elements = 25
    top.frame2.document.write("URL: " + URL_array1[i] + " ")
    top.frame2.document.write(" <INPUT TYPE=\\"button\\" VALUE=\\"GO\\"
      onClick=\\"top.frame1.fillNewWindow2(\" + i + \",\" + newIndex + \")\\">")
    top.frame2.document.write("<BR>Description: " + Description_array1[i] + " ")
    top.frame2.document.write("<BR><BR>")
  } // end for
} //end if
if (newIndex == 2) {
  for (var i = 0; i < URL_array2.length; i++) { // max number elements = 25
    top.frame2.document.write("URL: " + URL_array2[i] + " ")
    top.frame2.document.write(" <INPUT TYPE=\\"button\\" VALUE=\\"GO\\"
      onClick=\\"top.frame1.fillNewWindow2(\" + i + \",\" + newIndex + \")\\">")
    top.frame2.document.write("<BR>Description: " + Description_array2[i] + " ")
    top.frame2.document.write("<BR><BR>")
  } // end for
} //end if
if (newIndex == 3) {
  for (var i = 0; i < URL_array3.length; i++) { // max number elements = 25
    top.frame2.document.write("URL: " + URL_array3[i] + " ")
    top.frame2.document.write(" <INPUT TYPE=\\"button\\" VALUE=\\"GO\\"
      onClick=\\"top.frame1.fillNewWindow2(\" + i + \",\" + newIndex + \")\\">")
    top.frame2.document.write("<BR>Description: " + Description_array3[i] + " ")
    top.frame2.document.write("<BR><BR>")
  } // end for
} //end if
if (newIndex == 4) {
  for (var i = 0; i < URL_array4.length; i++) { // max number elements = 25
    top.frame2.document.write("URL: " + URL_array4[i] + " ")
    top.frame2.document.write(" <INPUT TYPE=\\"button\\" VALUE=\\"GO\\"
      onClick=\\"top.frame1.fillNewWindow2(\" + i + \",\" + newIndex + \")\\">")
    top.frame2.document.write("<BR>Description: " + Description_array4[i] + " ")
    top.frame2.document.write("<BR><BR>")
  } // end for
} //end if
if (newIndex == 5) {
  for (var i = 0; i <= URL_array5.length; i++) { // max number elements = 25
    top.frame2.document.write("URL: " + URL_array5[i] + " ")
    top.frame2.document.write(" <INPUT TYPE=\\"button\\" VALUE=\\"GO\\"
      onClick=\\"top.frame1.fillNewWindow2(\" + i + \",\" + newIndex + \")\\">")
    top.frame2.document.write("<BR>Description: " + Description_array5[i] + " ")
    top.frame2.document.write("<BR><BR>")
  } // end for
} //end if

```

```

top.frame2.document.write("</FORM></BODY></HTML>")
top.frame2.document.close()
} // end function printArray

function setCookieValues() {
    var newCategory = category
    var newCookieName = cookieName
    var newExpiration = expiration

    var oneDay = expiration * 24 * 60 * 60 * 1000

    var expDate = new Date();

    expDate.setTime (expDate.getTime() + oneDay)

    var path1 = "/phnt"
    var path2 = "/phnt/tool"
    var path3 = "/"

    if (newCategory == 1) {
        URL_value = URL_array1.join()
        Description_value = Description_array1.join()
        Category_value = Category_array1.join()
    } // end if
    if (newCategory == 2) {
        URL_value = URL_array2.join()
        Description_value = Description_array2.join()
        Category_value = Category_array2.join()
    } // end if
    if (newCategory == 3) {
        URL_value = URL_array3.join()
        Description_value = Description_array3.join()
        Category_value = Category_array3.join()
    } //end if
    if (newCategory == 4) {
        URL_value = URL_array4.join()
        Description_value = Description_array4.join()
        Category_value = Category_array4.join()
    } // end if
    if (newCategory == 5) {
        URL_value = URL_array5.join()
        Description_value = Description_array5.join()
    }
}

```

```

    Category_value = Category_array5.join()
} // end if

Cookie_value = URL_value + "" + Description_value + "" + Category_value
setCookie(newCookieName, Cookie_value, expDate, null, null, false)
} // end function setCookieValues

function setCookie ( name, value, expires, path, domain, secure) {
    var expString = ((expires == null) ? "" : ("; expires=" + expires.toGMTString()))
    var pathString = ((path == null) ? "" : ("; path=" + path))
    var domainString = ((domain == null) ? "" : ("; domain=" + domain))
    var secureString = ((secure == true) ? "; secure" : "")
    document.cookie = name + "=" + escape (value) + expString + pathString +
        domainString + secureString
} // end function setCookie

function getSetCookieValues() {
    cookieName = prompt("Enter List Name")
    if (cookieName != null) {
        category = prompt("Enter Category/Priority Number of List to be Saved")
        if (category != null) {
            expiration = prompt("Choose List Expiration Time: " + "\r" +
                "(1) = 1 day   (2) = 3 days   (3) = 1 week   (4) = 2 weeks")
            if (expiration != null) {
                setCookieValues()
            } // end if 3
        } // end if 2
    } // end if 1
} // end functiongetSetCookieValues

function getRetrieveCookieValues() { // need the name and new category
    retrieveCookieName = prompt("Enter Name of List to Retrieve.")
    if ((retrieveCookieName != null) && (retrieveCookieName != "undefined")) {
        retrieveCookieCategory = prompt("Enter Category/Priority Number for Retrieved
List")

        if (retrieveCookieCategory != null) {
            Cookie_value = getCookie()

            if (Cookie_value != null) {
                var temp1
                var temp2
                temp1 = Cookie_value.indexOf("")
                URL_value = Cookie_value.substring(0, temp1-1)

                temp1 += 1

```

```

temp2 = Cookie_value.lastIndexOf("")
Description_value = Cookie_value.substring(temp1, temp2)

temp2 += 1
temp1 = Cookie_value.length
Category_value = Cookie_value.substring(temp2, temp1)
} // end if
else {
    alert("There is No List Named : " + retrieveCookieName.toUpperCase())
} // end else
} // end if 2
} // end if 3
else {
    alert("You Must Enter Name of List to Retrieve.")
} // end else

// Next the retrieved cookies must be split back into arrays
// corresponding to the category number the user has specified
if (retrieveCookieCategory == 1) {
    URL_array1 = URL_value.split(",")
    Description_array1 = Description_value.split(",")
    Category_array1 = Category_value.split(",")
    index1 = URL_array1.length - 1
} // end if
if (retrieveCookieCategory == 2) {
    URL_array2 = URL_value.split(",")
    Description_array2 = Description_value.split(",")
    Category_array2 = Category_value.split(",")
    index2 = URL_array2.length - 1
} // end if
if (retrieveCookieCategory == 3) {
    URL_array3 = URL_value.split(",")
    Description_array3 = Description_value.split(",")
    Category_array3 = Category_value.split(",")
    index3 = URL_array3.length - 1
} // end if
if (retrieveCookieCategory == 4) {
    URL_array4 = URL_value.split(",")
    Description_array4 = Description_value.split(",")
    Category_array4 = Category_value.split(",")
    index4 = URL_array4.length - 1 // allows new entries to be added after cookie is
retrieved
} // end if
if (retrieveCookieCategory == 5) {
    URL_array5 = URL_value.split(",")
    Description_array5 = Description_value.split(",")

```

```

    Category_array5 = Category_value.split(",")
    index4 = URL_array5.length - 1
  } // end if
} // end function getRetrieveCookieValues

function getCookie () {
  var result = null
  var myCookie = " " + document.cookie + ";"
  var searchName = " " + retrieveCookieName + "="
  var startOfCookie = myCookie.indexOf(searchName)
  var endOfCookie
  if (startOfCookie != -1) {
    startOfCookie += searchName.length
    endOfCookie = myCookie.indexOf(";", startOfCookie)
    result = unescape(myCookie.substring(startOfCookie, endOfCookie))
  } // end if
  return result
} // end function getCookie

function clearCookie() {

  var clearCookieName = prompt("Enter the name of the cookie to delete > ")

  var threeDays = 3 * 24 * 60 * 60 * 1000
  var expDate = new Date()
  expDate.setTime(expDate.getTime() - threeDays)
  document.cookie = clearCookieName + "=ImOutOfHere; expires=" +
    expDate.toGMTString()

} // end function clearCookie

function fillNewWindow2(index, array_num) {
  // this function allows the new window to be filled with the URL returned by a function
  // such as search
  if (array_num == 1 ) {
    newWindow.location.href = URL_array1[index]
  } // end if
  if (array_num == 2) {
    newWindow.location.href = URL_array2[index]
  } // end if
  if (array_num == 3) {
    newWindow.location.href = URL_array3[index]
  } // end if
  if (array_num == 4) {
    newWindow.location.href = URL_array4[index]
  } // end if
}

```

```

if (array_num == 5) {
    newWindow.location.href = URL_array5[index]
} // end if
} // end fill NewWindow2

function searchCategory () {
    var searchKeyword = prompt("Enter Keyword to Find.")
    var searchNumber = prompt("Enter Category/Priority Number of List to Search.")

    if ((searchKeyword == null) || (searchKeyword == "undefined"))
        alert("You Must Enter 1 or More Keywords.")
    else if ((searchNumber == null) || (searchNumber == "undefined"))
        alert("You Must Enter Category/Priority Number.")
    else {
        var newArray = new Array()
        var newIndex = 0
        var temp1 = searchKeyword.toUpperCase() // toUpperCase
        var temp2

        if (searchNumber == 1) {
            for (var i = 0; i < Description_array1.length; i++) {
                temp2 = Description_array1[i].toUpperCase()

                if (temp2.indexOf(temp1) != -1) { // if the keyword is found
                    newArray[newIndex] = i // find the position in the original arrays
                    newIndex += 1
                } // end if
            } // end for
            if (newIndex == 0) {
                var phrase = "Keyword : " + searchKeyword + " NOT FOUND"
                alert(phrase)
            } else {
                top.frame2.document.clear()
                top.frame2.document.write("<HTML><HEAD><TITLE>Search
                Results</TITLE>")
                top.frame2.document.write("</HEAD><BODY><FORM>")
                top.frame2.document.write("<CENTER><H1>Search Results : " +
                searchKeyword + "</H1></CENTER>")
                for (var j = 0; j < newArray.length; j++) {
                    var tempIndex = newArray[j]

                    top.frame2.document.write("URL : " + URL_array1[tempIndex])
                    top.frame2.document.write(" <INPUT TYPE="button" VALUE="GO"
                    onClick="top.frame1.fillNewWindow2(" + tempIndex + "," + searchNumber
                    + ")">")
                }
            }
        }
    }
}

```



```

        top.frame2.document.write("<BR>Description : " +
            Description_array1[tempIndex] + "<BR>")
    } // end for
} // end if-else

top.frame2.document.write("</FORM></BODY></HTML>")
top.frame2.document.close()

} // end if

if (searchNumber == 2) {
    for (var i = 0; i < Description_array2.length; i++) {
        temp2 = Description_array2[i].toUpperCase

        if (temp2.indexOf(temp1) != -1) { // if the keyword is found
            newArray[newIndex] = i // find the position in the original arrays
            newIndex += 1
        } // end if
    } // end for
    if (newIndex == 0) {
        var phrase = "Keyword : " + searchKeyword + " NOT FOUND"
        alert(phrase)
    } else {
        top.frame2.document.clear()
        top.frame2.document.write("<HTML><HEAD><TITLE>Search
            Results</TITLE>")
        top.frame2.document.write("</HEAD><BODY><FORM>")
        top.frame2.document.write("<CENTER><H1>Search Results : " +
            searchKeyword + "</H1></CENTER>")
        for (var j = 0; j < newArray.length; j++) {
            var tempIndex = newArray[j]

            top.frame2.document.write("URL : " + URL_array2[tempIndex])
            top.frame2.document.write(" <INPUT TYPE=\"button\" VALUE=\"GO\"
                onClick=\"top.frame1.fillNewWindow2(" + tempIndex + "," + searchNumber
                + ")\">")
            top.frame2.document.write("<BR>Description : " +
                Description_array2[tempIndex] + "<BR>")
        } // end for
    } // end if-else

    top.frame2.document.write("</FORM></BODY></HTML>")
    top.frame2.document.close()

} // end if searchNumber

```

```

if (searchNumber == 3) {
  for (var i = 0; i < Description_array3.length; i++) {
    temp2 = Description_array3[i].toUpperCase

    if (temp2.indexOf(temp1) != -1) { // if the keyword is found
      newArray[newIndex] = i // find the position in the original arrays
      newIndex += 1
    } // end if
  } // end for
  if (newIndex == 0) {
    var phrase = "Keyword : " + searchKeyword + " NOT FOUND"
    alert(phrase)
  } else {
    top.frame2.document.clear()
    top.frame2.document.write("<HTML><HEAD><TITLE>Search
      Results</TITLE>")
    top.frame2.document.write("</HEAD><BODY><FORM>")
    top.frame2.document.write("<CENTER><H1>Search Results : " +
      searchKeyword + "</H1></CENTER>")
    for (var j = 0; j < newArray.length; j++) {
      var tempIndex = newArray[j]

      top.frame2.document.write("URL : " + URL_array3[tempIndex])
      top.frame2.document.write(" <INPUT TYPE="button" VALUE="GO"
        onClick="top.frame1.fillNewWindow2(" + tempIndex + "," + searchNumber
          + ")\>")
      top.frame2.document.write("<BR>Description : " +
        Description_array3[tempIndex] + "<BR>")
    } // end for
  } // end if-else

  top.frame2.document.write("</FORM></BODY></HTML>")
  top.frame2.document.close()

} // end if searchNumber

if (searchNumber == 4) {
  for (var i = 0; i < Description_array4.length; i++) {
    temp2 = Description_array4[i].toUpperCase()

    if (temp2.indexOf(temp1) != -1) { // if the keyword is found
      newArray[newIndex] = i // find the position in the original arrays
      newIndex += 1
    } // end if
  } // end for
}

```

```

if (newIndex == 0) {
    var phrase = "Keyword : " + searchKeyword + " NOT FOUND"
    alert(phrase)
} else {
    top.frame2.document.clear()
    top.frame2.document.write("<HTML><HEAD><TITLE>Search
    Results</TITLE>")
    top.frame2.document.write("</HEAD><BODY><FORM>")
    top.frame2.document.write("<CENTER><H1>Search Results : " +
    searchKeyword + "</H1></CENTER>")
    for (var j = 0; j < newArray.length; j++) {
        var tempIndex = newArray[j]

        top.frame2.document.write("URL : " + URL_array4[tempIndex])
        top.frame2.document.write(" <INPUT TYPE=\"button\" VALUE=\"GO\"
        onClick=\"top.frame1.fillNewWindow2(\" + tempIndex + \",\" + searchNumber
        + \")\">")
        top.frame2.document.write("<BR>Description : " +
        Description_array4[tempIndex] + "<BR>")
    } // end for
} // end if-else

top.frame2.document.write("</FORM></BODY></HTML>")
top.frame2.document.close()

} // end if searchNumber

if (searchNumber == 5) {
    for (var i = 0; i < Description_array5.length; i++) {
        temp2 = Description_array5[i].toUpperCase()

        if (temp2.indexOf(temp1) != -1) { // if the keyword is found
            newArray[newIndex] = i // find the position in the original arrays
            newIndex += 1
        } // end if
    } // end for
} // end if
if (newIndex == 0) {
    var phrase = "Keyword : " + searchKeyword + " NOT FOUND"
    alert(phrase)
} else {
    top.frame2.document.clear()
    top.frame2.document.write("<HTML><HEAD><TITLE>Search
    Results</TITLE>")
    top.frame2.document.write("</HEAD><BODY><FORM>")

```

```

top.frame2.document.write("<CENTER><H1>Search Results : " +
    searchKeyword + "</H1></CENTER>")
for (var j = 0; j < newArray.length; j++) {
    var tempIndex = newArray[j]

    top.frame2.document.write("URL : " + URL_array5[tempIndex])
    top.frame2.document.write(" <INPUT TYPE=\"button\" VALUE=\"GO\"
        onClick=\"top.frame1.fillNewWindow2(\" + tempIndex + \",\" + searchNumber
        + ")\">")
    top.frame2.document.write("<BR>Description : " +
        Description_array5[tempIndex] + "<BR>")
    } // end for
} // end if-else

top.frame2.document.write("</FORM></BODY></HTML>")
top.frame2.document.close()

} // end if searchNumber

} // end if searchKeyword != null
} // end function searchCookie

function helpMenu() {
    var helpWindow
    helpWindow = window.open("Help.html", "helpWin", "toolbar=0, width=700,
        height=400, resizable=0")
} // end function help

function webSearch() {
    netscape.security.PrivilegeManager.enablePrivilege("UniversalBrowserAccess")
    top.frames[1].document.open()
    var newURL = "WebSearch.html"
    if (newURL != null && newURL != "") {
        top.frames[1].location.href = newURL
    }
} // end function fillBottomFrame

function showHistory() {
    netscape.security.PrivilegeManager.enablePrivilege("UniversalBrowserAccess")
    var histInfo = ""
    for (var histCount=0; histCount < newWindow.history.length; histCount++) {
        histInfo += newWindow.history[histCount] + "\r"
    }
    if (newWindow.history.length != 0)
        alert(histInfo)
}

```

```

else
    alert("History is empty.")
} // end function showHistory

</SCRIPT>
</HEAD>
<BODY bgcolor="#FFFFFF">

<FORM method="POST">

    <IMG ALIGN=RIGHT SRC="main_pic.gif">
    <TABLE BORDER=0>
        <TR>
            <TD ALIGN=RIGHT>Location:</TD>
            <TD>
                <INPUT TYPE="text" NAME="getURL" SIZE=60 MAXLENGTH=60
                    onChange="fillNewWindow()">
                <INPUT TYPE="button" VALUE="GO" onClick="fillNewWindow()">
            </TD>
        </TR>

        <TR>
            <td></td>
            <td>
                <INPUT TYPE="button" NAME="viewDescription" VALUE="Show Entry"
                    onClick="showDescription()">
                <SELECT NAME="store" SIZE=1 onChange="storeEntry()">
                    <OPTION>(Store Entry)
                    <OPTION VALUE=1>Category/Priority - 1</OPTION>
                    <OPTION VALUE=2>Category/Priority - 2</OPTION>
                    <OPTION VALUE=3>Category/Priority - 3</OPTION>
                    <OPTION VALUE=4>Category/Priority - 4</OPTION>
                    <OPTION VALUE=5>Category/Priority - 5</OPTION>
                </SELECT>
                <SELECT NAME="print" SIZE=1 onChange="printCategory()">
                    <OPTION> (Show List)
                    <OPTION VALUE=1>Print Category/Priority - 1</OPTION>
                    <OPTION VALUE=2>Print Category/Priority - 2</OPTION>
                    <OPTION VALUE=3>Print Category/Priority - 3</OPTION>
                    <OPTION VALUE=4>Print Category/Priority - 4</OPTION>
                    <OPTION VALUE=5>Print Category/Priority - 5</OPTION>
                </SELECT>
            </td>
        </TR>
    </TABLE>

```

```

<TR>
  <TD ALIGN=RIGHT>Description :</TD>
  <TD><TEXTAREA NAME="currDescription" ROWS=2 COLS=36
    WRAP="soft"></TEXTAREA></TD>
</TR>

<TR>
  <TD ALIGN=RIGHT>URL (Address) :</TD>
  <TD><TEXTAREA NAME="currURL" ROWS=1 COLS=59
    WRAP="soft"></TEXTAREA></TD>
</TR>

</table>
<hr>
<table border=0>
  <TR>
    <TD></TD>
    <TD>
      <INPUT TYPE="button" NAME="setCookie" value="Save List"
        onClick="getSetCookieValues()">
      <INPUT TYPE="button" NAME="retrieveCookie" value="Retrieve List"
        onClick="getRetrieveCookieValues()">
      <INPUT TYPE="button" NAME="deleteCategory" VALUE="Delete List"
        onClick="clearCookie()">
      <INPUT TYPE="button" NAME="HELP" VALUE=" HELP "
        onCLick="helpMenu()">
      <INPUT TYPE="button" NAME="search" VALUE="Keyword Search"
        onClick="searchCategory()">
      <input type = "button" value="WWW Search" onClick="webSearch()">
      <input type = "button" value="Show History" onClick="showHistory()">
    </td>
  </TR>

</TABLE>
</FORM>

</BODY>
</HTML>

<HTML>
<HEAD>

```

Viewer Frame

```

<TITLE>WIOMS - Viewer</TITLE>
</HEAD>

<BODY BGCOLOR="#FFFFFF">
<P>
  <CENTER><H3>Welcome to the Web Information Organization and Management
    System (WIOMS)</H3></CENTER>
</P>
<BR><BR><BR><BR><BR><BR>
<P align=center>
  <FONT><A href="mailto:tynan_osu@email.okstate.edu">Creator:&nbsp; Tynan
    Grayson</A></FONT>
</P>
</BODY>
</HTML>

```

Web Search

```

<HTML>
<HEAD>
<TITLE>Web Search</TITLE>
</HEAD>
<BODY>

<SCRIPT>
  var MAX_ENGINES = 30;
  var searchString_STRING = "hunting+the+searchString";

function makeArray(n) {
  for (var i = 1; i <= n; i++) {
    this[i] = 0;
  }
  this.maxlen = n;
  this.len = 0;
  return this;
} // end function makeArray

var searchEngines = new makeArray(MAX_ENGINES);

function findSubstring(stringToFind, searchLocation) {
  var i, stringToFindn = stringToFind.length, searchLocationLen =
    searchLocation.length;
  for (i=0; i<=searchLocationLen-stringToFindn; i++) {
    if (stringToFind == searchLocation.substring(i,i+stringToFindn))

```

```

    return i;
  }
  return false;
} // end function findSubstring

function engine(name, opts, home, search) {
  var searchString = findSubstring(searchString_STRING, search);
  this.name = name;
  this.opts = opts;
  this.home = home;
  this.pre_searchString = search.substring(0,searchString);
  this.post_searchString= search.substring(searchString+searchString_STRING.length,
  search.length);
} // end function engine

function add(name, opts, home, search) {
  searchEngines.len++;
  if (searchEngines.len <= searchEngines.maxlen) {
    searchEngines[searchEngines.len] = new engine(name, opts, home, search)
  }
  else {
    alert("Better increase MAX_ENGINES: " + searchEngines.len + ">" +
    searchEngines.maxlen)
  }
} // end function add

// add search engines
add("the Web: AltaVista", "SELECTED", "http://altavista.digital.com/", "http://
altavista.digital.com/cgi-bin/query?pg=q&what=web&fmt=d&q=hunting+the+
searchString" );
add("the Web: Yahoo!", "", "http://www.yahoo.com/", "http://search.yahoo.com/bin/
search?p=hunting+the+searchString" );

function handleForm(form) {
  form.submit(); // allows the use of <enter> to initiate the first search
  var i, oldq=form.query.value, newq="";
  for (i=0; i<oldq.length; i++) {
    var thischar = oldq.charAt(i);
    if (thischar != ' ')
      newq += thischar;
    else if (lastchar != ' ')
      newq += '+';
    lastchar = thischar;
  } // end for
  var searchEngine = searchEngines[1+form.service.selectedIndex];
  location.href = newq ? searchEngine.pre_searchString + newq +

```



```

    searchEngine.post_searchString : searchEngine.home;
} // end function handleForm

function displayForm() {
    document.writeln('<h1><Center>World Wide Web Search</center></h1><br>');
    document.writeln('<TABLE border=0><TR><TD>');
    document.writeln('<FORM OnSubmit="handleForm(this); return false">');
    document.writeln('Search : <SELECT name="service">');
    for (i=1; i <= searchEngines.len; i++) { □      document.writeln("<OPTION " +
        searchEngines[i].opts + "> " + searchEngines[i].name);
    } // end for
    document.writeln('</SELECT>');
    document.writeln(' <INPUT value="" size=45 name="query">');
    document.writeln('<input type=submit value=" GO">');
    document.writeln('</FORM>');
    document.writeln('</td></TR></TABLE>');
}

displayForm();
</SCRIPT>
<P>
</BODY>
</HTML>

```

Help Menu

```

<HTML>
<HEAD>
<TITLE>WIOMS - Help</TITLE>
<SCRIPT LANGUAGE="JavaScript">
//This script changes the graphic on mouse over and a graphic in
//another part of the screen
{
    alt0 = new Image();
    alt0.src = "help_intro2.gif";
    alt1 = new Image();
    alt1.src = "help_navigate.gif";
    alt2 = new Image();
    alt2.src = "help_show_mod.gif";
    alt3 = new Image();
    alt3.src = "help_stor_ent.gif";
    alt4 = new Image();
    alt4.src = "help_show_lis.gif";
    alt5 = new Image();

```

```

alt5.src = "help_save_lis.gif";
alt6 = new Image();
alt6.src = "help_retrv_lis.gif";
alt7 = new Image();
alt7.src = "help_del_lis.gif";
alt8 = new Image();
alt8.src = "help_key_sear.gif";
alt9 = new Image();
alt9.src = "help_www_sear.gif";

graphic1= new Image();
graphic1.src = "help_button_off.gif";
graphic1on = new Image();
graphic1on.src = "help_button_on.gif";
}

function imageChange(imageID,imageName,imageID2,imageName2) {
{
document.images[imageID].src = eval(imageName + ".src");
document.images[imageID2].src = eval(imageName2 + ".src");
}
}
}
</SCRIPT>

<CENTER>
<TABLE border="0" width="100%" height="100%">
<TR>
<TD align=left valign=top height="100%">

<A HREF="Help.html"
ONMOUSEOVER="imageChange('global','alt1','one','graphic1on')"
ONMOUSEOUT="imageChange('global','alt0','one','graphic1')">
<IMG SRC="help_button_off.gif" align="abscenter" BORDER="0"
NAME="one"></A>Navigate<BR>

<A HREF="Help.html"
ONMOUSEOVER="imageChange('global','alt2','two','graphic1on')"
ONMOUSEOUT="imageChange('global','alt0','two','graphic1')">
<IMG SRC="help_button_off.gif" align="abscenter" BORDER="0"
NAME="two"></A>Show/Modify Entry<br>

<A HREF="Help.html"
ONMOUSEOVER="imageChange('global','alt3','three','graphic1on')"
ONMOUSEOUT="imageChange('global','alt0','three','graphic1')">
<IMG SRC="help_button_off.gif" align="abscenter" BORDER="0"
NAME="three"></A>Store Entry<br>

```

```
<A HREF="Help.html"
ONMOUSEOVER="imageChange('global','alt4','four','graphic1on')"
ONMOUSEOUT="imageChange('global','alt0','four','graphic1')">
  <IMG SRC="help_button_off.gif" align="abscenter" BORDER="0"
NAME="four"></A>Show List<br>
```

```
<A HREF="Help.html"
ONMOUSEOVER="imageChange('global','alt5','five','graphic1on')"
ONMOUSEOUT="imageChange('global','alt0','five','graphic1')">
  <IMG SRC="help_button_off.gif" align="abscenter" BORDER="0"
NAME="five"></A>Save List<BR>
```

```
<A HREF="Help.html"
ONMOUSEOVER="imageChange('global','alt6','six','graphic1on')"
ONMOUSEOUT="imageChange('global','alt0','six','graphic1')">
  <IMG SRC="help_button_off.gif" align="abscenter" BORDER="0"
NAME="six"></A>Retrieve List<br>
```

```
<A HREF="Help.html"
ONMOUSEOVER="imageChange('global','alt7','seven','graphic1on')"
ONMOUSEOUT="imageChange('global','alt0','seven','graphic1')">
  <IMG SRC="help_button_off.gif" align="abscenter" BORDER="0"
NAME="seven"></A>Delete List<BR>
```

```
<A HREF="Help.html"
ONMOUSEOVER="imageChange('global','alt8','eight','graphic1on')"
ONMOUSEOUT="imageChange('global','alt0','eight','graphic1')">
  <IMG SRC="help_button_off.gif" align="abscenter" BORDER="0"
NAME="eight"></A>Keyword Search<br>
```

```
<A HREF="Help.html"
ONMOUSEOVER="imageChange('global','alt9','nine','graphic1on')"
ONMOUSEOUT="imageChange('global','alt0','nine','graphic1')">
  <IMG SRC="help_button_off.gif" align="abscenter" BORDER="0"
NAME="nine"></A>WWW Search<BR>
</TD>
```

```
<TD align=center valign=center>
  <IMG SRC="help_intro.gif" WIDTH="450" HEIGHT="350" NAME="global" >
</TD>
</TR>
</TABLE>
</CENTER>
```

```
</HEAD>
```

```
<BODY BGCOLOR=#FFFFFF>  
</BODY>  
</HTML>
```

APPENDIX II

GLOSSARY

Application Program	Computer programs that perform useful work not related to the computer itself.
Current Document	Refers to the document currently loaded in the New Window created by WIOMS.
Event	A result of user/GUI interaction.
HTML	A set of codes that can be inserted into text files to indicate special typefaces, inserted images, and links to other hypertext documents.
Internet	A cooperative message-forward system, linking computer networks all over the world.
Cookies	Persistent Client State HTTP Cookies are a general mechanism which server side connections (such as CGI scripts) can use to both store and retrieve information on the client side of the connection.
URL	A way of specifying the location of publicly available information on the Internet.
Web Indexer Robot	An autonomous World Wide Web browser that communicates with World Wide Web servers using Hypertext Transfer Protocol. It visits a given Web site, traverses hyperlinks in a breadth first manner, retrieves Web pages, extracts keywords and hyperlink data from the pages, and inserts the keywords and hyperlink data into an index [LEA95].

APPENDIX III

LIST OF ACRONYMS

HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
MEMOS	Memory Extender Mechanism for Online Searching
WIOMS	Web Information Organization and Retrieval System
URL	Uniform Resource Locator
WWW	World Wide Web

VITA

Tynan D Grayson

Candidate for the Degree of

Master of Science

Thesis: A WEB INFORMATION ORGANIZATION AND MANAGEMENT
SYSTEM

Major Field: Computer Science

Biographical:

Personal Data: Born in Oklahoma City, Oklahoma, on July 22, 1973, the daughter of Milton L., Sr. and Carolyn Anderson.

Education: Graduated from Capitol Hill High School, Oklahoma City, Oklahoma in May, 1991; received Bachelor of Science degree in Computer Science from Langston University, Langston, Oklahoma in May, 1996. Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in December, 1999.

Experience: Employed by Oklahoma State University, Department of Computer Science as both a graduate research assistant and a graduate teaching assistant; Employed by Oklahoma State University, Graduate College, as an instructor of World Wide Web development and ethics during the summers of 1998 and 1999; Oklahoma State University, Department of Computer Science, 1996 to present.

Professional Memberships: Association of Computing Machinery.