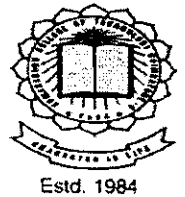




P-1801



EFFECTIVELY FINDING RELEVANT WEB PAGE LINKS FROM LINKAGE INFORMATION

A PROJECT REPORT

Submitted by

J.SHARAN	71203104044
S.P.SUREN	71203104052
R.VINOTH	71203104058

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

**KUMARAGURU COLLEGE OF TECHNOLOGY,
COIMBATORE**

ANNA UNIVERSITY:: CHENNAI 600 025

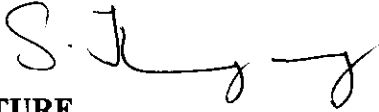
APRIL 2007



ANNA UNIVERSITY: CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this project report “**EFFECTIVELY FINDING RELEVANT WEB PAGE LINKS FROM LINKAGE INFORMATION**” is the bonafide work of “**J.SHARAN (71203104044), S.P.SUREN (71203104052), R.VINOTH (71203104058)**”, who carried out the project work under my supervision.



SIGNATURE

Dr.S.Thangasamy

HEAD OF THE DEPARTMENT

Dept. of Computer Science & Engg.,
Kumaraguru College of Technology,
Coimbatore-641006.



SIGNATURE

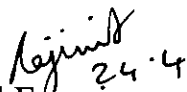
Mrs.S.Devaki

SUPERVISOR

Assistant Professor,

Dept. of Computer Science & Engg.,
Kumaraguru College of Technology,
Coimbatore-641006.

Submitted for Viva Voce Examination held on 24.04.07


Internal Examiner

External Examiner


DECLARATION

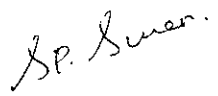
We hereby declare that the project entitled “**EFFECTIVELY FINDING RELEVANT WEB PAGE LINKS FROM LINKAGE INFORMATION**” is a record of the original work done by us and to the best of our knowledge.


The report is submitted in partial fulfillment of the requirements for the award for the Degree of Bachelor of Engineering in Computer Science and Engineering of Anna University, Chennai.

Place: Coimbatore.

Date: 23.04.07


(J. Sharan)


(S.P. Suren)


(R. Vinoth)

ACKNOWLEDGEMENT

We would like to extend our gratitude to the Principal of our college, **Dr. Joseph.V.Thanikal** for providing us the required resources to proceed with our project.

We would like to express our sincere thanks to **Dr.S.Thangasamy**, Head of the Department, Department of Computer Science and Engineering, Kumaraguru College of Technology, for his guidelines that motivated us to develop a good product.

We are grateful to **Mrs.P.Devaki**, Assistant Professor, Department of Computer Science and Engineering, Kumaraguru College of Technology for her encouragement and support at various levels of this project work. As our project guide, we also thank her for her inspiration and guidance throughout this project. We are very grateful to her for the help rendered to us in handling various tough spots during this project.

We also express our sincere thanks to **Ms.S.Rajini**, Senior Lecturer, Project Coordinator, Department of Computer Science and Engineering, Kumaraguru College of Technology for her valuable guidance and encouragement at every stage of this project.

Most of all, we thank our parents and friends for their blessings, help and support without which we would not be able to do anything.

ABSTRACT

ABSTRACT

The project deals with effectively finding the relevant web page links from linkage information using VB.Net when concerned with the INTERNET.

The project presents two hyperlink analysis-based algorithms to find relevant pages for a given Web page (URL). The algorithm takes advantage of linear algebra theories to reveal deeper relationships among the Web pages and to identify relevant pages more precisely and effectively. The experimental results show the feasibility and effectiveness of the algorithm. This algorithm could be used for various Web applications, such as enhancing Web search. The ideas and techniques in this work would be helpful to other Web-related researches.

The input is given as URL in which the source page of the URL is analyzed which displays information about the web directory information, domain listing, parent links, and child links. We can retrieve the information by clicking the link and the URL displays all the relevant links with the relevant information, which may be more than one lakh.

TABLE OF CONTENTS

TABLE OF CONTENTS

CHAPTERS	TITLE	PAGE NO.
	ACKNOWLEDGEMENT	iv
	ABSTRACT	v
1	INTRODUCTION	1
2	STUDY PROPOSAL	4
	2.1 Problem Definition	4
	2.2 Existing System	4
	2.3 Proposed System	5
3	SYSTEM PROPOSAL	7
	3.1 Software Specifications	7
	3.2 Hardware Specifications	7
	3.3 System Analysis	7
	3.3.1 Language Description	8
4	ALGORITHM DESCRIPTION	10
	4.1 Singular Value Decomposition	10
	4.2 Latent Linkage Information	11
5	SYSTEM DESIGN	16
	5.1 Architectural Design	16
	5.2 Process Design	17
	5.2.1 Use Case Diagram	17
	5.2.2 Sequence Diagram	18
	5.3 Form Design	19
	5.3.1 Components	20
	5.4. Form Layout	21

6	FEASIBILITY ASSESSMENT	22
	6.1 Technical Feasibility	22
	6.2 Social Feasibility	22
	6.3 Economic Feasibility	22
7	TESTING	23
	7.1 Black Box Testing	23
	7.2 White Box Testing	23
	7.3 Unit Testing	24
	7.4 Integration Testing	24
	7.5 Validation Testing	24
	7.6 Performance Testing	24
8	FUTURE ENHANCEMENTS	25
9	CONCLUSION	26
10	APPENDIX	28
	10.1 Snap shots	28
	10.2 Sample code	31
	REFERENCES	78

INTRODUCTION

CHAPTER 1 INTRODUCTION

The World Wide Web is a rich source of information and continues to expand in size and complexity. How to efficiently and effectively retrieve required Web pages is becoming a great challenge. Traditional Web page search is based on user's query terms and Web search engines, such as *AltaVista* and *Google*. The user issues the query terms (keywords) to a search engine and the search engine returns a set of pages that may (hopefully) be related to the query topics or terms. For an interesting page, if the user wants to search the relevant pages further, he/she would prefer those relevant pages to be in hand. Here, a relevant Web page is the one that addresses the same topic as the original page, but is not necessarily semantically identical. Providing relevant pages for a searched Web page would prevent users from formulating new queries for which the search engine may return many undesired pages. Furthermore, for a search engine, caching the relevant pages instead for a set of searched pages would greatly speed up the Web search and increase the search efficiency. That is why many search engines, such as *Google* and *AltaVista*, are concerned more about building this functionality of searching relevant pages.

There are many ways to find relevant pages. For example, as indicated, *Netscape* uses Web page content analysis, usage pattern information, as well as linkage analysis to find relevant pages. Among the approaches of finding relevant pages, hyperlink analysis has its own advantages.

Primarily, the hyperlink is one of the most obvious features of the Web and can be easily extracted by parsing the Web page codes. Most importantly, hyperlinks encode a considerable amount of latent human judgment in most cases. In fact, with a few exceptions, the creators of Web pages create links to other pages, usually with an idea in mind that the linked pages are relevant to the linking pages. Therefore, a hyperlink, if it is reasonable, reflects the human semantic judgment and this judgment is objective and independent of the synonymy and polysemy of the words in the pages. This latent semantics, once revealed, could be used to find deeper relationships among the pages, as well as to find the relevant pages for a given page. The hyperlink analysis has proven success in many Web related areas, such as page ranking in the search engine *Google* Web page community construction Web search improvement Web clustering and visualization and relevant page finding. When hyperlink analysis is applied to the relevant page finding, its success depends on how to solve the following two problems:

- 1) How to construct a page source that is related to the given page and
- 2) How to establish effective algorithms to find relevant pages from the page source.

Ideally, the page source, a page set from which the relevant pages are selected, should have the following properties:

1. The size of the page source (the number of pages in the page source) is relatively small.
2. The page source is rich in relevant pages.

The best relevant pages for the given page, based on the statement, should be those that address the same topic as the original page and are semantically relevant to the original one. For convenience, in this work, we adapt the following concepts: If there is a hyperlink from page P to page Q, P is called a parent of Q and Q is called a child of P; if two pages have at least one common parent page, these two pages are called *siblings*.

The input is given in the text box provided which is the page queried by the user. The web link prober then connects to the internet and opens the source page for the input. The links in the source page are extracted and displayed to the user in the *tree view component* of the input form. The tree view component classifies the links found in the source page and displays according to the category of domains. It also displays the *parent nodes* and *child nodes* to the user with the details of the webmasters currently online. *Independent nodes* are also displayed to the user.

STUDY PROPOSAL

CHAPTER 2

STUDY PROPOSAL

2.1 Problem Definition

When the user browses through a website to obtain specific information at regular interval of time, the user does by giving text to the search engine to which it returns the websites or pages specific to that text. It does not analyze the source code of the web page and also limited links can only be retrieved. Users do not get website which are used before and the pages which get expired.

2.2 Existing System

In the existing system we have only search engines based on text information in which HITS(*Hyperlink-Induced Topic Search*) algorithm is used in order to implement the search process, which does not analyze the source code. HITS algorithm is applied directly to this page source and the top *authority pages* (e.g., 10 pages) with the highest authority weights are considered to be the relevant pages of the given page. Here, the authority pages are those that contain the most definitive, central, and useful information in the context of particular topics. On the other hand, the experiments of the above work show that the identified relevant pages are related to the given page in a broad sense, but are not semantically relevant to the given page in most cases. For example, given a page (URL):

<http://www.honda.com>, which is the home page of Honda Motor Company, the relevant pages returned by these algorithms are those home pages of different motor companies (e.g., Ford, Toyota, Volvo, etc.). Although these

relevant pages all address the same topic “*motor company*,” there are no relevant pages referring to Honda Motor Company, Honda Motor, or anything else about Honda and, furthermore, there exist no hyperlinks between the most of the relevant pages and the given page (URL). This kind of relevant pages could be considered relevant in a broad sense to the given page. In practical Web search, however, users usually would prefer those relevant pages that address the same topic as the given page, as well as being semantically relevant to the given page (best relevant pages).

2.3 Proposed System

In the proposed system we find information using the URL and we use the Singular Value Decomposition (SVD) algorithm and Latent Linkage Information (LLI) algorithm. The system proposes an efficient hyper link –based Algorithm to find the relevant links for a given web page (URL). The algorithm is advantaged with linear Algebra Theories to reveal deeper relationship among the web page to identify relevant links more precisely and efficiently. The hyperlink analysis has proven success in many Web related areas. It analyses the source code and provides the information efficiently. Latent Linkage Information (LLI) algorithm, finds relevant pages more effectively and precisely by using linear algebra theories, especially the singular value decomposition of matrix, to reveal deeper relationships among the pages. Experiments are conducted and it is shown that the proposed algorithms are feasible and effective in finding relevant pages, as the relevant pages returned by this algorithm contain those that address the same topic as the given page, as well as those that address the same topic and are semantically relevant to the given page. This is the ideal situation for which we look.

Advantages

- Web page community construction.
- Web search improvement.
- Web clustering and visualization and relevant page finding.
- Web oriented researches.
- Finding the weight age of a particular web site.
- Finding the number of links of a particular web site.
- The specific links can be found in a particular web site.
- The parent and child links of a particular web site can be found.
- The images present in the website also can be found.

SYSTEM PROPOSAL

CHAPTER 3

SYSTEM PROPOSAL

3.1 Software Specification

Software required: Visual Studio.Net (VB.Net)

Operating system: Windows2000 or Windows XP

3.2 Hardware Specification

CPU Type	:	Pentium IV
Hard Disk	:	40 GB
RAM Memory	:	512 MB
Monitor	:	17" Samtron color monitor

3.3 System Analysis

The first stage of the software development is study of the system under consideration and its requirement analysis. The system analysis is process of gathering facts, interpreting them, diagnosing various possible problems and using the above information to recommend improvement to the existing system. Existing system is analyzed and requirements are identified. Finally what proposed system was supposed to do is determined.

3.3.1 Language Description

.NET represents an entire range of technologies and concepts that form a platform on which you can develop applications. .NET is a layer that exists beneath the programs and provides a set of base services and functions. This layer contains a set of applications and operating system called .NET servers; a foundation set of objects called .NET framework and a set of services and support all .NET languages and CLR.

ADVANTAGES

1. Developer productivity

Developers of all backgrounds find .NET framework faster because the code is provided in the class libraries and .NET framework handles memory management. Hence enabling developers to read or achieve high productive gains.

2. Improved Reliability

.NET framework can monitor the health of running application, isolate application hence application built using .NET application stay up and run longer.

3. Increased Performance

Compilation, caching technique and server application are made faster in order of 300-500% with .NET framework.

4. Powerful granular security

Code access security technique in .NET framework is designed for internet environment .NET frameworks collect evident about origin and author of the application .NET frameworks compare the evidence with default security policies and makes decision whether to run the application.

5. Ease of Deployment

.NET frameworks make it easy to deploy run and manage application.

6. Mobility support

.NET frameworks provide one unified programming model for developing smart client web application for both pc and mobile device.

7. Flexible data access

.NET is designed for web-based style of data access ADO.NET data access tree up database connection and results in greater scalability.

ALGORITHM DESCRIPTION

CHAPTER 4

ALGORITHM DESCRIPTION

4.1 SINGULAR VALUE DECOMPOSITION (SVD) :

The SVD definition of a matrix is as follows: Let $A = [a_{ij}]_{m \times n}$ be a real $m \times n$ matrix. Without loss of generality, we suppose $m \geq n$ and the rank of A is $\text{rank}(A) = r$. Then, there exist orthogonal matrices $U_{m \times m}$ and $V_{n \times n}$ such that

$$A = U \begin{pmatrix} \Sigma_1 \\ 0 \end{pmatrix} V^T = U \Sigma V^T,$$

where

$$U^T U = I_m, V^T V = I_n, \Sigma_1 = \text{diag}(\sigma_1, \dots, \sigma_r), \sigma_i \sigma_{i+1} > 0$$

for $1 \leq i \leq r-1$, $\sigma_j = 0$ for $j \geq r+1$, Σ is an $m \times n$ matrix, U^T and V^T are the transpositions of matrices U and V , respectively, I_m and I_n represent $m \times m$ and $n \times n$ identity matrices, separately. The *rank* of A indicates the maximal number of independent rows or columns of A . Equation (1) is called the singular value decomposition of matrix A . The singular values of A are diagonal elements of (i.e., $\sigma_1, \sigma_2, \dots, \sigma_r$). The columns of U are called left singular vectors and those of V are called right singular vectors.

Since the singular values of A are in non increasing order, it is possible to choose a proper parameter k such that the last $r - k$ singular values are much smaller than the first k singular values and these k singular values dominate the decomposition.

4.2 LATENT LINKAGE INFORMATION (LLI) ALGORITHM:

We suppose the size of BS is m (e.g., the number of pages in BS is m) and size of P_u is n , the sizes of FS and C_u are p and q , respectively. Without loss of generality, we also suppose $m > n$ and $p > q$. The topological relationships between the pages in BS and P_u are expressed in a linkage matrix A , and the topological relationships between the pages in FS and C_u are expressed in another linkage matrix B . The linkage matrices A and B are concretely constructed as follows:

$$A = (a_{ij})_{m \times n},$$

$$a_{ij} = \begin{cases} 1 & \text{when page } i \text{ is a child of page } j, \text{ page } i \in BS, \text{ page } j \in P_u, \\ 0 & \text{otherwise.} \end{cases}$$

$$B = (b_{ij})_{p \times q},$$

$$b_{ij} = \begin{cases} 1 & \text{when page } i \text{ is a parent of page } j, \text{ page } i \in FS, \text{ page } j \in C_u, \\ 0 & \text{otherwise.} \end{cases}$$

These two matrices imply more beneath their simple definitions. In fact, the i th row of matrix A can be viewed as the coordinate vector of page i (page $i \in BS$) in an n -dimensional space spanned by the n pages in P_u and the i th row of matrix B can be viewed as the coordinate vector of page i (page $i \in FS$) in a q -dimensional space spanned by the q pages in C_u .

Similarly, the j th column of matrix A can be viewed as the coordinate vector of *page* j (*page* $j \in P_u$) in an m -dimensional space spanned by the m pages in BS . The meaning is similar for the columns in matrix B . In other words, the topological relationships between pages are transferred, via the matrices A and B , to the relationships between vectors in different multidimensional spaces.

Since A and B are real matrices, there exist SVDs of A and B :

$$A = U_{m \times m} \Sigma_{m \times n} V^T_{n \times n}, B = W_{p \times q} \Omega_{p \times q} X^T_{q \times q}.$$

As indicated above, the rows of matrix A are coordinate vectors of pages of BS in an n -dimensional space.

Therefore, all the possible inner products of pages in BS can be expressed as AA^T , i.e., $(AA^T)_{ij}$ is the inner product of *page* i and *page* j in BS . Because of the orthogonal properties of matrices U and V , we have $AA^T = (U\Sigma)(U\Sigma)^T$.

Matrix $U\Sigma$ is also an $m \times n$ matrix. It is obvious from this expression that matrix $U\Sigma$ is equivalent to matrix A and the rows of matrix $U\Sigma$ could be viewed as coordinate vectors of pages in BS in *another* n -dimensional space.

The SVD of a matrix is not a simple linear transformation of the matrix, it reveals statistical regulation of matrix elements to some extent. Accordingly, the coordinate vector transformation from one space to another space via SVD makes sense. For the same reason, the rows of matrix $V\Sigma^T$, which is an $n \times m$ matrix, are coordinate vectors of pages in P_u in *another* m -dimensional space. Similarly, for matrix B , the rows of matrix $W\Omega$ are coordinate vectors of pages in FS in another q -dimensional space and the rows of matrix $X\Omega^T$ are coordinate vectors of pages in C_u in another p -dimensional space.

Matrix $U\Sigma$ is also an $m \times n$ matrix. It is obvious from this expression that matrix $U\Sigma$ is equivalent to matrix A and the rows of matrix $U\Sigma$ could be viewed as coordinate vectors of pages in BS in *another* n -dimensional space.

The SVD of a matrix is not a simple linear transformation of the matrix, it reveals statistical regulation of matrix elements to some extent. Accordingly, the coordinate vector transformation from one space to another space via SVD makes sense. For the same reason, the rows of matrix $V\Sigma^T$, which is an $n \times m$ matrix, are coordinate vectors of pages in P_u in *another* m -dimensional space. Similarly, for matrix B , the rows of matrix $W\Omega$ are coordinate vectors of pages in FS in another q -dimensional space and the rows of matrix $X\Omega^T$ are coordinate vectors of pages in C_u in another p -dimensional space.

the coordinate vector transformation from one space to another space via SVD makes sense. For the same reason, the rows of matrix $V\Sigma^T$, which is an $n \times m$ matrix, are coordinate vectors of pages in P_u in *another* m -dimensional space. Similarly, for matrix B , the rows of matrix $W\Omega$ are coordinate vectors of pages in FS in another q -dimensional space and the rows of matrix $X\Omega^T$ are coordinate vectors of pages in C_u in another p -dimensional space.

Similarly, for matrix B , the rows of matrix $W\Omega$ are coordinate vectors of pages in FS in another q -dimensional space and the rows of matrix $X\Omega^T$ are coordinate vectors of pages in C_u in another p -dimensional space.

Similarly, for matrix B , the rows of matrix $W\Omega$ are coordinate vectors of pages in FS in another q -dimensional space and the rows of matrix $X\Omega^T$ are coordinate vectors of pages in C_u in another p -dimensional space.

Similarly, for matrix B , the rows of matrix $W\Omega$ are coordinate vectors of pages in FS in another q -dimensional space and the rows of matrix $X\Omega^T$ are coordinate vectors of pages in C_u in another p -dimensional space.

Next, we discuss matrices A and B separately. For the SVD of matrix A , matrices U and V can be denoted, respectively, as

$$U_{m \times m} = [u_1, u_2, \dots, u_m]_{m \times m}$$

and

$$V_{n \times n} = [v_1, v_2, \dots, v_n]_{n \times n},$$

where $u_i (i = 1, \dots, m)$ is an m -dimensional vector

$u_i = (u_{1,i}, u_{2,i}, \dots, u_{m,i})^T$ and $v_i (i = 1, \dots, n)$ is an n -dimensional vector

$v_i = (v_{1,i}, v_{2,i}, \dots, v_{n,i})^T$. Suppose $rank(A) = r$ and singular values of matrix A are as follows:

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > \sigma_{r+1} = \dots = \sigma_n = 0.$$

For a given threshold $\varepsilon (0 < \varepsilon \leq 1)$, we choose a parameter k such that

$(\sigma_k - \sigma_{k+1}) / \sigma_k \geq \varepsilon$. Then, we denote $U_k = [u_1, u_2, \dots, u_k]_{m \times k}$,

$V_k = [v_1, v_2, \dots, v_k]_{n \times k}$, $\Sigma_k = diag(\sigma_1, \sigma_2, \dots, \sigma_k)$, and $A_k = U_k \Sigma_k V_k^T$.

The best approximation matrix A_k contains main linkage information among the pages and makes it possible to filter those irrelevant pages, which usually have fewer links to the parents of given u , and effectively find relevant pages. In this algorithm, the relevance of a page to the given page u is measured by the similarity between them. For measuring the page similarity based on A_k , we choose the i th row R_i of the matrix $U_k \Sigma_k$ as the coordinate vector of page i (page $i \in BS$) in a k -dimensional subspace S :

$$R_i = (u_{i1}\sigma_1, u_{i2}\sigma_2, \dots, u_{ik}\sigma_k), \quad i = 1, 2, \dots, m. \quad (1)$$

For the given page u , since it is linked by every parent page, it is represented as a coordinate vector with respect to the pages in $P_u : u = (g_1, g_2, \dots, g_n)$, where $g_i = 1, i \in [1, n]$. The projection of coordinate vector u in the k -dimensional subspace S is represented as

$$u' = uV_k\Sigma_k = (g'_1, g'_2, \dots, g'_k), \quad (2)$$

where $g'_i = \sum_{t=1}^n g_t v_{it} \sigma_i$, $i = 1, 2, \dots, k$.

Equations (1) and (2) map the pages in BS and the given page u into the vectors in the same k -dimensional subspace S in which it is possible to measure the similarity (relevance degree) between a page in BS and the given page u . We take the commonly used cosine similarity measurement for this purpose, i.e., for two vectors $x = (x_1, x_2, \dots, x_k)$ and $y = (y_1, y_2, \dots, y_k)$ in a k -dimensional space, the similarity between them is defined as

$$sim(x \cdot y) = \frac{|x \cdot y|}{\|x\|_2 \|y\|_2},$$

where $x \cdot y = \sum_{i=1}^k x_i y_i$, $\|x\|_2 = \sqrt{x \cdot x}$. In this way, the similarity between a page i in BS and the given page u is defined as

$$BSS_i = sim(R_i \cdot u') = \frac{|R_i \cdot u'|}{\|R_i\|_2 \|u'\|_2}, \quad i = 1, 2, \dots, m.$$

For the given selection threshold δ , the relevant pages in BS with respect to the given page u is the set

$$BSR = \{ p_i \mid BSS_i \geq \delta, p_i \in BS, i = 1, 2, \dots, m \}.$$

In the same way, for the SVD of matrix $B = W_{p \times q} \Omega_{p \times q} X^T_{q \times q}$, we suppose $rank(B) = t$ and singular values of matrix B are

$\omega_1 \geq \omega_2 \geq \dots \geq \omega_t > \omega_{t+1} = \dots = \omega_q = 0$. For a given threshold ε ($0 < \varepsilon < 1$)⁴, we choose a parameter l such that $(\omega_l - \omega_{l+1}) / \omega_l \geq \varepsilon$.

Then, we denote $B_l = W_l \Omega_l X_l^T$, where

$$W_l = [w_{ij}]_{p \times l}, X_l = [x_{ij}]_{l \times q}, \Omega_l = diag(\omega_1, \omega_2, \dots, \omega_l).$$

The i th row R'_i of the matrix $W_l \Omega_l$ is the coordinate vector of page i (page $i \in FS$) in a l -dimensional subspace L :

$$R'_i = (\omega_{i1}\omega_1, \omega_{i2}\omega_2, \dots, \omega_{il}\omega_l), \quad i = 1, 2, \dots, p.$$

The projection of coordinate vector u in the l -dimensional subspace L is represented as

$$u'' = uX_l \Omega_l = (g_1'', g_2'', \dots, g_l''),$$

where

$$g_l'' = \sum_{j=1}^q g_j x_{ji} \omega_l, \quad i = 1, 2, \dots, l.$$

Therefore, the similarity between a page i in FS and the given page u is

$$FSS_i = sim(R'_i, u'') = \frac{|R'_i \cdot u''|}{\|R'_i\|_2 \|u''\|_2}, \quad i = 1, 2, \dots, p.$$

For the given selection threshold δ , the relevant pages in FS with respect to the given page u is the set

$$FSR = \{ p_i \mid FSS_i \geq \delta, p_i \in FS, i = 1, 2, \dots, p \}.$$

Finally, the relevant pages of the given page (URL) u is a page set $RP = BSR \cup FSR$.

The complexity or computational cost of the LLI is dominated by the SVD computation of the linkage matrices A and B . Without loss of generality, we suppose $m = \max(m, p)$ and $n = \max(n, q)$. Then, the complexity of the LLI algorithm is $O(m^2n + n^3)$. If $n \ll m$, this complexity is approximately $O(m^2)$. Since the number of pages in the page source can be controlled by the algorithm and this number is relatively very small compared with the number of pages on the Web, the LLI algorithm is feasible for application.

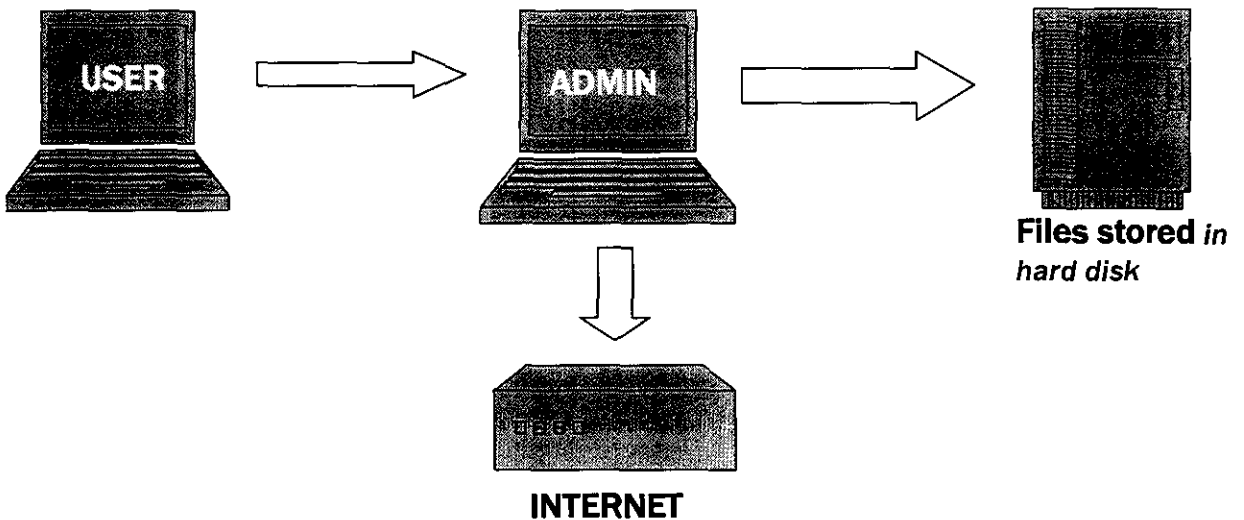
SYSTEM DESIGN

CHAPTER 5 SYSTEM DESIGN

The system design deals with various designs involved in this project which can be classified as:

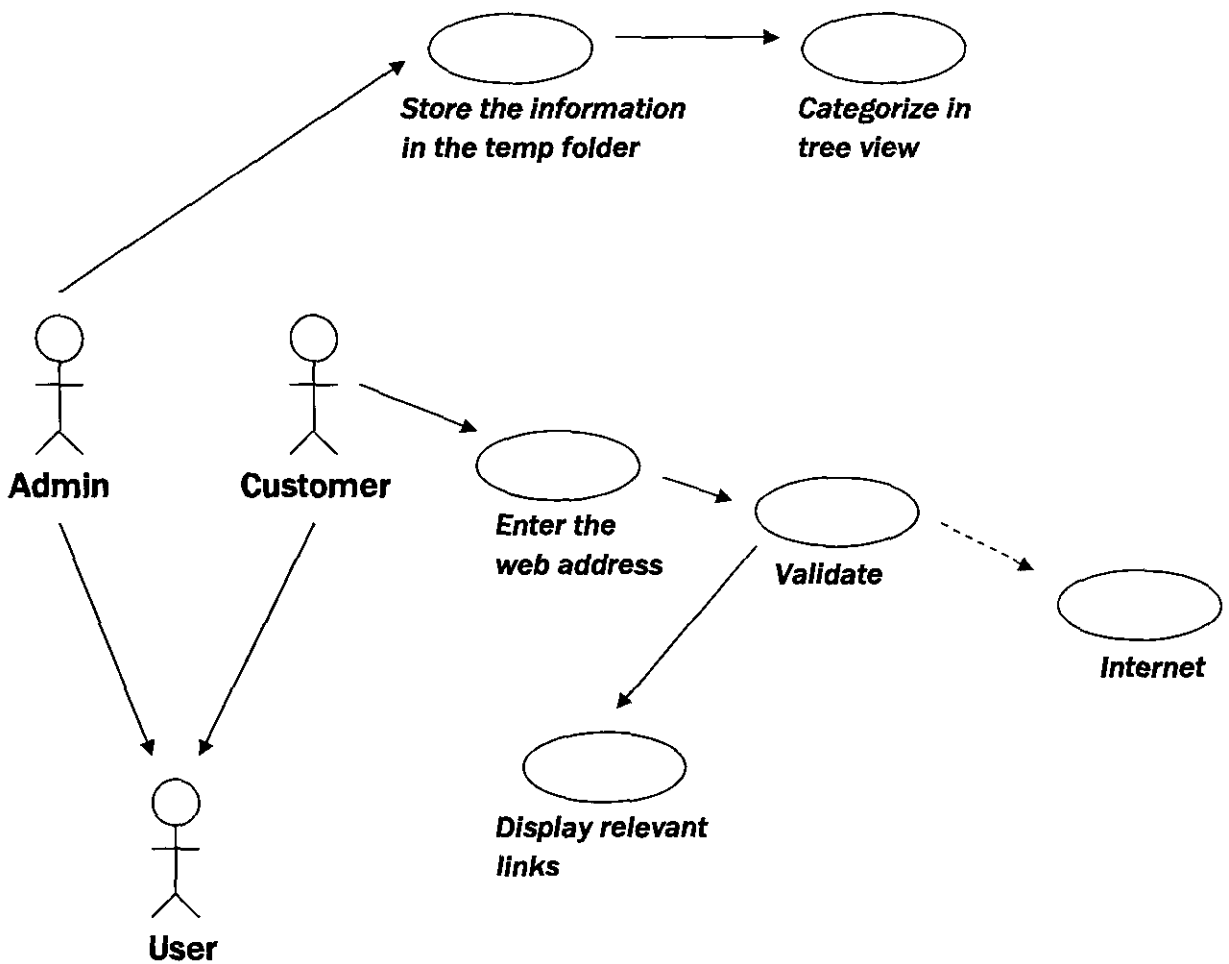
- 1) Architectural Design
- 2) Process Design
- 3) Form Design
- 4) Form layout

5.1 ARCHITECTURAL DESIGN

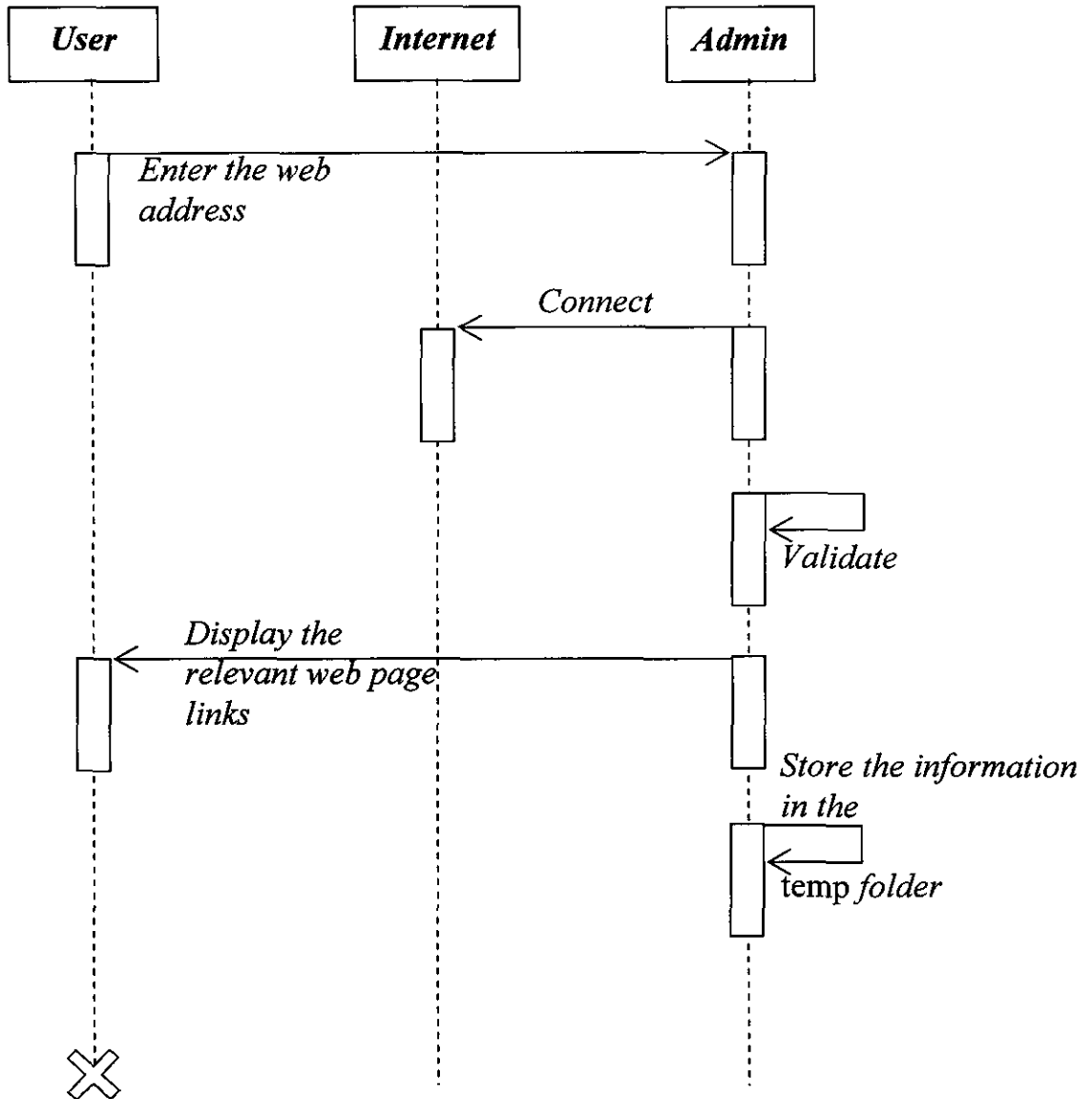


5.2 PROCESS DESIGN

5.2.1 Use Case Diagram



5.2.2 Sequence Diagram



5.3 FORM DESIGN

We are designing a form using VB.NET to retrieve information. This is showed below. This form has root directory and parent links. In the form design we have various buttons, which are used to implement the process.

Start

It is used to start the process and it keeps on searching the given URL and displays all the links.

Stop

It is used to stop the search process from which it is searching.

Restart

It is used to restart the process from anywhere of the procedure.

Favorites

It is used to store the links, which are used by the user often. It displays links, radio station guide msn.com etc.

History

This is used to show the previous links, which are used by the users. It displays the searched information.

Exit

It is used to exit the application.

5.3.1. COMPONENTS

Tree view

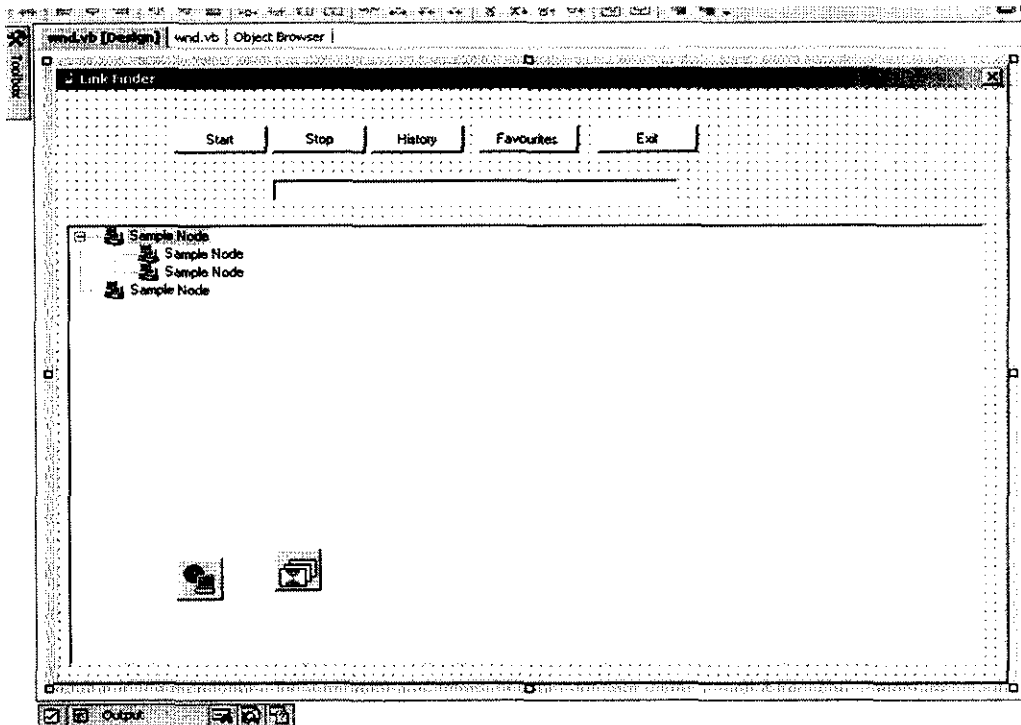
We use tree view to display a hierarchy of nodes. Each node is not only displayed visually, but also has child nodes. An example of this is the windows explorer, which uses a tree view in its left pane to display the hierarchy of folders on disk. With the use of this component memory space will be reduced. A tree view also can be displayed with checkboxes next to the nodes, if the tree view uses check boxes. The main properties of the tree view are the nodes and selected node. The nodes property contains the list of nodes in the tree view, and the selected node property sets the currently selected node. Tree view class supports nodes themselves.

The nodes collection for a node holds the nodes child tree node objects. You can add, remove a tree node, when you do, all child tree nodes are added, removed at the same time. Each tree node can contain a collection of other tree node objects, which means you can use expressions like this: My node, Nodes (3), Nodes (5) to refer to child nodes. You can also use the full path property to specify nodes in terms of their absolute.

Inet

This is used to connect system to internet.

5.3. FORM LAYOUT



FEASIBILITY ASSESSMENT

CHAPTER 6

FEASIBILITY ASSESSMENT

The feasibility study is performed to test the technical, social and economical feasibility of developing this system. Investigating the existing system and generating ideas about the new system do this.

4.1. Technical feasibility

This system technically feasible as it facilitates through user to know about the current updating by the simple operation of a button click. It also aids the user by informing the status of the information retrieval. The project ensures provider's security as it operates only in the dial up connection.

4.2. Social feasibility

For people with knowledge of data ware housing this is a new technical solution where in the web administration, customers of the commerce website, web miners and wireless users will find their monitoring of the system judicious.

4.3. Economic feasibility

Cost certainly becomes a secondary factor when there is reduction of time and human effort. The software designed reduced time and requires less effort of an individual as remembering the facts or data's are not of much importance. The project if implemented by the service provider would help anybody on the net to have excellent knowledge of the changes or improvement made. From the above feasibility study the proposed system is found to be cost efficient, less time consuming, flexible and reliable and has the high degree of acceptance.

TESTING

CHAPTER 7

TESTING

Testing plays a vital role to the success of the project, which is the last stage of the software development. The testing stage is proposed to affirm the quality of the project, to find and eliminate any residual errors from the previous stage, to validate software and to eliminate the operational reliability of the system.

7.1. Black Box Testing

This test demonstrates that all the software functions are operational, input is properly accepted, correct outputs are generated and the integrity of the external information is maintained. This testing was conducted to detect errors. The internal coding is not considered and only the user enters the input in the text box and checks whether he obtains the relevant links in the tree view component.

7.2. White box testing

White box testing are the software predicates on close examination of procedure details. It provides test cases that exercise specific test for conditions and loops. White box testing was carried out in the order to guarantee that

1. All independent paths within a module were exercised at least once.
2. All logical decision on this true and false side was exercised.

7.3. Unit testing

The individual modules are tested for the proper functioning and are found to be satisfactory. The algorithm modules are tested for accurate identification of nodes and links from source code of the web link. The search engine development module is tested for the recognition of web page details. The algorithm module is tested for its Html stripping method and the display of relevant links in the tree view component.

7.4. Integrated testing

As modules are successfully tested, an integrated test plan is developed to incorporate each module into the overall software structure. As a whole the web address is given and the output nodes are verified whether it suits the input address.

7.5. Validation testing

This test is performed to validate the output obtained. The URL is given as the input and the corresponding links are obtained.

7.6. Performance testing

It is carried out to test the runtime performance of the system developed. The systems processing time is found to be dependent on the browsing speed, it comes around 3 seconds.

FUTURE ENHANCEMENTS

CHAPTER 7

FUTURE ENHANCEMENTS

This project deals with retrieval of information of web links by giving input URL has following future enhancements

- Retrieval of images.
- Retrieval of audio clips.
- Retrieval of video clips.

CONCLUSION

CHAPTER 8

CONCLUSION

In this work, we have proposed the algorithm to find relevant pages of a given page: and the *LLI (Latent Linkage Information)* algorithm. This algorithm is based on hyperlink analysis among the pages and takes a new approach to construct the page source. The new page source reduces the influence of the pages in the same Website (or mirror site) to a reasonable level in the page similarity measurement, avoids some useful information being omitted, and prevents the results from being distorted by malicious hyperlinks. This algorithm could identify the pages that are relevant to the given page in a broad sense, as well as those pages that are semantically relevant to the given page. Furthermore, the LLI algorithm reveals deeper (mathematical) relationships among the pages and finds out relevant pages more precisely and effectively.

Experimental results show the advantages of this algorithm. The ideas in this work would also be helpful to other linkage-related analysis. The proposed algorithm in this work, as well as those in the previous work, find relevant pages statically, as they only deal with the “*static*” links among the pages.

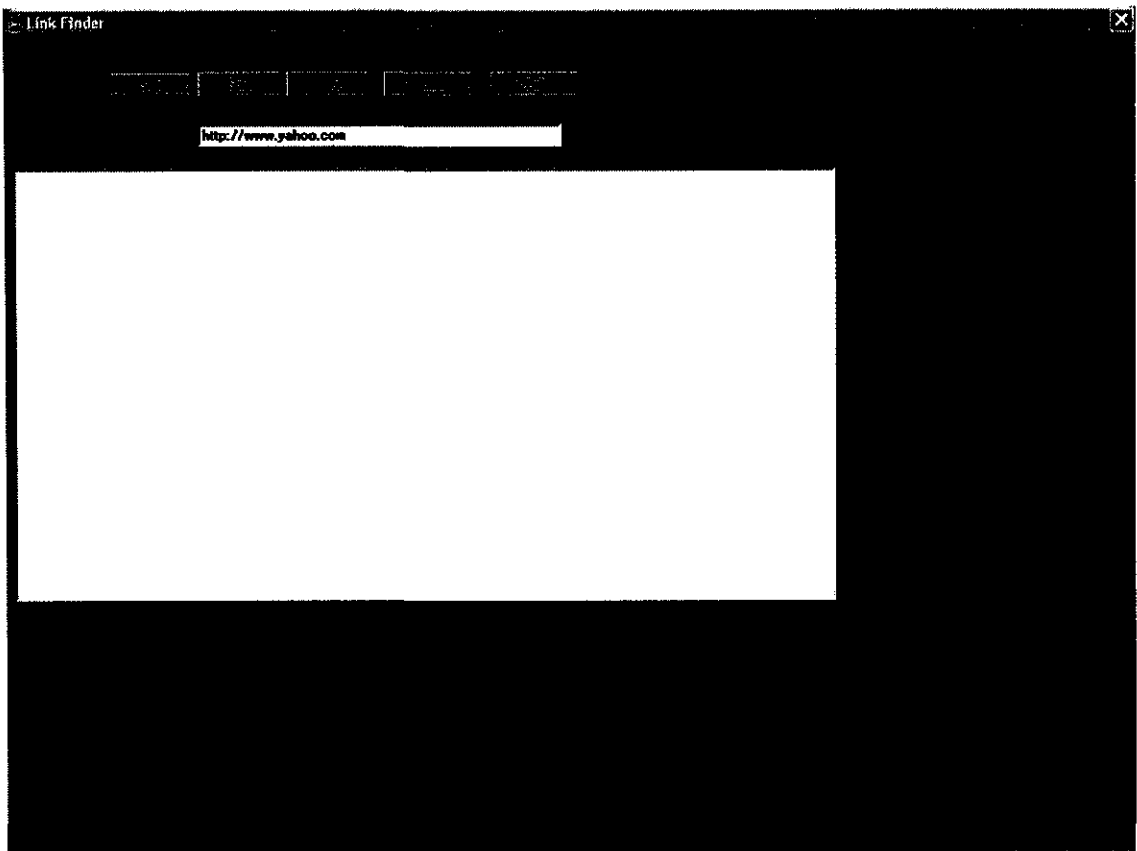
If the algorithm is implemented on the top of a hyperlink server such as the *Connectivity Server*, they are at most semi dynamic since the hyperlink information they use depends on the information update in the hyperlink database of the server. Extending the current algorithm to deal with dynamic links, such as those produced by a *CGI (Common Gateway Interface)* script is a valuable and a challenging problem.

The page similarity in the LLI algorithm could also be adapted for page clustering if the number of pages to be clustered is not huge. Assigning more semantics to hyperlinks, especially for XML documents, is another promising approach to increase the effectiveness in finding relevant pages (documents), clustering pages (documents), etc.

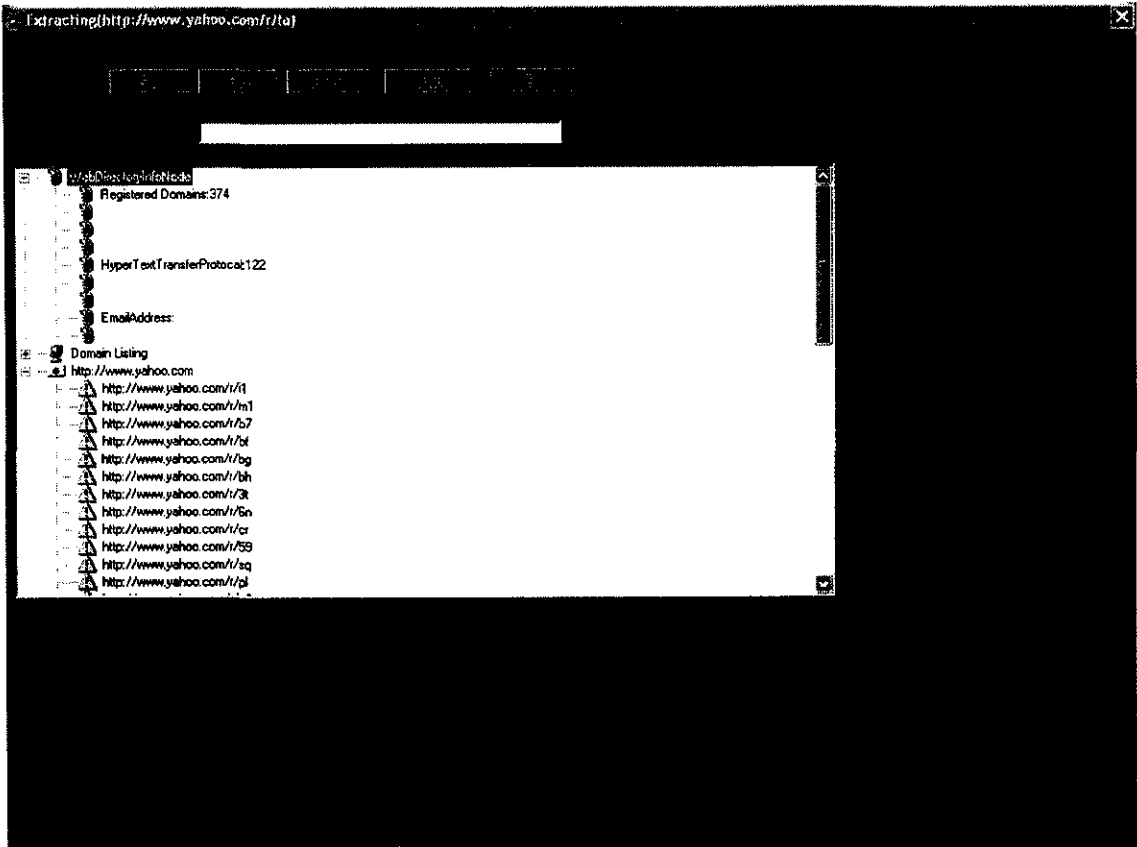
CHAPTER 9

APPENDIX

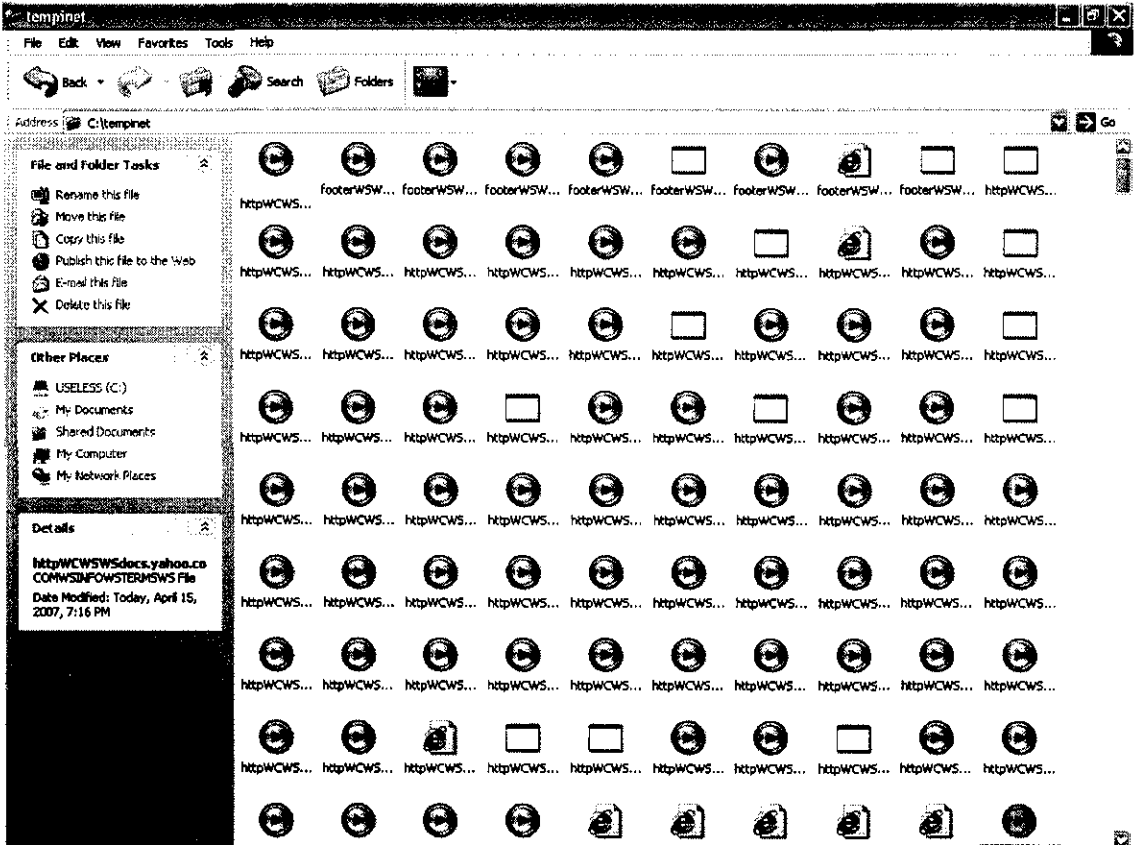
9.1. SNAPSHOTS



User entering the input in the provided text box



Extracted links from the user provided page shown in the tree view component



C:\tempinet folder showing the information stored by the admin

9.2 SAMPLE CODES

Imports System

Imports System.Reflection

Imports System.Runtime.InteropServices

' General Information about an assembly is controlled through the following
' set of attributes. Change these attribute values to modify the information
' associated with an assembly.

' Review the values of the assembly attributes

<Assembly: AssemblyTitle("")>

<Assembly: AssemblyDescription("")>

<Assembly: AssemblyCompany("")>

<Assembly: AssemblyProduct("")>

<Assembly: AssemblyCopyright("")>

<Assembly: AssemblyTrademark("")>

<Assembly: CLSCompliant(True)>

'The following GUID is for the ID of the typelib if this project is exposed to
COM

<Assembly: Guid("F2185C34-1FFD-41CC-9CA5-9815F7961B0D")>

' Version information for an assembly consists of the following four values:

,

' Major Version

' Minor Version

' Build Number

' Revision

,

' You can specify all the values or you can default the Build and Revision Numbers

' by using the '*' as shown below:

```
<Assembly: AssemblyVersion("1.0.*")>
```

Friend Class HtmlStrip

```
Dim Stripped(500) As Object
```

```
Dim CurrentIndex As Integer
```

```
Public idMaxLenth As Integer
```

```
Public szBlockStart As Object
```

```
Public szBlockEnd As Object
```

```
Function aGet(ByRef index As Object) As Object
```

```
    aGet = Stripped(index)
```

```
End Function
```

```
Sub aPut(ByRef index As Object, ByRef value As Object)
```

```
    Stripped(index) = value
```

```
End Sub
```

```
'<A HREF="MAILTO: FINDS ALL EMAIL LINKS"></A>
```

```
Sub SetClassToFindEmails()
```

```
    ' set header and footer block variables
```

```
    ' these determine what is to be found!
```

```
    Me.szBlockStart = "<a href=" & Chr(34) & "mailto:"
```

```
    Me.szBlockEnd = Chr(34)
```

```
    Me.idMaxLenth = 30
```

```
End Sub
```

"THIS FINDS EVERYTHING IN QUOTATIONS"

```
Sub SetClassToFindAllBlocks()
```

```
    Me.szBlockStart = Chr(34)
```

```
    Me.szBlockEnd = Chr(34)
```

```
    Me.idMaxLength = 30
```

```
End Sub
```

```
' <A HREF="FINDS ALL LINKS"></A>
```

```
Sub SetClassToFindLinks()
```

```
    Me.szBlockStart = "<a href=" & Chr(34)
```

```
    Me.szBlockEnd = Chr(34)
```

```
    Me.idMaxLength = 30
```

```
End Sub
```

```
Sub SetClassToFindScr()
```

```
    Me.szBlockStart = "scr=" & Chr(34)
```

```
    Me.szBlockEnd = Chr(34) ' you could add [& ">"]
```

```
    Me.idMaxLength = 30 ' no longer than 30 character
```

```
End Sub
```

```
' strips document of all links or emails or
```

```
' whatever is set to find..
```

```
Sub StripDocument(ByRef DocumentData As Object, ByRef Append As  
Boolean)
```

```
    Dim CurrentOffset As Integer
```

```
    Dim ChStart As Integer
```

```
    Dim ChEnd As Integer
```

```
' new append functions for links and images
```

```
If Append = False Then
```

```

    CurrentIndex = 0
Else
    CurrentIndex = CurrentIndex + 1
End If
CurrentOffset = 1
Do
    ' find header of block (get byte position)
    ChStart = InStr(CurrentOffset, DocumentData, szBlockStart)
    If ChStart > 0 Then
        '(**) if not header found exit do
        ' get footer of block (get byte position)
        ChEnd = InStr(ChStart + Len(szBlockStart), DocumentData,
szBlockEnd)
        If ChEnd > 0 Then
            '(**if no end found exit do
            ' check maxlenth
            If ChStart - ChEnd > idMaxLenth Then
                CurrentOffset = ChEnd ' move offset up.
            Else
                ' extract block from document
                Stripped(CurrentIndex) = Mid(DocumentData, ChStart, ChEnd
- ChStart)
                Stripped(CurrentIndex) = Mid(Stripped(CurrentIndex),
InStr(1, Stripped(CurrentIndex), Chr(34)) + 1, Len(Stripped(CurrentIndex)))
                ' move the index up for the next array position
                CurrentIndex = CurrentIndex + 1
                ' put offset above found email - or it will find it agin!

```

CurrentOffset = ChEnd

'Easy Debug Way! You can remove this it is not needed.

' Its just for demonstration and testing. Use g and p functions

' for accessing the array outside the function.

End If

Else

Exit Do

End If

Else

Exit Do

End If

System.Windows.Forms.Application.DoEvents()

Loop Until CurrentOffset > Len(DocumentData)

End Sub

End Class

Public Declare Function shellexecute Lib "shell32.dll" Alias "ShellExecuteA"
(ByVal hwnd As Integer, ByVal lpOperation As String, ByVal lpFile As
String, ByVal lpParameters As String, ByVal lpDirectory As String, ByVal
nShowCmd As Integer) As Integer

Public Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As
Integer)

Public cmlstop As Boolean

Public exitflag As Boolean

Public TempInternetFilePath As Object

Public llhwnd As Object

```

Public Sub StoreInternetPage(ByRef varPageData As Object, ByRef
varPageAddress As Object)
    Dim s As Object
    On Error Resume Next
    s = encode(varPageAddress, "W", "WW")
    s = encode(s, "/", "WS")
    s = encode(s, ":", "WC")
    s = encode(s, "?", "WQ")
    s = encode(s, "%", "WP")
    s = encode(s, "$", "WD")
    MkDir("c:\tempinet\")
    FileOpen(1, "c:\tempinet\" & s, OpenMode.Binary)
    FilePutObject(1, 1, varPageData)
    FileClose(1)
End Sub

```

```

Public Function RestoreFilenameToAddress(ByRef varFilename As
Object) As Object
    Dim s As Object
    s = encode(varFilename, "WD", "$")
    s = encode(s, "WP", "%")
    s = encode(s, "WQ", "?")
    s = encode(s, "WC", ":")
    s = encode(s, "WS", "/")
    RestoreFilenameToAddress = encode(s, "WW", "W")
End Function

```

```
Public Function encode(ByRef varString As Object, ByRef varBlock As
Object, ByRef varNewBlock As Object) As Object
    Dim i As Object
    For i = 1 To Len(varString)
        If Mid(varString, i, Len(varBlock)) = varBlock Then
            encode = encode & varNewBlock
            i = i + Len(varBlock) - 1
        Else
            encode = encode & Mid(varString, i, 1)
        End If
    Next i
End Function
End Module
```

```
Dim Searched(900) As Object
Dim SearchedIndex As Integer
Dim OODMODE As Boolean
Dim OOD_PURL As String
Dim INFMODE As Boolean
' Settings
Sub Lc2kSetMode_OOD(ByRef bValue As Boolean, ByRef purl As
Object)
    If bValue = True Then
        OODMODE = True
        OOD_PURL = purl
    Else
        OODMODE = False
```



```

    End If
End Sub
' Run Module
Sub Lc2kEngine(ByRef Url As String, ByRef view As
AxComctlLib.AxTreeView, ByRef inet As AxInetCtrlsObjects.AxInet, ByRef
PMaxNodes As Integer, ByRef ProcessorTime As Integer, ByRef Frm As
System.Windows.Forms.Form)
    Dim Gt As Object
    Dim i As Object
    Dim Data As Object
    Dim oldimage As Object
    On Error GoTo errh
    If cmlstop = True Then
        GoTo CleanUp
    Else
    End If
    Dim Stack As New W32LIB_Stack
    Dim Strip As New HtmlStrip
    Dim CurUrl As String
    Dim SDevice As New FSD
    INFMODE = True
    Stack.Reset_Renamed(1)
    Stack.Reset_Renamed(0)
    If INFMODE = True Then
        view.Nodes.Clear()
        view.Nodes.Add(, "info", "WebDirectoryInfoNode", 14)
        'view.Nodes.Add(, "info", "WebDirectoryInfoNode", 14)

```

```
view.Nodes.Add("info", 4, "rd", , 14)
view.Nodes.Add("info", 4, "gd", , 14)
view.Nodes.Add("info", 4, "tx", , 14)
view.Nodes.Add("info", 4, "cg", , 14)
view.Nodes.Add("info", 4, "hd", , 14)
view.Nodes.Add("info", 4, "ex", , 14)
view.Nodes.Add("info", 4, "hl", , 14)
view.Nodes.Add("info", 4, "em", , 14)
view.Nodes.Add("info", 4, "ot", , 14)
```

End If

' **** Major Domains ****'

```
view.Nodes.Add(, , "mdomains", "Domain Listing", 3)
```

If OODMODE = False Then

```
view.Nodes.Add(, , Url, Url, 15)
```

End If

```
Stack.Push(0, Url)
```

Do

Do

repop:

```
WaitProcessorTime(ProcessorTime, True)
```

```
CurUrl = Stack.pop(0)
```

```
If InStr(1, CurUrl, wnd.DefInstance.txtexclude.Text) > 0 Then
```

GoTo repop

```
If cmlstop = True Then
```

```
GoTo CleanUp
```

```
Else
```

```
End If
```

```

Frm.Show()
Frm.Text = "Init (" & CurUrl & ")"
If HasUrlBeenSearchedBefore(CurUrl) = True Then
    If Stack.lne(0) = True Then Exit Do
    GoTo repop
End If
SetUrlSearched(CurUrl)
Frm.Text = "Getting (" & CurUrl & ")"
oldimage = view.Nodes.Item(GetKeyIndex(CurUrl, view)).Image
view.Nodes.Item(GetKeyIndex(CurUrl, view)).Image = 11
Data = GetDocument(CurUrl, inet)
view.Nodes.Item(GetKeyIndex(CurUrl, view)).Image = oldimage
' ***** StoreInternetPage *****
StoreInternetPage(Data, CurUrl)
' ***** Add speacil node section
AddMajorDomain(CurUrl, view)
Frm.Text = "Extracting(" & CurUrl & ")"
Strip.SetClassToFindLinks()
Strip.StripDocument(Data, False)
Strip.SetClassToFindScr()
Strip.StripDocument(Data, True)
i = 0
Do
    WaitProcessorTime(ProcessorTime, True)
    Gt = Strip.aGet(i)
    If Gt = "" Then Exit Do
    If exitflag = True Then GoTo CleanUp

```

Gt = FixUrl(Gt, CurUrl)

If HasUrlBeenSearchedBefore(Gt) = False Then

Stack.Push(1, Gt)

Else

Gt = Gt & ".dl"

End If

If PMaxNodes < view.Nodes.Count Then GoTo CleanUp

If OODMODE = True Then CurUrl = OOD_PURL

' ***** StoreInternetPage *****

StoreInternetPage("", Gt)

' ***** Add Major Domain *****

AddMajorDomain(Gt, view)

AddNode(view, CurUrl, Gt)

i = i + 1

Loop While Strip.aGet(i) <> ""

If OODMODE = True Then GoTo CleanUp

Loop Until Stack.lne(0) = True

Stack.Reset_Renamed(0)

Stack.CopyStack(1, 0)

Stack.Reset_Renamed(1)

WaitProcessorTime(ProcessorTime, True)

Loop Until Stack.Peek(0, 0) = ""

CleanUp:

Stack.Reset_Renamed(0) : Stack.Reset_Renamed(1)

Frm.Text = "Done. LastDoc(" & CurUrl & ") Nodes:" &

view.Nodes.Count

Exit Sub

errh:

```
MsgBox(" " & Err.Description)
Lc2kInternetPageSupport.Sleep(500)
Err.Clear()
GoTo CleanUp
Stack.Reset_Renamed(0) : Stack.Reset_Renamed(1)
SearchedIndex = 0
GoTo repop
Exit Sub
End Sub
```

Function HasUrlBeenSearchedBefore(ByRef Url As Object) As Boolean

```
Dim i As Object
For i = 0 To SearchedIndex
    If Searched(i) = Url Then
        HasUrlBeenSearchedBefore = True
        Exit Function
    End If
Next i
HasUrlBeenSearchedBefore = False
End Function
```

Sub SetUrlSearched(ByRef Url As Object)

```
Searched(SearchedIndex + 1) = Url
SearchedIndex = SearchedIndex + 1
End Sub
```

```
Sub WaitProcessorTime(ByRef ProcessorTime As Object, ByRef UseL As Boolean)
```

```
    Dim s As Object
```

```
    Dim X As Object
```

```
    If UseL = True Then
```

```
        For X = 0 To ProcessorTime * 2
```

```
            System.Windows.Forms.Application.DoEvents()
```

```
        Next X
```

```
    Exit Sub
```

```
Else
```

```
    s = VB.Timer()
```

```
    Do While VB.Timer() - s < ProcessorTime
```

```
        System.Windows.Forms.Application.DoEvents()
```

```
    Loop
```

```
End If
```

```
End Sub
```

```
Function GetDocument(ByRef Url As String, ByRef inet As AxInetCtrlsObjects.AxInet) As Object
```

```
    On Error GoTo gde
```

```
    GetDocument = LCase(inet.OpenURL(Url))
```

```
    Exit Function
```

```
gde:
```

```
    GetDocument = ""
```

```
End Function
```

```
End Module
```

```
Dim fem, ftxt, fexe, fhtml, fdomains, fgdomains, fcgiasp, fdupl, fhlp, fot  
As Object
```

```
' if key has been used before then get its index
```

```
' (view.nodes.item(=index).text = "whatever")
```

```
Function GetKeyIndex(ByRef Key As Object, ByRef view As  
AxComctlLib.AxTreeView) As Object
```

```
Dim X As Object
```

```
For X = 1 To view.Nodes.Count
```

```
    If view.Nodes.Item(X).Key = Key Then
```

```
        GetKeyIndex = X
```

```
        Exit Function
```

```
    End If
```

```
Next X
```

```
GetKeyIndex = -1
```

```
Exit Function
```

```
End Function
```

```
' Find out if the key has been used before.
```

```
Function HasKeyBeenUsed(ByRef view As Object, ByRef Key As Object)  
As Object
```

```
Dim X As Object
```

```
For X = 1 To view.Nodes.Count
```

```
    If view.Nodes.Item(X).Key = Key Then
```

```
        HasKeyBeenUsed = True
```

```
        Exit Function
```

```
    End If
```

```
Next X
```

HasKeyBeenUsed = False

Exit Function

End Function

Sub AddNode(ByRef view As Object, ByRef ParentDocument As Object,
ByRef ThisDocument As Object)

' Takes care of setting the icons

' find the parents and everthing else.

Dim doctype As Object

Dim aicon As Object

Dim SMVParentDocument As Object

Dim SMParentMissing As Object

On Error Resume Next

' Check for parent of ThisDocument

If VFindParent(view, ParentDocument) = False Then

view.Nodes.Add(, , ParentDocument, ParentDocument)

SMParentMissing = True

SMVParentDocument = ParentDocument

End If

' Set Normal Icon

aicon = 13

' Get Document Type

doctype = GetUrlDocumentType(ThisDocument)

' Registerd Normal Domains

If doctype = "com" Or doctype = "net" Or doctype = "org" Then

aicon = 1

fdomains = fdomains + 1


```

        view.Nodes.Item(GetKeyIndex("rd", view)).Text = "Registered
Domains:" & fdomains
    End If
' Government Domains
If doctype = "gov" Then
    fgdomains = fgdomains + 1
    view.Nodes.Item(GetKeyIndex("gd", view)).Text = "Government
Domains:" & fgdomains
    aicon = 12
End If
' HTML or HTM Documents
If doctype = "htm" Or doctype = "html" Then
    aicon = 3
    fhtml = fhtml + 1
    view.Nodes.Item(GetKeyIndex("hd", view)).Text =
"HyperTextTransferProtocal:" & fhtml
End If
' ASP
If doctype = "asp" Then
    fcgiasp = fcgiasp + 1
    view.Nodes.Item(GetKeyIndex("cg", view)).Text = "CGI & ASP:" &
fcgiasp
    aicon = 4
End If
' CGI
If doctype = "cgi" Then
    fcgiasp = fcgiasp + 1

```

```

        view.Nodes.Item(GetKeyIndex("cg", view)).Text = "CGI & ASP:" &
fcgiasp
        aicon = 6
    End If
    ' Executables! Setups! Compressed Files! Encrypted!
    If doctype = "exe" Or doctype = "zip" Or doctype = "cab" Or doctype =
"ace" Or doctype = "dll" Then
        fexe = fexe + 1
        view.Nodes.Item(GetKeyIndex("ex", view)).Text =
"Executable/Compressed: " & fexe
        aicon = 7
    End If
    ' Link, mostly used by the internal working of
    ' this application
    If doctype = "dl" Then
        fdupl = fdupl + 1
        view.Nodes.Item(GetKeyIndex("dl", view)).Text = "DuplicateLinks:"
& fdupl
        aicon = 8
    End If
    ' Readable Documents (TextEditor, WordProcessor)
    If doctype = "txt " Or doctype = "doc" Or doctype = "wrđ" Then
        ftxt = ftxt + 1
        view.Nodes.Item(GetKeyIndex("tx", view)).Text =
"Text/Documents:" & ftxt
        aicon = 9
    End If

```

```

' DocumentType = HtmlHelp or WindowsHelp
If doctype = "chp" Or doctype = "hlp" Then
    fhlp = fhlp + 1
    view.Nodes.Item(GetKeyIndex("hl", view)).Text = "HelpFiles(hlp):"
& fhlp
    aicon = 10
End If
' DocumentType = Email!
If InStr(1, ThisDocument, "mailto:") = 1 Then
    aicon = 2
    view.Nodes.Item(GetKeyIndex("em", view)).Text = "EmailAddress:"
& fem
    End If
' temp!!!!
view.Nodes.Add(ParentDocument, 4, ThisDocument, ThisDocument,
aicon)
WriteRawTreeData(ParentDocument, 4, ThisDocument, ThisDocument,
aicon)
If aicon = 5 Then
    fot = fot + 1
    view.Nodes.Item(GetKeyIndex("ot", view)).Text = "Other:" & fot
End If
End Sub
'write rawtree data
Sub WriteRawTreeData(ByRef Parent As Object, ByRef aType As Object,
ByRef Key As Object, ByRef Child As Object, ByRef Icon As Object)
    FileOpen(1, "c:\urls.rtd", OpenMode.Append)

```

Debug.Write(Parent)

PrintLine(1, Parent)

PrintLine(1, aType)

PrintLine(1, Key)

PrintLine(1, Child)

PrintLine(1, Icon)

FileClose(1)

End Sub

'Gets the documents path is there is one.

Function GetUrlPath(ByRef document As Object) As Object

If Len(document) = 0 Then Exit Function

Dim X As Integer : X = Len(document)

Do

If Mid(document, X, 1) = "/" Then

 GetUrlPath = Mid(document, 1, X - 1) & "/"

 Exit Function

End If

X = X - 1

Loop Until X = 1

End Function

' find parent in view list, is used to see if document is

' already in the list also..

Function VFindParent(ByRef view As Object, ByRef Parent As Object) As

Boolean

Dim X As Object

For X = 1 To view.Nodes.Count

 If view.Nodes.Item(X).Key = Parent Then

VFindParent = True : Exit Function

End If

Next X

VFindParent = False

End Function

'Gets Documents Domain Type www.yahoo.com (=COM)

Function GetUrlDocumentDomainType(ByRef document As Object) As
Object

Dim yy As Object

Dim yyy As Object

Dim Y As Object

On Error GoTo le

Dim X As Integer : X = 1

' if www is present or http://

If InStr(1, document, "www.") > 0 Then

Y = InStr(1, document, "www.") + 5

Else

If InStr(1, document, "http://") > 0 Then

Y = InStr(1, document, "http://") + 8

Else

Y = 1

End If

End If

If InStr(Y, document, ".") > 0 Then

yyy = InStr(Y, document, ".")

Else

Exit Function

End If

If InStr(Y, document, "/") > 0 Then

yy = InStr(Y, document, "/") - 1

Else

' http://www.aol.com(/)hello/mypage.htm

' if / is presetn seperating domain from path

yy = Len(document)

End If

GetUrlDocumentDomainType = Mid(document, yy + 1, yy - yyy)

Exit Function

le:

GetUrlDocumentDomainType = ""

End Function

'Gets url extension page.htm (EXT=htm)

Function GetUrlDocumentType(ByRef document As Object) As Object

Dim bwithfolder As Object

Dim X As Integer : X = Len(document)

' if path contains only a folder then this means no file

' attached to extract!

If bwithfolder = True Then Exit Function

Do

If Mid(document, X, 1) = "." Then

GetUrlDocumentType = Mid(document, X + 1, Len(document))

Exit Function

End If

X = X - 1

Loop Until X = 1



End Function

'Fixes Url. ParentDocument is the document

' the link was found on. The link is refranced by Document

Function FixUrl(ByRef document As Object, ByRef ParentDocument As
Object) As Object

Dim urlpath As Object

Dim aext As Object

Dim bwithfolder As Object

Dim bwithdomain As Object

Dim bwithopar As Object

bwithopar = False

bwithdomain = False

bwithfolder = False

' is document without parent /~

If Mid(document, 1, 1) = "/" Then

 bwithopar = True

End If

' is document domain (com,net,org)

aext = GetUrlDocumentDomainType(document)

If aext = "org" Or aext = "com" Or aext = "net" Then bwithdomain =

True

' is document folder directory or file

' document is folder ~/

If InStr(1, document, "http://") > 0 Or InStr(1, document, "www.") > 0

Then bwithdomain = True

If Mid(document, Len(document), 1) = "/" Then

 bwithfolder = True

```

    ' document is file ~?
Else
    bwithfolder = False
End If
If bwithdomain = False And bwithopar = False Then GoTo FixPath
' fix document path if it has no parent
If bwithopar = True Then
FixPath:
    'fix document if url path is curropted by getpath.
    urlpath = GetUrlPath(ParentDocument)
    If LCase(urlpath) = "http://" Then
        If Mid(urlpath, Len(urlpath), 1) = "/" Then
            urlpath = ParentDocument
        Else
            urlpath = ParentDocument & "/"
        End If
    End If
    If Mid(document, 1, 1) <> "/" Then document = "/" & document
    If urlpath <> "" Then
        If Mid(urlpath, Len(urlpath), 1) <> "/" Then urlpath = urlpath & "/"
    End If
    document = urlpath & Mid(document, 2, Len(document))
End If
FixUrl = document
End Function
End Module

```



```
Dim EmailMessageRecp, EmailMessage As Object
```

```
' Get(s) Complete Domain Name And Type
```

```
Public Function GetUrlDomainName(ByRef varUrl As Object) As Object
```

```
    Dim ch3 As Object
```

```
    Dim ch2 As Object
```

```
    Dim OODMODE As Object
```

```
    If cmlstop = True Then
```

```
        OODMODE = True
```

```
        Exit Function
```

```
    Else
```

```
    End If
```

```
' lower case url
```

```
Dim ch1 As Integer
```

```
varUrl = LCase(varUrl)
```

```
ch1 = InStr(1, varUrl, "http://")
```

```
If ch1 > 0 Then ch1 = ch1 + 7
```

```
If ch1 < 1 Then ch1 = 1
```

```
ch2 = InStr(ch1, varUrl, "www.")
```

```
If ch2 < 1 Then
```

```
    If ch1 = 1 Then
```

```
        ch2 = 1
```

```
    Else
```

```
        ch2 = ch1
```

```
    End If
```

```
End If
```

```
' if no / at end then keep going
```

```
ch3 = InStr(ch2, varUrl, "/")
```

```
' ch2 - start of domain name
```

```
' ch3 - end of domain name+typename(com,net,org)
```

```
If ch3 = 0 Then ch3 = Len(varUrl) + 1
```

```
GetUrlDomainName = Mid(varUrl, ch2, ch3 - ch2)
```

```
End Function
```

```
' Removes Document Arguments and Extension
```

```
Public Function RemoveUrlArguments(ByRef varUrl As Object) As
```

```
Object
```

```
Dim i As Object
```

```
i = bbInstr(varUrl, "/.")
```

```
RemoveUrlArguments = Mid(varUrl, 1, i - 1)
```

```
End Function
```

```
' BackWord ByteInstr (multi-char find)
```

```
Public Function bbInstr(ByRef varString As Object, ByRef varTofind As
```

```
Object) As Object
```

```
Dim Y As Object
```

```
Dim i As Object
```

```
i = Len(varString)
```

```
Do
```

```
For Y = 1 To Len(varTofind)
```

```
    If Mid(varString, i, 1) = Mid(varTofind, Y, 1) Then
```

```
        bbInstr = i
```

```
        Exit Function
```

```
    End If
```

```
Next Y
```

```
i = i - 1
```

```
Loop Until i = 1
```

```
bbInstr = -1
```

```
End Function
```

```
Sub AddMajorDomain(ByRef Url As Object, ByRef view As  
AxComctlLib.AxTreeView)  
    Dim ii As Object  
    Dim i As Object  
    Dim MajorDomain As Object  
    If cmlstop = True Then  
        Exit Sub  
    Else  
        End If  
    ' for all major domains *****  
    MajorDomain = GetUrlDomainName(Url)  
    If exitflag = True Then Exit Sub  
    If GetKeyIndex(MajorDomain, view) = -1 Then  
        view.Nodes.Add("mdomains", 4, MajorDomain, MajorDomain)  
        view.Nodes.Add(MajorDomain, 4, MajorDomain & "Count", "1")  
        view.Nodes.Add(MajorDomain, 4, MajorDomain & "Email",  
"WebMaster@" & MajorDomain)  
        EmailMessageRecp = EmailMessageRecp & "WebMaster@" &  
MajorDomain & ", "  
        EmailMessage = EmailMessage & MajorDomain & Chr(13)  
        Clipboard.SetDataObject(EmailMessageRecp & Chr(13) &  
EmailMessage)  
    Else  
        ' *** below code also includes icon settings for servers ****
```

```

' reg server
i = GetKeyIndex(MajorDomain & "Count", view)
ii = GetKeyIndex(MajorDomain, view)
view.Nodes.Item(i).Text = CStr(CDbl(view.Nodes.Item(i).Text) + 1)
' meduim server
If CDbl(view.Nodes.Item(i).Text) > 50 Then
    view.Nodes.Item(ii).Image = 15
End If
' Big server
If CDbl(view.Nodes.Item(i).Text) > 150 Then
    view.Nodes.Item(ii).Image = 16
End If
If CDbl(view.Nodes.Item(i).Text) > 250 Then
    view.Nodes.Item(ii).Image = 17
End If
If CDbl(view.Nodes.Item(i).Text) > 350 Then
    view.Nodes.Item(ii).Image() = 18
End If
End If
End Sub
End Module

```

```

Public Declare Function FindWindowEx Lib "user32" Alias
"FindWindowExA" (ByVal hWnd1 As Integer, ByVal hWnd2 As Integer,
ByVal lpsz1 As String, ByVal lpsz2 As String) As Integer

```

```

Public Declare Function SendMessage Lib "user32" Alias
"SendMessageA" (ByVal hwnd As Integer, ByVal wParam As Integer, ByVal
wParam As Integer, ByVal lParam As Object) As Integer
Public Const WM_GETTEXT As Short = &HDS
Public Const WM_GETTEXTLENGTH As Short = &HES
Private Declare Function shellexecute Lib "shell32.dll" Alias
"ShellExecuteA" (ByVal hwnd As Integer, ByVal lpOperation As String,
ByVal lpFile As String, ByVal lpParameters As String, ByVal lpDirectory As
String, ByVal nShowCmd As Integer) As Integer
Public Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As
Integer)
Public Declare Function FindWindow Lib "user32" Alias "FindWindowA"
(ByVal lpClassName As String, ByVal lpWindowName As String) As
Integer
Public Declare Function SendMessageLong Lib "user32" Alias
"SendMessageA" (ByVal hwnd As Integer, ByVal wParam As Integer, ByVal
wParam As Integer, ByVal lParam As Integer) As Integer
Public Declare Function SendMessageByString Lib "user32" Alias
"SendMessageA" (ByVal hwnd As Integer, ByVal wParam As Integer, ByVal
wParam As Integer, ByVal lParam As String) As Integer
Const WM_USER As Short = &H400S
Const EM_LIMITTEXT As Integer = WM_USER + 21
Public Const EM_GETLINECOUNT As Short = &HBAS
Public Const EM_LINEINDEX As Short = &HBBS
Public Const EM_LINELENGTH As Short = &HC1S
Public cmlflag As Boolean
Public ioption As Object

```

```

Public lhwnd As Object
End Module

'// coded stack routines
Friend Class W32LIB_Stack
    Private Stack(50, 9000) As Object
    Private sx(50) As Object
    Sub LoadDebugWindow()
        Dim debugwin As Object
        Dim Y As Object
        Dim X As Object
        '// no copy if stack is empty
        For X = 0 To 50
            If GetStackPointer(X) = 0 Then
                Else
                    '// copy
                    For Y = 0 To GetStackPointer(X)
                        debugwin.Win32Stack.Push(X, Stack(X, Y))
                    Next Y
                End If
            Next X
            debugwin.Show()
        End Sub

        Sub WaitForDebugWindow()
            Dim debugwin As Object
            debugwin.CmdFlag = False

```

Do

System.Windows.Forms.Application.DoEvents()

Loop Until debugwin.CmdFlag = True

End Sub

Function GetStackPointer(ByRef stack_id As Object) As Object

GetStackPointer = sx(stack_id)

End Function

Function Peek(ByRef stack_id As Object, ByRef stack_index As Object)

As Object

'// return given address in stack

Peek = Stack(stack_id, stack_index)

If Peek = "" Then Peek = False

End Function

Sub CopyStack(ByRef stack_id_from As Object, ByRef stack_id_to As
Object)

'// set counter for destination stack

Dim X As Object

sx(stack_id_to) = sx(stack_id_from)

For X = 0 To sx(stack_id_from)

'// copy stack

Stack(stack_id_to, X) = Stack(stack_id_from, X)

Next X

End Sub

```
Sub Reset_Renamed(ByRef stack_id As Object)
```

```
    sx(stack_id) = 0 '// reset counter
```

```
    Stack(stack_id, 0) = 0
```

```
End Sub
```

```
Sub Push(ByRef stack_id As Object, ByRef value As Object)
```

```
    Stack(stack_id, sx(stack_id)) = value
```

```
    sx(stack_id) = sx(stack_id) + 1
```

```
End Sub
```

```
Function lne(ByRef stack_id As Object) As Object
```

```
    If sx(stack_id) = 0 Then
```

```
        lne = True
```

```
    Else
```

```
        lne = False
```

```
    End If
```

```
End Function
```

```
Function ppeek(ByRef stack_id As Object, ByRef offset As Object) As  
Object
```

```
    If sx(stack_id) + offset <= -1 Then
```

```
        ppeek = False
```

```
        Exit Function
```

```
    End If
```

```
    ppeek = Stack(stack_id, sx(stack_id) + offset)
```

```
End Function
```



```
Sub poke(ByRef stack_id As Object, ByRef stack_index As Object, ByRef
value As Object)
    Stack(stack_id, stack_index) = value
End Sub
```

```
Function pop(ByRef stack_id As Object) As Object
    If sx(stack_id) - 1 = -1 Then
        pop = False
        Exit Function
    End If
    sx(stack_id) = sx(stack_id) - 1
    pop = Stack(stack_id, sx(stack_id))
End Function
End Class
```

Public Class Start

```
Inherits System.Windows.Forms.Form
#Region " Windows Form Designer generated code "
Public Sub New()
    MyBase.New()
    'This call is required by the Windows Form Designer.
    InitializeComponent()
    'Add any initialization after the InitializeComponent() call
End Sub
```

'Form overrides dispose to clean up the component list.

```
Protected Overrides Sub Dispose(ByVal disposing As Boolean)
```

```
If disposing Then
    If Not (components Is Nothing) Then
        components.Dispose()
    End If
End If
MyBase.Dispose(disposing)
End Sub
```

'Required by the Windows Form Designer

Private components As System.ComponentModel.IContainer

'NOTE: The following procedure is required by the Windows Form
Designer

'It can be modified using the Windows Form Designer.

'Do not modify it using the code editor.

Friend WithEvents LinkLabel1 As System.Windows.Forms.LinkLabel

Friend WithEvents PictureBox1 As System.Windows.Forms.PictureBox

<System.Diagnostics.DebuggerStepThrough(> Private Sub

InitializeComponent()

```
    Dim resources As System.Resources.ResourceManager = New
System.Resources.ResourceManager(GetType(Start))
```

```
    Me.LinkLabel1 = New System.Windows.Forms.LinkLabel
```

```
    Me.PictureBox1 = New System.Windows.Forms.PictureBox
```

```
    Me.SuspendLayout()
```

```
    'LinkLabel1
```

```
    Me.LinkLabel1.Font = New System.Drawing.Font("Microsoft Sans
Serif", 20.0!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
```

```
Me.LinkLabel1.Location = New System.Drawing.Point(688, 424)
Me.LinkLabel1.Name = "LinkLabel1"
Me.LinkLabel1.Size = New System.Drawing.Size(80, 40)
Me.LinkLabel1.TabIndex = 0
Me.LinkLabel1.TabStop = True
Me.LinkLabel1.Text = "GO"
'PictureBox1
Me.PictureBox1.BackgroundImage =
CType(resources.GetObject("PictureBox1.BackgroundImage"),
System.Drawing.Image)
Me.PictureBox1.Location = New System.Drawing.Point(72, 168)
Me.PictureBox1.Name = "PictureBox1"
Me.PictureBox1.Size = New System.Drawing.Size(736, 200)
Me.PictureBox1.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.CenterImage
Me.PictureBox1.TabIndex = 1
Me.PictureBox1.TabStop = False
'Start
Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
Me.BackColor = System.Drawing.Color.LightCyan
Me.ClientSize = New System.Drawing.Size(792, 477)
Me.Controls.Add(Me.PictureBox1)
Me.Controls.Add(Me.LinkLabel1)
Me.Name = "Start"
Me.Text = "Start"
Me.WindowState =
System.Windows.Forms.FormWindowState.Maximized
```

```
Me.ResumeLayout(False)
```

```
End Sub
```

```
#End Region
```

```
Private Sub LinkLabel1_LinkClicked(ByVal sender As System.Object,  
ByVal e As System.Windows.Forms.LinkLabelLinkClickedEventArgs)
```

```
Handles LinkLabel1.LinkClicked
```

```
Dim wind As New wnd
```

```
wind.Show()
```

```
End Sub
```

```
End Class
```

```
Friend Class FSD
```

```
Sub Store(ByRef Data As Object, ByRef id As Object)
```

```
Dim b As Object
```

```
Dim dat As Object
```

```
FileOpen(1, "data.dat", OpenMode.Binary) : FileClose(1)
```

```
FileOpen(1, "data.dat", OpenMode.Input)
```

```
FileOpen(2, "temp.dat", OpenMode.Output)
```

```
Do
```

```
  If EOF(1) Then Exit Do
```

```
  Input(1, dat) : System.Windows.Forms.Application.DoEvents()
```

```
  If Mid(dat, 1, InStr(1, dat, "#") - 1) = id Then
```

```
    b = ""
```

```
    PrintLine(2, id & "#" & CSV(Data))
```

```
  Else
```

```
    PrintLine(2, dat)
```

```
End If
Loop Until EOF(1)
If b <> "*" Then
    PrintLine(2, id & "#" & CSV(Data))
End If
FileClose(1, 2)
FileCopy("temp.dat", "data.dat")
End Sub
```

```
Function Recv(ByRef id As Object) As Object
    Dim dat As Object
    FileOpen(1, "data.dat", OpenMode.Binary) : FileClose(1)
    FileOpen(1, "data.dat", OpenMode.Input)
    Do
        If EOF(1) Then Exit Do
        Input(1, dat) : System.Windows.Forms.Application.DoEvents()
        If Mid(dat, 1, InStr(1, dat, "#") - 1) = id Then
            Recv = CVS(Mid(dat, InStr(1, dat, "#") + 1, Len(dat)))
        End If
    Loop Until EOF(1)
    FileClose(1)
End Function
```

```
Function CSV(ByRef xstring As Object) As Object
    Dim X As Object
    For X = 1 To Len(xstring)
```

```
        CSV = CSV & AA(Asc(Mid(xstring, X, 1))) :  
System.Windows.Forms.Application.DoEvents()  
    Next X  
End Function
```

```
Function CVS(ByRef xstring As Object) As Object  
    Dim esv As Object  
    Dim X As Object  
    For X = 1 To Len(xstring)  
        esv = Mid(xstring, X, 3)  
        CVS = CVS & Chr(esv)  
        X = X + 2  
    Next X  
End Function
```

```
Function AA(ByRef xs As Object) As Object  
    If Len(xs) = 3 Then AA = xs  
    If Len(xs) = 2 Then AA = "0" & xs  
    If Len(xs) = 1 Then AA = "00" & xs  
    If Len(xs) = 0 Then AA = "000"  
End Function
```

```
Sub ShowIds(ByRef view As AxComctlLib.AxTreeView)  
    Dim Data As Object  
    Dim kid As Object  
    FileOpen(1, "data.dat", OpenMode.Input)  
    kid = view.Nodes.Count
```

```
view.Nodes.Add(, "K" & kid, "DeviceStorageFile Listing", , 13)
Do
    Input(1, Data) : System.Windows.Forms.Application.DoEvents()
    view.Nodes.Add("K" & kid, 4, view.Nodes.Count & Mid(Data, 1,
InStr(1, Data, "#") - 1), Mid(Data, 1, InStr(1, Data, "#") - 1), , 6)
    Loop Until EOF(1)
    FileClose(1)
End Sub
End Class
```

```
Imports System
Imports System.Runtime.InteropServices
Imports System.ComponentModel
Imports System.Drawing
Imports System.Convert
Imports AxInetCtlsObjects
'Imports Axmscomctl
Imports weblinks.HtmlStrip
Imports weblinks.Lc2kInternetPageSupport
Imports weblinks.Lc2kRunModule
Imports weblinks.Lc2kSupport1
Imports weblinks.Lc2kSupport2
Imports weblinks.ModCMLchecking
Imports weblinks.W32LIB_Stack
Imports VB = Microsoft.VisualBasic
Public Class wnd
    Inherits System.Windows.Forms.Form
```

#Region " Windows Form Designer generated code "

Public Sub New()

MyBase.New()

'This call is required by the Windows Form Designer.

InitializeComponent()

'Add any initialization after the InitializeComponent() call

End Sub

'Form overrides dispose to clean up the component list.

Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)

If disposing Then

If Not (components Is Nothing) Then

components.Dispose()

End If

End If

MyBase.Dispose(disposing)

End Sub

'Required by the Windows Form Designer

Private components As System.ComponentModel.IContainer

'NOTE: The following procedure is required by the Windows Form
Designer

'It can be modified using the Windows Form Designer.

'Do not modify it using the code editor.

Friend WithEvents btnstart As System.Windows.Forms.Button

Friend WithEvents btnfavorites As System.Windows.Forms.Button


```

Friend WithEvents btnhistory As System.Windows.Forms.Button
Friend WithEvents btnstop As System.Windows.Forms.Button
Friend WithEvents txtamn As System.Windows.Forms.TextBox
Friend WithEvents txtaurl As System.Windows.Forms.TextBox
Friend WithEvents txtexclude As System.Windows.Forms.TextBox
Friend WithEvents btnrestart As System.Windows.Forms.Button
Friend WithEvents btngeturl As System.Windows.Forms.Button
'Friend WithEvents ImageList1 As Axmscomctl.AxImageList
Friend WithEvents inet As AxInetCtrlsObjects.AxInet
Friend WithEvents View As AxComctlLib.AxTreeView
Friend WithEvents ImageList As AxComctlLib.AxImageList
Friend WithEvents Button1 As System.Windows.Forms.Button
<System.Diagnostics.DebuggerStepThrough(> Private Sub
InitializeComponent()
    Dim resources As System.Resources.ResourceManager = New
System.Resources.ResourceManager(GetType(wnd))
    Me.btnstart = New System.Windows.Forms.Button
    Me.btngeturl = New System.Windows.Forms.Button
    Me.btnfavorites = New System.Windows.Forms.Button
    Me.btnhistory = New System.Windows.Forms.Button
    Me.btnrestart = New System.Windows.Forms.Button
    Me.btnstop = New System.Windows.Forms.Button
    Me.txtamn = New System.Windows.Forms.TextBox
    Me.txtaurl = New System.Windows.Forms.TextBox
    Me.txtexclude = New System.Windows.Forms.TextBox
    Me.inet = New AxInetCtrlsObjects.AxInet
    Me.View = New AxComctlLib.AxTreeView

```

```

Me.ImageList = New AxComctlLib.AxImageList
Me.Button1 = New System.Windows.Forms.Button
CType(Me.inet,
System.ComponentModel.ISupportInitialize).BeginInit()
CType(Me.View,
System.ComponentModel.ISupportInitialize).BeginInit()
CType(Me.ImageList,
System.ComponentModel.ISupportInitialize).BeginInit()
Me.SuspendLayout()
'btnstart
Me.btnstart.BackColor = System.Drawing.Color.LightCyan
Me.btnstart.Font = New System.Drawing.Font("Microsoft Sans Serif",
8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.btnstart.Location = New System.Drawing.Point(96, 32)
Me.btnstart.Name = "btnstart"
Me.btnstart.Size = New System.Drawing.Size(75, 24)
Me.btnstart.TabIndex = 0
Me.btnstart.Text = "Start"
'btngeturl
Me.btngeturl.Enabled = False
Me.btngeturl.Font = New System.Drawing.Font("Microsoft Sans Serif",
8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.btngeturl.Location = New System.Drawing.Point(240, 208)
Me.btngeturl.Name = "btngeturl"
Me.btngeturl.Size = New System.Drawing.Size(104, 32)

```

```
Me.btngeturl.TabIndex = 2
```

```
Me.btngeturl.Text = "Get i.explore url"
```

```
Me.btngeturl.Visible = False
```

```
'btnfavorites
```

```
Me.btnfavorites.Font = New System.Drawing.Font("Microsoft Sans  
Serif", 8.25!, System.Drawing.FontStyle.Regular,  
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
```

```
Me.btnfavorites.Location = New System.Drawing.Point(344, 32)
```

```
Me.btnfavorites.Name = "btnfavorites"
```

```
Me.btnfavorites.Size = New System.Drawing.Size(80, 24)
```

```
Me.btnfavorites.TabIndex = 3
```

```
Me.btnfavorites.Text = "Favourites"
```

```
'btnhistory
```

```
Me.btnhistory.Font = New System.Drawing.Font("Microsoft Sans  
Serif", 8.25!, System.Drawing.FontStyle.Regular,  
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
```

```
Me.btnhistory.Location = New System.Drawing.Point(256, 32)
```

```
Me.btnhistory.Name = "btnhistory"
```

```
Me.btnhistory.Size = New System.Drawing.Size(75, 24)
```

```
Me.btnhistory.TabIndex = 4
```

```
Me.btnhistory.Text = "History"
```

```
'btnrestart
```

```
Me.btnrestart.Font = New System.Drawing.Font("Microsoft Sans Serif",  
8.25!, System.Drawing.FontStyle.Regular,  
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
```

```
Me.btnrestart.Location = New System.Drawing.Point(256, 32)
```

```
Me.btnrestart.Name = "btnrestart"
```

```
Me.btnrestart.Size = New System.Drawing.Size(75, 24)
Me.btnrestart.TabIndex = 5
Me.btnrestart.Text = "Restart"
'btnstop
Me.btnstop.Font = New System.Drawing.Font("Microsoft Sans Serif",
8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.btnstop.Location = New System.Drawing.Point(176, 32)
Me.btnstop.Name = "btnstop"
Me.btnstop.Size = New System.Drawing.Size(75, 24)
Me.btnstop.TabIndex = 6
Me.btnstop.Text = "Stop"
'txtamn
Me.txtamn.Location = New System.Drawing.Point(616, 152)
Me.txtamn.Name = "txtamn"
Me.txtamn.TabIndex = 7
Me.txtamn.Text = "100000"
Me.txtamn.Visible = False
'txtaurl
Me.txtaurl.Font = New System.Drawing.Font("Microsoft Sans Serif",
8.25!, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
Me.txtaurl.Location = New System.Drawing.Point(176, 80)
Me.txtaurl.Name = "txtaurl"
Me.txtaurl.Size = New System.Drawing.Size(328, 20)
Me.txtaurl.TabIndex = 8
Me.txtaurl.Text = ""
```

```
'txtexclude
    Me.txtexclude.Font = New System.Drawing.Font("Microsoft Sans
Serif", 8.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
    Me.txtexclude.Location = New System.Drawing.Point(512, 152)
    Me.txtexclude.Name = "txtexclude"
    Me.txtexclude.TabIndex = 9
    Me.txtexclude.Text = "exclude"
    Me.txtexclude.Visible = False
'inet
    Me.inet.Enabled = True
    Me.inet.Location = New System.Drawing.Point(96, 416)
    Me.inet.Name = "inet"
    Me.inet.OcxState = CType(resources.GetObject("inet.OcxState"),
System.Windows.Forms.AxHost.State)
    Me.inet.Size = New System.Drawing.Size(38, 38)
    Me.inet.TabIndex = 10
'View
    Me.View.Location = New System.Drawing.Point(8, 120)
    Me.View.Name = "View"
    Me.View.OcxState = CType(resources.GetObject("View.OcxState"),
System.Windows.Forms.AxHost.State)
    Me.View.Size = New System.Drawing.Size(744, 392)
    Me.View.TabIndex = 11
'ImageList
    Me.ImageList.Enabled = True
    Me.ImageList.Location = New System.Drawing.Point(176, 408)
```

```
Me.ImageList.Name = "ImageList"
Me.ImageList.OcxState =
CType(resources.GetObject("ImageList.OcxState"),
System.Windows.Forms.AxHost.State)
Me.ImageList.Size = New System.Drawing.Size(38, 38)
Me.ImageList.TabIndex = 16
'Button1
Me.Button1.Font = New System.Drawing.Font("Microsoft Sans Serif",
8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Button1.Location = New System.Drawing.Point(440, 32)
Me.Button1.Name = "Button1"
Me.Button1.Size = New System.Drawing.Size(80, 24)
Me.Button1.TabIndex = 17
Me.Button1.Text = "Exit"
'wnd
Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
Me.BackColor = System.Drawing.Color.LightCyan
Me.ClientSize = New System.Drawing.Size(768, 525)
Me.Controls.Add(Me.btnhistory)
Me.Controls.Add(Me.Button1)
Me.Controls.Add(Me.txtaur1)
Me.Controls.Add(Me.btnstop)
Me.Controls.Add(Me.btnrestart)
Me.Controls.Add(Me.btnfavorites)
Me.Controls.Add(Me.btnstart)
Me.Controls.Add(Me.ImageList)
```

```

Me.Controls.Add(Me.inet)
Me.Controls.Add(Me.View)
Me.Controls.Add(Me.btngeturl)
Me.Controls.Add(Me.txtamn)
Me.Controls.Add(Me.txtexclude)
Me.Icon = CType(resources.GetObject("$this.Icon"),
System.Drawing.Icon)
Me.MaximizeBox = False
Me.MinimizeBox = False
Me.Name = "wnd"
Me.Text = "Link Finder"
Me.WindowState =
System.Windows.Forms.FormWindowState.Maximized
CType(Me.inet, System.ComponentModel.ISupportInitialize).EndInit()
CType(Me.View, System.ComponentModel.ISupportInitialize).EndInit()
CType(Me.ImageList,
System.ComponentModel.ISupportInitialize).EndInit()
Me.ResumeLayout(False)

End Sub

Private Shared m_vb6FormDefInstance As wnd
Private Shared m_InitializingDefInstance As Boolean
Public Shared Property DefInstance() As wnd
    Get
        If m_vb6FormDefInstance Is Nothing OrElse
m_vb6FormDefInstance.IsDisposed Then
            m_InitializingDefInstance = True

```

```
        m_vb6FormDefInstance = New wnd
        m_InitializingDefInstance = False
    End If
    DefInstance = m_vb6FormDefInstance
End Get
Set(ByVal Value As wnd)
    m_vb6FormDefInstance = Value
End Set
End Property
#End Region
```

REFERENCES

REFERENCES

1. K. Bharat, A. Broder, M. Hen zinger, P. Kumar, and S. Venkatasubramanian, "The Connectivity Server: Fast Access to Linkage Information on the Web," *Proc. Seventh Int'l World Wide Web Conf.*, pp. 469-477, 1998.
2. K. Bharat and M. Hen zinger, "Improved Algorithms for Topic Distillation in a Hyper linked Environment," *Proc. 21st Int'l ACM Conf. Research and Development in Information Retrieval*, pp. 104-111, 1998.
3. S. Brin and L. Page, "The Anatomy of a Large-Scale Hyper textual Web Search Engine," *Proc. Seventh Int'l World Wide Web Conf.*, Apr 1998.
4. L.A. Carr, W. Hall, and S. Hitchcock, "Link Services or Link Agents?" *Proc. Ninth ACM Conf. Hypertext and Hypermedia*.
5. S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan, "Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text," *Proc. Seventh Int'l World Wide Web Conf.*, pp. 65-74, 1998.
6. AltaVista search engine, <http://www.altavista.com/>, 2003.