



“THESIS”

WEB CONTENT MANAGEMENT ENHANCED WITH

LINK SEMANTICS

A PROJECT REPORT

Submitted by

AARTHI GURURAJ(71201104001)

JAISHREE.J(71201104016)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI 600025

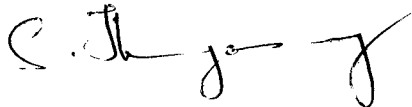
APRIL 2005



ANNA UNIVERSITY: CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this project report “**THESUS -WEB CONTENT MANAGEMENT USING LINK SEMANTICS**” is the bonafide work of “**AARTHI GURURAJ (71201104001) and JAISHREE.J (71201104016)**” who carried out the project work under my supervision.



SIGNATURE

Dr.S.Thangasamy

HEAD OF THE DEPARTMENT

Department of Computer Science & Engg.
Kumaraguru College of Technology,
Chinnavedapatti Post,
Coimbatore-641006



SIGNATURE

Mrs.D.Chandrakala

SUPERVISOR

Senior Lecturer

Department of Computer Science & Engg.
Kumaraguru College of technology,
Chinnavedapatti Post,
Coimbatore-641006

Submitted for viva-voce examination held on 21.04.2005



INTERNAL EXAMINER



EXTERNAL EXAMINER

ANNA UNIVERSITY : CHENNAI 600025

EVALUATION CERTIFICATE

College : **KUMARAGURU COLLEGE OF TECHNOLOGY**

Branch : **COMPUTER SCIENCE AND ENGINEERING**

Semester : **EIGHT (08)**

| S.NO | Name of the Student | Title of the Project | Name of Supervisor |
|-------------|----------------------------|--|---------------------------|
| 1. | Aarthi Gururaj | THESUS-Web | Mrs. D. Chandrakala |
| 2. | J.Jaishree | Content Management Enhanced with Link Semantics | |

The report of the project work submitted by the above student in partial fulfillment for the award of BACHELOR OF ENGINEERING degree in COMPUTER SCIENCE AND ENGINEERING of Anna University was evaluated and confirmed to be the report of the work done by the above student and then evaluated.

(INTERNAL EXAMINER)

(EXTERNAL EXAMINER)

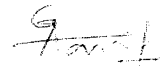
DECLARATION

We hereby declare that the project entitled “**THESUS, Web Content Management Enhanced with Link Semantics**” is a record of original work done by us and to the best of our knowledge, a similar work has not been submitted to Anna University or any Institutions, for fulfillment of the requirement of the course study.

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

Place: Coimbatore

Date: 13/04/2005



(Aarthi Gururaj)



(Jaishree.J)

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

With profound gratitude, we express our deepest thanks to Mrs.D.Chandrakala, Senior Lecturer, Dept. of Computer Science and Engineering, our internal guide, who has taken all measures to guide us through the project, and been a constant source of inspiration and motivation at various levels of the project.

Our sincere thanks to all the lab assistants who have been operational in aiding us implement the system.

We would like to thank the Head, Computer Science and Engineering, Dr.S.Thangasamy and Mrs. Devaki for guiding us through the project.

Our sincere thanks to the Department of Computer Science and Engineering, Kumaraguru College of Technology, for extending its fullest support by all means to enable us to complete the project.

Last but not the least, we extend our gratitude to all the student peers, and all those who directly or indirectly helped us in successful completion of the project.

ABSTRACT

ABSTRACT

With the unstoppable growth of the World Wide Web, users now turn to the Web whenever they need information. Currently most search features are based on raw lexical content and provide number of irrelevant results. We show how the use of the hyperlinks to a page can be used efficiently to classify a page in a concise manner. This enhances the browsing and querying of web pages, thereby eliminating irrelevant results. The links are processed using a hierarchy of concepts, akin to ontology. Information about the semantics of a page is derived by analyzing the links pointing to the given page. Thematic subsets of World Wide Web documents are constructed based on the initial set of keywords and hence the name “TheSus”. Querying process is restricted to the Music domain only and the ontology is also based on this domain. THESUS is not a search engine. Yet the functions of TheSus simulate few concepts that any search engine can implement in order to be more efficient. The entire process is executed offline.

TABLE OF CONTENTS

| CHAPTER NO. | TITLE | PAGE NO. |
|-------------|--------------------------------|----------|
| | ACKNOWLEDGEMENT | iii |
| | ABSTRACT | iv |
| | LIST OF FIGURES | vii |
| | LIST OF SYMBOLS | viii |
| 1. | INTRODUCTION | 1 |
| | 1.1 LINKAGE INFORMATION | 2 |
| | 1.2 INFORMATION RETRIEVAL | 2 |
| | 1.2.1 Keyword based Retrieval | 3 |
| 2. | LITERATURE REVIEW | 4 |
| | 2.1 EXISTING SYSTEM | 4 |
| | 2.2 PROPOSED SYSTEM | 5 |
| 3. | PROPOSED LINE OF ATTACK | 6 |
| 4. | PROGRAMMING ENVIRONMENT | 7 |
| | 4.1 HARDWARE REQUIREMENTS | 7 |
| | 4.2 SOFTWARE REQUIREMENTS | 7 |
| 5. | SYSTEM DESIGN | 8 |
| | 5.1 FLOW OF CONTROL | 8 |
| | 5.2 DATABASE DESIGN | 9 |

TABLE OF CONTENTS

| CHAPTER NO. | TITLE | PAGE NO. |
|--------------------|-----------------------------|-----------------|
| 6. | DETAILED DESIGN | 10 |
| | 6.1 INFORMATION ACQUISITION | 11 |
| | 6.2 INFORMATION EXTRACTION | 12 |
| | 6.3 INFORMATION ENHANCEMENT | 13 |
| | 6.4 DISPLAY MODULE | 14 |
| | 6.5 REGISTRATION MODULE | 15 |
| 7. | TESTING | 16 |
| 8. | FUTURE ENHANCEMENTS | 18 |
| 9. | CONCLUSION | 19 |
| 10. | APPENDIX | 20 |
| | 10.1 SAMPLE CODE | 20 |
| | 10.2 SAMPLE OUTPUT | 34 |
| 11. | REFERENCE | 37 |

LIST OF FIGURES

LIST OF FIGURES

- System Overview
- Illustration: TheSus System Architecture
- Illustration: Information Acquisition
- Illustration: Information Extraction
- Sample Ontology
- Illustration: Registration

LIST OF SYMBOLS

LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE

- **TheSus** - Thematic Subsets
- **Ontology** - List of keywords related to a domain
- **Link Semantics** - Analyses the semantic contents of link
- **Link Analysis** - using links for analysis of given keyword
- **Link Management** - managing the contents of page using link
- **Information retrieval** - retrieving information from text document
- **IE** - Information Extraction
- **IA** - Information Acquisition
- **Ont** - Ontology
- **DB** - Database

INTRODUCTION

1. INTRODUCTION

The World Wide Web serves as a huge, widely distributed, global information service center for news, advertisements, and other information. The Web also contains rich and dynamic content of hyperlink information that prove to be a useful resource. However, some of the discrepancies that are met while using the web include

- The Web seems to be huge
- The complexity of web pages is far greater than that of any large text document collection
- The Web is a highly dynamic information source
- Only a small portion of the information on the web is truly relevant or useful.

Hence we need to go in for “*Authoritative Web Pages*”.

The Authoritative Web pages are those whose contents are highly related and have authority over the context covered. It is found that the secrecy of “authority” is found in the “*Web page linkages*”.

Sometimes many words are introduced in the content irrespective of the context under which the page is based. The concept proposed in TheSus eliminates this irrelevancy to a certain extent, aiding for a more concise and theme based search, thereby reducing the time a user spends in identifying the URLs relevant to him/her.

1.1 Using Linkage Information:

The Web consists of not only pages, but also of hyperlinks pointing from one document to another. These hyperlinks contain enormous amount of “latent human annotation” that can help to infer the notion of authority. The tremendous amount of Web linkage information provides rich information about the relevance, quality, and the structure of the Web’s content, and thus is a rich source of information.

TheSus illustrates this concept that can be used by search engine where links are considered a more useful resource for finding the authoritative pages rather than content of the page.

The TheSus incorporates the concept of Text-Data Analysis, which uses the Information Retrieval mechanism.

1.2 Information Retrieval:

Information Retrieval is the organization and retrieval of information from a large number of text based documents. A typical information retrieval problem is to locate relevant documents based on user input such as keyword, queries or example documents.

1.2.1 Keyword Based Retrieval:

In a keyword based retrieval mechanism, a document is represented by a string, which can be identified by a set of keywords. A good retrieval method considers the synonyms of the words while answering such queries. For example, for the given string “**car**”, synonyms such as automobile, vehicle should also be considered in the search.

The two types of problems encountered in Keyword based retrieval are

- Synonymy problem
- Polysemy problem

Synonymy problem is such that the given keyword may not appear anywhere in a document but the document may be closely related to that particular context.

Polysemy problem is that the same keyword may mean different things in different contexts.

These two problems are addressed via the use of “**ONTOLOGY**” that has been proposed.

LITERATURE REVIEW

2.1 LITERATURE REVIEW

2.1 Existing System:

The existing concept of web search engines uses the crawling system to move from one page to another and searches the content of the pages for relevance to any given query.

Once any match to the given query is found, they are updated to the database and the matches are ranked based on the Page Rank Algorithm. The results are displayed to the user. Sometimes in order to increase the demand for a page, creators of the page introduce certain words that are irrelevant to the context of the web document yet might seem important.

For example, when searching for the keyword “wind”, The user may mean the wind under different contexts like weather, wind instrument, windpipe and so on. Unless the user is very specific in his query he cannot find an exact option.

The appearance of such redundant keywords under irrelevant context wastes time of those users who are specific to any particular domain.

The redundancies also mislead the naïve users and also those who go in for information about a particular topic of interest.

2.2 Proposed System:

In order to eliminate the irrelevancy caused from the presence of such words in a particular domain, we propose two new concepts for refining the search process.

- One is the use of hyperlinks while matching for the given query.
- Second is the use of “Ontology”.

Under the presumption that hyperlinks contain information related to the context of the page, we go in for extracting words from the hyperlinks and matching them with the given query. The hyperlinks and its neighboring areas contain information relevant to the pages content. Hence we make use of this to reduce redundancy.

The use of Ontology relates a given keyword to one particular domain. The ontology contains words associated to a particular domain. If the keyword matches the words in the ontology, then files relating to the ontology domain are alone extracted. The remaining ones are discarded.

By implementing these two concepts, we propose to eliminate the irrelevant URLs, making the search process more concise and meaningful!

PROPOSED LINE OF ATTACK

3. PROPOSED LINE OF ATTACK

In order to eliminate the extraction of irrelevant URLs from misleading data, we intend to approach the TheSus System by splitting it into various modules, such as Information Extraction, Information Acquisition, Information Enhancement that focus on domain related querying.

Each of these modules performs its assigned functions of querying for keywords and URLs and finally displays the result ranked by the importance of data using Page Rank Algorithm.

The platform on which the TheSus system is implemented is Windows XP, with source coding done using **JAVA** and the front-end implementations using **JSP**. **JSP** pages incorporate the HTML content also, thereby aiding in the generation of dynamic pages as and when the queries are processed.

Client side validations are implemented using **Java Script** for java is platform independent and hence the system can be run on any platform. Database used is **MS-Access**. **Access** has its own advantage such that it loads only those components that are required for a particular application under concern.

PROGRAMMING ENVIRONMENT

4. PROGRAMMING ENVIRONMENT

4.1 Hardware Requirements:

Operating System: Windows 2000 and above versions

RAM: 64 MB

Processor: Pentium IV

4.2 Software Requirements:

- Jdk1.3 or its higher version
- J2sdk1.4
- Apache Tomcat Web Server
- Web browser
- MS-Access Database

SYSTEM DESIGN

5. FEM DESIGN

5.1 Flow of control:

- The TheSus system accepts users input through the User Interface that is designed similar to that of any search engine.
- A query string is accepted from the user and passed to the Information Acquisition module.
- This module queries through the pages that are available in the database.
- Every page is mapped with the user given keyword.
- If a match to the specific string is found, the page is returned to the next module namely, Information Extraction.
- The Information Extraction module searches the hyperlinks for a given user query.
- Since the hyperlinks are a source of information on the authority of the page, the presence of the given keyword in the hyperlink places that page above all the other pages.
- The URL of the page is passed to the Display module.
- If no match is found in the content or in the hyperlink, the user query is passed to the Information Enhancement module.
- This module makes use of ontology to find the synonyms of given keyword or its related terms.
- The Web pages are again searched for the keyword generated.
- If a match is found, the URLs of these documents are placed in the display module.
- The display module is accessed finally to generate all the results along with the count of the number of results found.

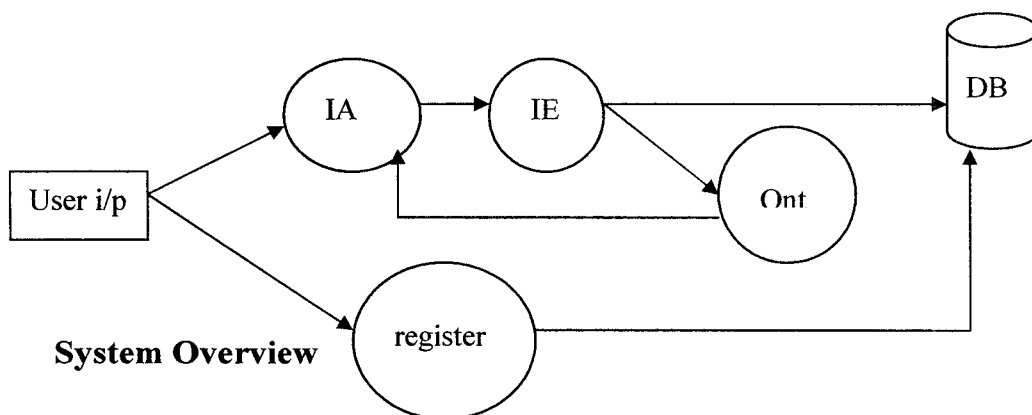
5.2 Database Design:

The database consists of 5 tables namely

- ThesusFileName
- ThesusFile URL
- DisplayURL
- Ontology
- Register

The first two tables act as an index to enable easier location of Web pages. They contain the list of Web pages and the URL that points to the Web page. The DisplayURL table houses the URLs of those pages that have some authority over the requested document. The Ontology table consists of that list of root keywords that may appear in any of the documents. This consists of some of frequently occurring synonyms of a particular domain.

The Registration module facilitates the users to register new URLs, the organization to which it is associated with, and the details of the customer who registers the details with the system.



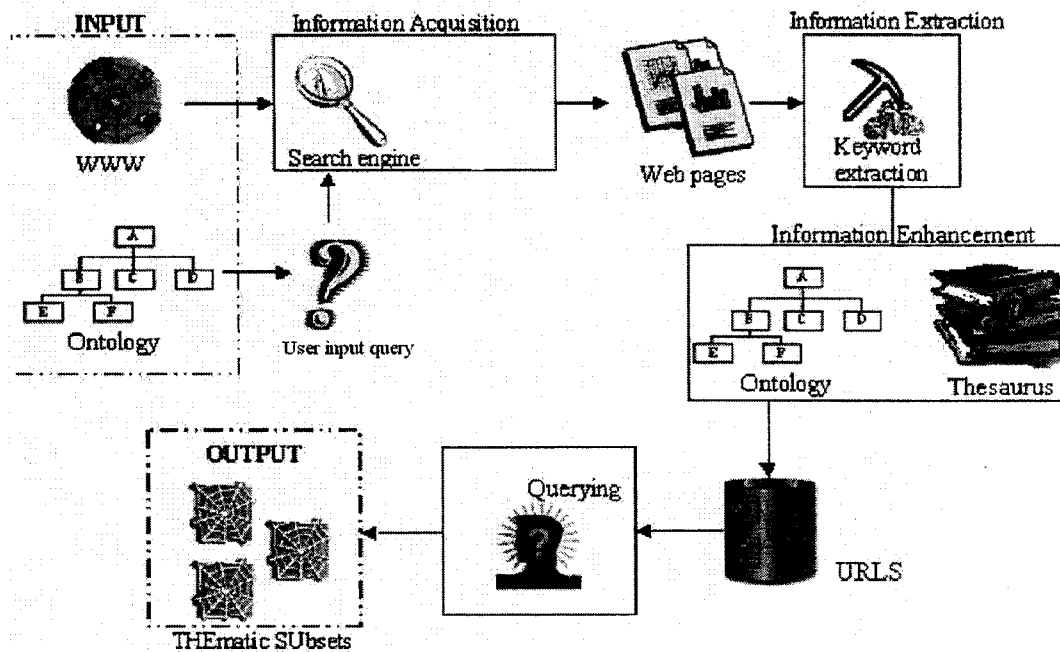
DETAILED DESIGN

6 DETAILED DESIGN

The entire system of TheSus is divided into modules namely,

- Information Acquisition
- Information Extraction
- Information Enhancement
- Querying
- Display module
- Registration module

Illustration: TheSus System Architecture



Lets see each of these modules in detail.

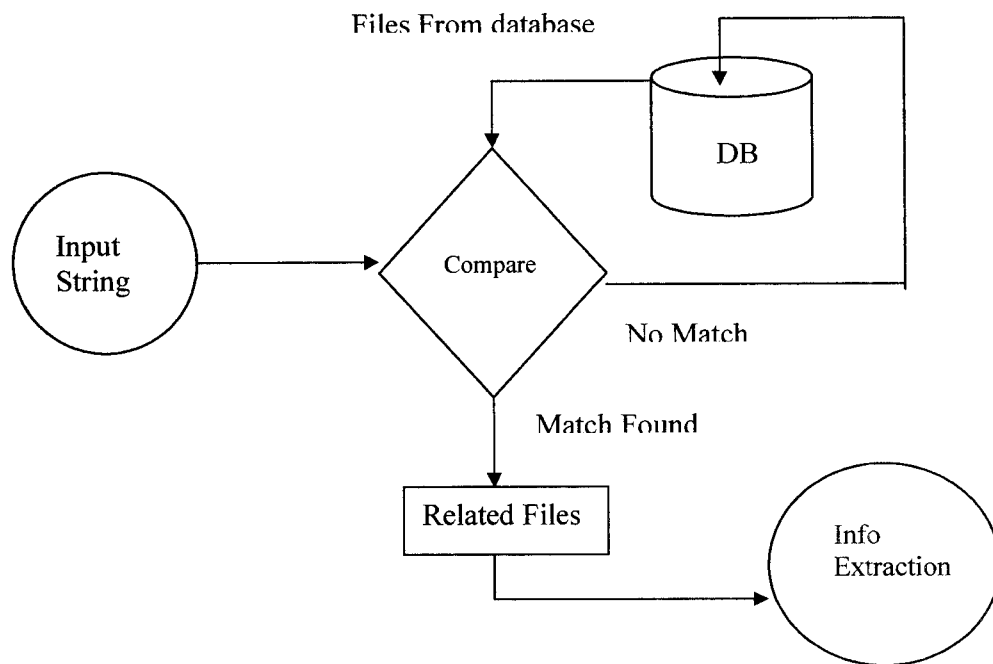
6.1 Information Acquisition:

The keyword for which the user wants relevant search result is passed from the client side JSP page to this module.

Once the keyword is obtained, the files in the database are searched for the presence of keyword. Searching the file includes reading the file using a File reader and then using “Stream Tokenizer” to split the file and compare with the keyword.

Once the required word is found, all the files that contain the keyword are passed to the Information Extraction module.

Illustration: Information Acquisition



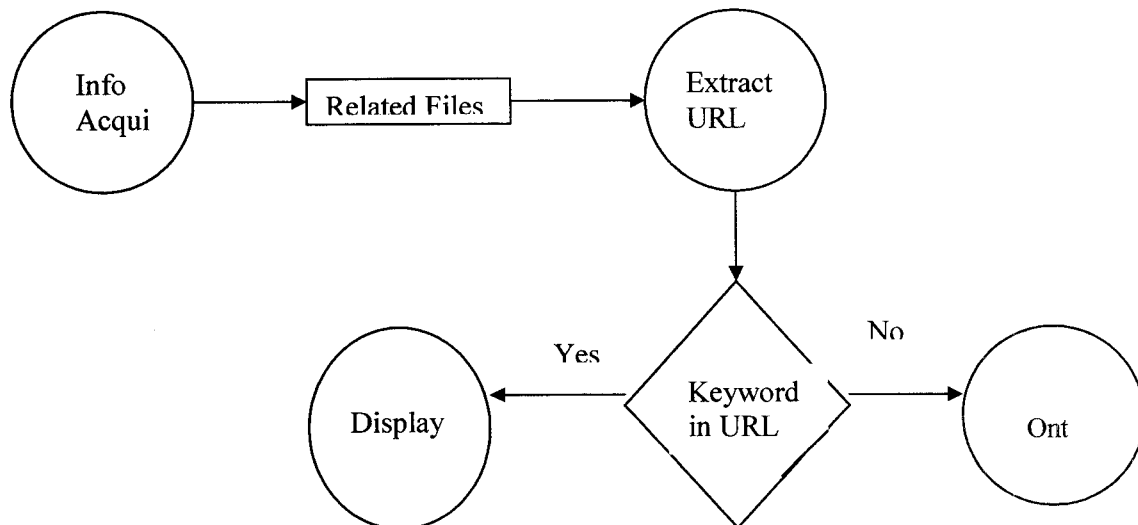
6.2 Information Extraction:

For those files that contain the required keyword, the next step is to extract the URLs from them. The incoming and outgoing links are taken and parsed with the help of “String Tokenizer”. The URLs are checked for the presence of the query string. This process is one of the enhancements of the proposed system. To check the presence of keyword in the URL we find the position of the starting letter of the keyword in the URL and extract the sub string thereon.

If the sub string matches with that of the query string, the URL is passed on to the display module.

If no match is found for the keyword in the content of the web pages or in the URL, we pass on the keyword to the Ontology, implemented in the Information enhancement module.

Illustration: Information Extraction



6.3 Information Enhancement:

The Information Enhancement module comes into play when the given keyword does not occur in any of the pages. This module makes use of Ontology. Ontology defines a common vocabulary for researchers who need to share information in a domain. All the important and most sought after words are included in the ontology.

Our ontology mainly focuses on the “music” domain and contains almost all words related to this domain. We develop this ontology as an XML file because we can define tags of our own in xml files.

This module parses the Ontology xml file using Stream Tokenizer. The companion tags form the root. If a match to the query string is found in the root, then that word and its similar ones under it are searched for a match in the pages and the corresponding URLs are returned to the display module.

If a match is found in the tags following the root, then we search for the presence of the root word in the contents. If no match is found, we return a null to the display module.

Sample Ontology:

```
- <media>
- <sound>
- <digital>
  <cds />
- <computer>
  <mp3 />
  <midi />
  </computer>
</digital>
- <analog>
  <vinyl />
  <tape />
  </analog>
</sound>
- <image>
  <photographs />
```

```
<pop>
<disco />
<grunge />
<reggae />
<techno />
<rock />
</pop>
- <world>
  <indian />
  <arabic />
  <chinese />
- <american>
  <country />
  </american>
  <french />
  <folk />
</style>
```

This ontology module aids in reducing the irrelevancy and concentrates to a particular domain thereby yielding better search results. The use of ontology to enhance the search process is yet another steps forward in implementing domain related search!

6.4 Display Module:

The display module consists of extracting the URLs that are found relevant to the context. The display module consists of database access where all the URLs are stored. Using the “connection” object creates database access prior to which the MS-Access driver is initialized.

Query statements are created and the result set is passed to the UI, Display page and displayed to the user.

If the display module accesses a null database, we forward the UI age to intimate that no results have been found for the particular word.

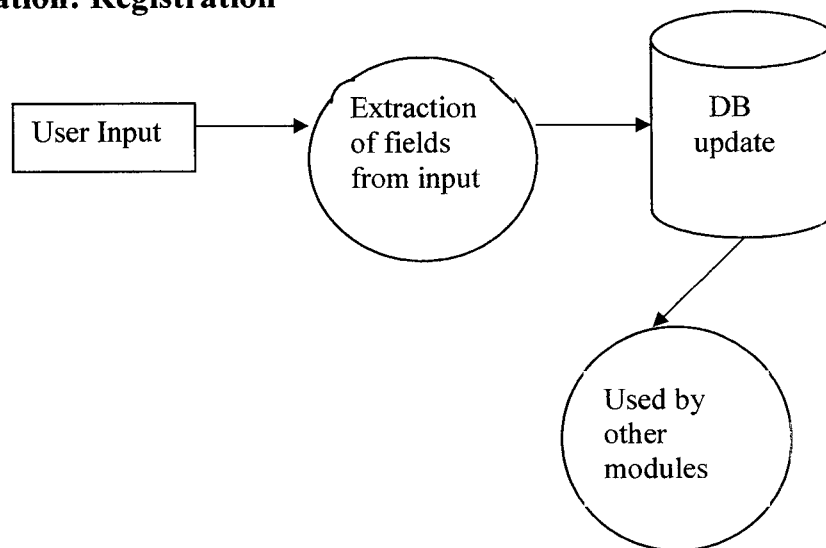
6.5 Registration module:

This module enables the user to register new URLs with the TheSus system so that they can be included in the search process. The details such as Customer name, Organization name, the URL registered and the index file are extracted from the users input page.

The URL and the index file are updated to the respective tables so that they can be used at later stages of searching. The customers details are updated to the Customer table so that further enquires are possible.

This module does much of database access. A connection is established using the connection object. Query to update the database are written in the statement object. The value returned by the result set is checked for valid update and commit operation is performed. Otherwise the operations are rolled back and user is prompted with the message of invalid update.

Illustration: Registration



TESTING

7. TESTING

Testing ensures that all the discrepancies in the system are identified and corrected so that one can expect efficient outcome from the system. The THESUS system is tested with some sample keywords in order to check the outcome and its efficiency.

We test the system with keywords that relate to the domain as well as with those that are totally irrelevant. Since the feature of the system itself is the domain related searching, we see to that results obtained are relevant to the domain.

Testing is done based on the number files that contain the query string specified and the number of relevant files retrieved. We have for verification, some sample files in the client side of the system that simulate the Web pages.

Unit testing was done at the design stage to check the concurrency of each module. Sample inputs were given and the performance was checked by executing the program in command prompt. Let us see some of the sample inputs that the system was run on and its performance.

Input: guitar

The sample files for this particular keyword was maintained in such a way that all the simulating Web pages contained the word “guitar”.

When executed we obtained a result of only 5 results. The 5 URLs that were generated were found to contain the keyword in the URL or were mapped from the Ontology.

Input: jazz

The keyword digital generated 3 resultant URLs wherein all the 3 were found to contain the keyword “jazz”. The total number of files that contained the keyword jazz are 5. The remaining 2 pages are eliminated for that fact that they are probably not domain specific.

Input: digital

The keyword “digital” when searched for generated 5 URLs, all 5 of which did not contain the word digital. The web pages were simulated in such a way that none of them had this word in it. Since digital is a part of the domain that may mean a digital music file or digital encoding or the like, the word was mapped to the ontology and some related files are being displayed.

Input: hello

The hello keyword returned a null result thereby directing the system to No Result page. This particular word neither relates to the domain nor is present in any file.

Testing Registration:

Giving new URLs and checking the updation process in the database test the registration module. A proper update results in a pop-up dialog in the user interface that displays the confirmation of registered details. Else the pop-up displays a “ Details not Registered” dialog box.

These are a few inputs that were used to test the system for efficiency.

FUTURE ENHANCEMENTS

8. FUTURE ENHANCEMENTS

- The system shall be updated for searching using multiple keywords.
- The current system is demonstrated using single ontology only.
- The TheSus system can be enhanced by bringing in user intervention to determine the users domain of interest when given more number of ontologies, each of a particular domain.
- This gives the user complete freedom to determine how relevant his search results are going to be.
- Use of tools to generate the various senses of given keyword could be used to identify the users domain of interest easily.

These will make the system more user friendly!

CONCLUSION

9. CONCLUSION

The TheSus System proposes a concept of information retrieval from Web pages through keyword retrieval mechanism. Authoritative Web pages are found by focusing on the hyperlink information, namely Link Semantics in order to find pages that are of interest to the user. Irrelevant pages are eliminated as a result of focus on Link Semantics. Related data are not lost as a result of synonimical search.

With little bit of enhancement on querying over multiple keyword strings, the TheSus system proves to be a useful solution for knowledge hunters and domain related searchers!!!

APPENDIX

10.APPENDIX

10.1 Sample source code:

```
public ArrayList getFileName(String kw) throws IOException,
SQLException
{
    System.out.println("In class InfoAccqui: "+ kw);
    DbConnection tc=new DbConnection();
    Connection objCon=tc.getConnection();
    Statement stmt=objCon.createStatement();
    ResultSet rs=stmt.executeQuery("select FileName from
ThesusFileName");
    String fn=new String();
    ArrayList al=new ArrayList();
    boolean flag=false;
    while(rs.next())
    {
        fn=rs.getString("FileName");
        System.out.println("***** File under Check is: "+fn);
        File i=new File("C:/THESUS Proj/",fn);
        FileReader fr=new FileReader(i);
        StreamTokenizer st=new StreamTokenizer(fr);
        st.resetSyntax();
        st.wordChars(33,255);
        st.whitespaceChars(0, ' ');
        st.eolIsSignificant(true);
        String str;
        while(st.nextToken()!=StreamTokenizer.TT_EOF)
```

```

    {if(st.ttype==StreamTokenizer.TT_WORD)
    { str=st.sval;
    if(str.compareToIgnoreCase(kw)==0)
        { flag=true;
    System.out.println("Keyword Exists in this file");
    // put the file name in an arraylist.
    al.add(fn);//check for repeated file names
    }
    else
    {flag=false; }
    }//end of if
    }//end of second while
public void getURL(ArrayList al,String kw) throws IOException,
SQLException
{ System.out.println("In class InfoExtract!");
System.out.println(al.size());
DbConnection tc=new DbConnection();
try
{ Connection objCon=tc.getConnection();
Statement stmt=objCon.createStatement();
ArrayList ar=new ArrayList();
int n=0;
//delete URLs of previous transaction
Statement s=objCon.createStatement();
String delqry="delete from DisplayURL";
int z=s.executeUpdate(delqry);
// end of delete

```

```

for(int i=0;i<al.size();i++)
{
    String fn=(String) al.get(i);
System.out.println(fn);
ResultSet rs=stmt.executeQuery("select URL from ThesusFileURL
where FileName='"+fn+"'");
while(rs.next())
{
    String url=rs.getString("URL");
System.out.println("Extracted URL is:"+url);
int k=url.indexOf(kw);
//System.out.println("kkkkkkkkkkkkkkkk "+ k);
int l=url.length();
if(k!=-1)
{
// System.out.println(k+" "+kw.length());
String qrykw=url.substring(k,k+kw.length());
System.out.println("Query Keyword"+ qrykw);
if(qrykw.compareTo(kw)==0)
{
++n;
System.out.println("Keyword exists in the URL");
// write current urls to database table display
Statement s1=objCon.createStatement();
String qry="insert into DisplayURL values('"+n+"','"+url+"')";
int r=s1.executeUpdate(qry);
if(r>0)
{System.out.println("URL updated to database!");}
else
{ System.out.println("URL update Failed!");}
}
}
}

```

```

    }
    else
    { // call ontology module.return an arraylist with
keyword,filename,url.
//ar.add(new OntologySource(kw,fn,url));
System.out.println("No matches found in URL and Content!! Go in
for Ontology!!!");
    }}
if(k==-1)
{
System.out.println("Keyword not in URL");
//ar.add(new OntologySource(kw,fn,url));
}
} //end of while
} //end of for
public ArrayList getKwSenses(String kw) throws SQLException,
IOException
{ //Connection objCon=ThesusConnection.getConnection();
//Statement st=objCon.createStatement();
ArrayList sen=new ArrayList();
// OntologySource os= new OntologySource();
File os=new File("C:/THESUS Proj/SampleOntology.txt");
FileReader fr=new FileReader(os);
StreamTokenizer tok=new StreamTokenizer(fr);
tok.resetSyntax();
tok.wordChars(33,255);
tok.whitespaceChars(0, ' ');

```

```

tok.eolIsSignificant(true);
String str,start=null,temp=null,rootKw=null;
int l=0;
while(tok.nextToken()!=StreamTokenizer.TT_EOF)
if(tok.ttype==StreamTokenizer.TT_WORD)//if1
{
str=tok.sval;
System.out.println("@@@@@@@@@@: "+str);
rootKw=checkToken(str,kw,tok,rootKw);
if(flag==true)
{ break; }
} }
return sen;
} // end of method

```

```

public String checkToken(String str,String kw,StreamTokenizer
tok,String rootKw)
{
String temp=null,xKw=null;
boolean flag;
int l=str.length();
if(str.compareTo("-")==0)
{ rootKw=storeVal(str,tok);
if((rootKw.charAt(0)=='<')&&(rootKw.charAt((rootKw.length()-
1)=='>'))
{
xKw=rootKw.substring(1,(rootKw.length()-1);

```



```

int g=rootKw.length();
//rootKw=rootKw.substring(1,g-1);
rootKw=xKw;
checkResult(kw,xKw,rootKw);
}}
else if((str.charAt(0)=='<')&&(str.charAt(1-2)=='/'))
{
xKw=str.substring(1,1-2);
// int g=rootKw.length();
//rootKw=rootKw.substring(1,g-1);
checkResult(kw,xKw,rootKw);
}
else if((str.charAt(0)=='<')&&(str.charAt(1)=='/'))
{
if(str.compareTo(rootKw)==0)
{
System.out.println("match found for rootKw!");
}
}
return rootKw;
}
public void checkResult(String kw,String xKw,String rootKw)
{
boolean flag=false;
if(kw.compareTo(xKw)==0)
{
System.out.println("!!!! Keyword exists: "+xKw);
//call method to check db for that kw
System.out.println("Root Keyword is: "+rootKw);
checkDbase(rootKw);
flag=true;
}
}

```

```

}
else
{ return; }
}
public String storeVal(String str,StreamTokenizer tok)
{ String rootKw=null;
try
{ if(tok.nextToken()!=StreamTokenizer.TT_EOF)
{
rootKw=tok.sval;
System.out.println("Root Kw is: "+rootKw);
}
}
catch (IOException e)
{System.out.println("Error in storeval!");
e.printStackTrace();
}
return rootKw;    }
public void checkDbase(String rootKw)
{ OntDbaseCheck o=new OntDbaseCheck();
o.DbaseCheck(rootKw);
}
public void DbaseCheck(String rootKw)
{
System.out.println("In OntDbaseCheck: keyword is: "+rootKw);
DbConnection tc=new DbConnection();
Connection objCon=tc.getConnection();

```

```

String fn=new String();
int n=0;
try
{ Statement st=objCon.createStatement();
ResultSet rs=st.executeQuery("Select FileName from Ontology
where Keyword='"+rootKw+"'");
while(rs.next())
{
fn=rs.getString("FileName");
}
//delete URLs of previous transaction
Statement s=objCon.createStatement();
String delqry="delete from DisplayURL";
int z=s.executeUpdate(delqry);
// end of delete
Statement stmt=objCon.createStatement();
ResultSet rs1=stmt.executeQuery("Select URL from ThesusFileURL
where FileName='"+fn+"'");
while(rs1.next())
{
n=n+1;
String url=rs1.getString("URL");
Statement s1=objCon.createStatement();
String qry="insert into DisplayURL values("+n+", '"+url+"'");
int r=s1.executeUpdate(qry);
if(r>0)
{ System.out.println("URL updated to database!");

```

```
}  
else  
{ System.out.println("URL update Failed!");  
}  
}  
catch (SQLException e)  
{System.out.println("SQL Exception generated for statement  
creation!");  
e.printStackTrace();  
}  
tc.releaseConnection();  
}
```

Sample JSP Code:

```
<%@ page import="java.util.ArrayList" %>
<jsp:include page="InfoAccqui.jsp" flush="true" />
<jsp:useBean id="InfoExtract" class="testthesus.InfoExtract">
<%
    try
        { Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); }
    catch(Exception e)
        { out.println("jsp file sql exception"); }
%>
</jsp:useBean>
<html><body>
<%
    String kw=request.getParameter("txtSearch");
    ArrayList res1=request.getParameter("res");
    InfoExtract.getURL(res1,kw);
    out.println("URL extracted");
%>
</body></html>
<jsp:useBean id="Reg" class="testthesus.Register" />
<%
    try
        { Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); }
    catch(Exception e)
        { out.println(e.toString()); }
%>
```

```

<html><head>
<script>
    function pass(form)
    { alert("Details Registered !"); }
    function fail(form)
    { alert("Sorry!! Registration Failed !!"); }
</script>
</head>
<body bgcolor="#6699cc">
<form method="POST" name="Reg" action="Home.jsp">
<%
String txtURL=request.getParameter("txtURL");
String fn=request.getParameter("txtFileName");
String orgname=request.getParameter("txtOrg");
String cust=request.getParameter("txtCustomer");
int val=Reg.setDetails(txtURL,fn,cust,orgname);
    if(val==1)
    { %>
        <script>
            alert("Details Registered!!");
        </script>
    <%}
    else{ %>
        <script>
            alert("Details Not Registered!!");
        </script><%} %>

```

```

</form></body></html>
<%@ page import="java.sql.*" %>
<%! int n=0;%>
<%
    try
        {Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");    }
    catch(Exception e)
        {    out.println(e.toString());    }
%>
<HTML>
<HEAD>
<title>Result</title>
</HEAD>
<body MS_POSITIONING="GridLayout" bgColor="#6699cc">
<form method="post" name="Result">
<input type="hidden" value="3" name="TransId">
<%
    try
        { Connection con=DriverManager.getConnection("jdbc:odbc:Thesus");
          Statement stmt=con.createStatement();
          ResultSet rs=stmt.executeQuery("Select URLdisp from DisplayURL");
          while(rs.next())
            { n=n+1;
              String u=rs.getString(1);
            }
%>

<TR>

```

```

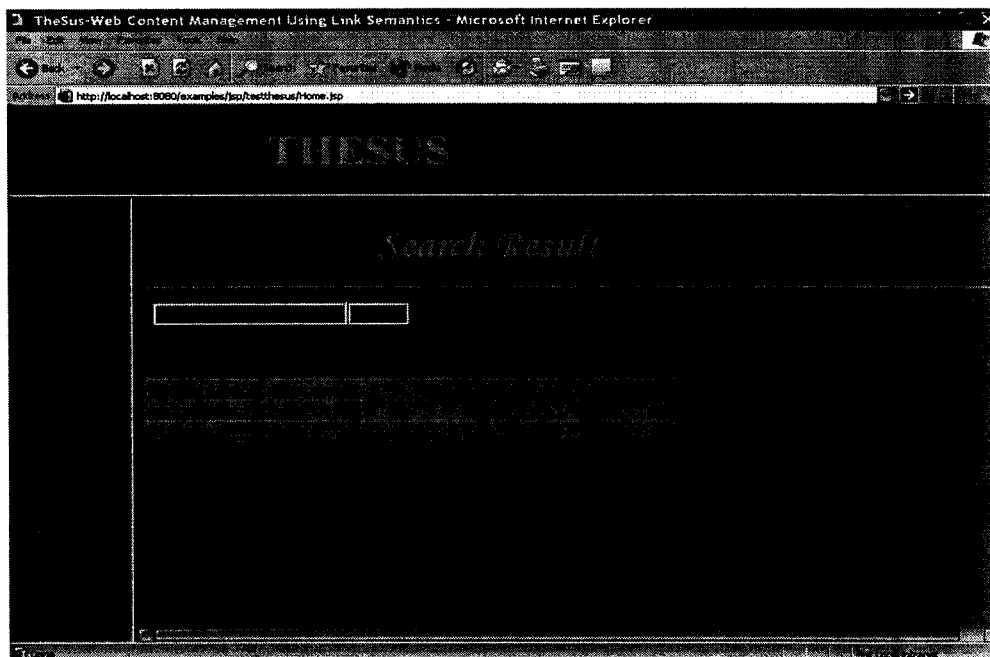
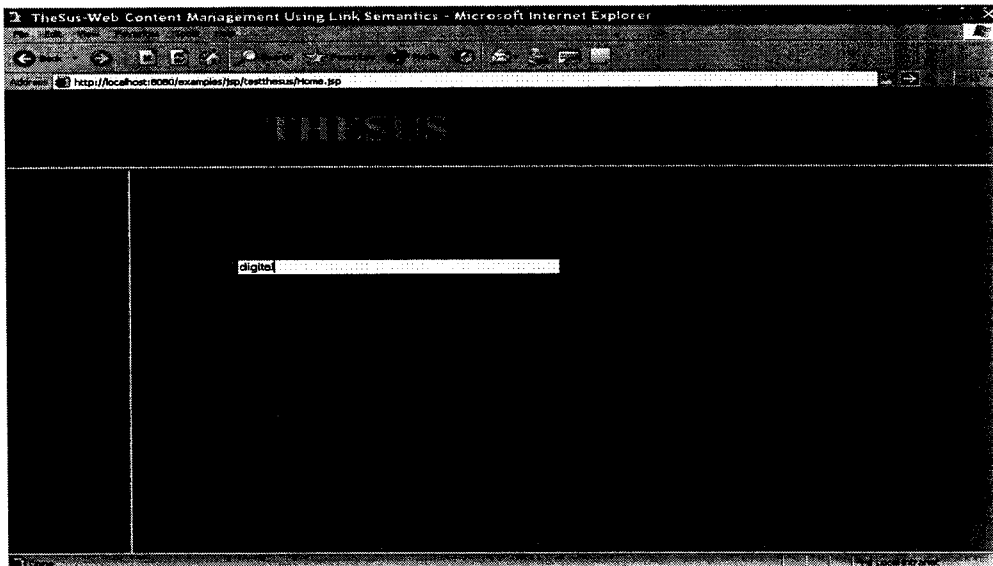
<TD bgColor="#66ccff
<A href=" "><%out.priir ln(u);%>&nbsp;</A>
</TD></TR>
<% } %>
<% }
catch(Exception e)
{
}%>
</TABLE>
</form></body></HTML>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<html>
<head>
<title>RightFrame</title>
<meta name="vs_defaultClientScript" content="JavaScript">
<meta name="ProgId" content="VisualStudio.HTML">
<meta name="Originator" content="Microsoft Visual Studio .NET 7.1">
<script>
function validate(form)
{
    if(document.Search.txtSearch.value.length==0)
    {
        alert("Enter Search text !!");
        document.Search.txtSearch.value=" ";
    }
    if(document.Search.txtSearch.value.length!=0)
    {
        alert("Form Submit");

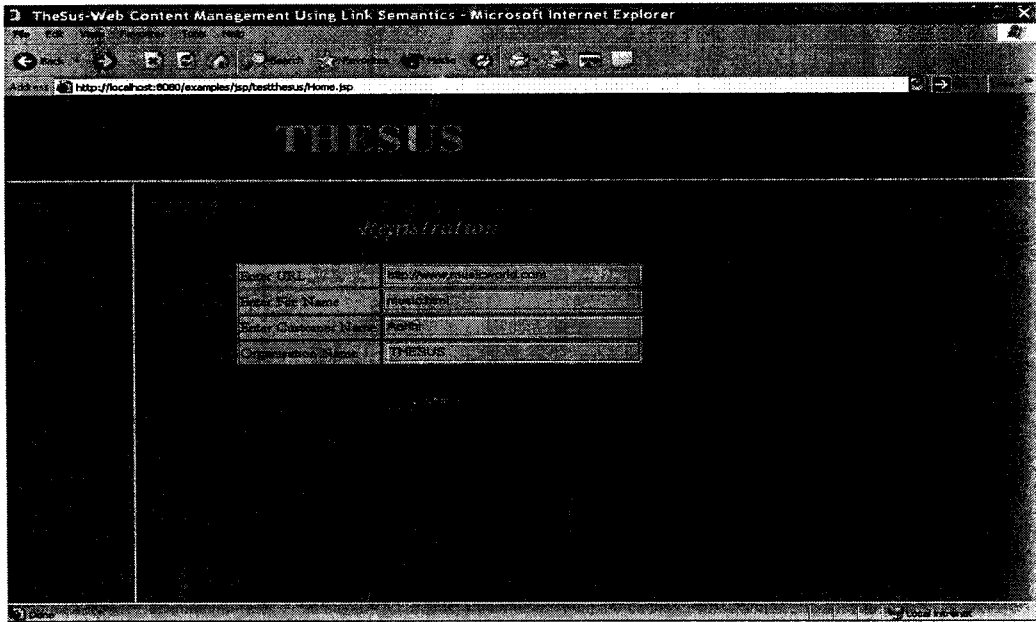
```

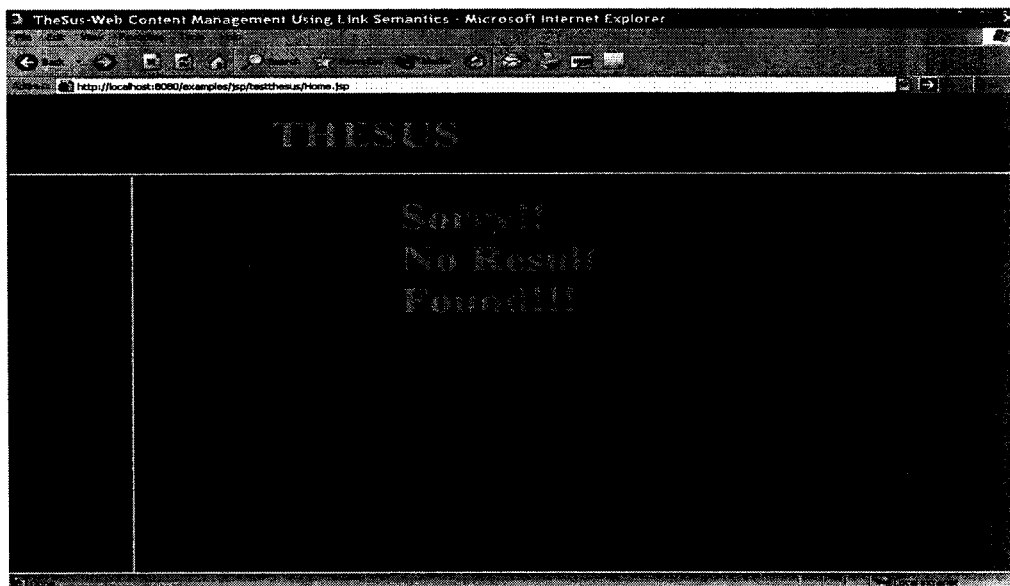
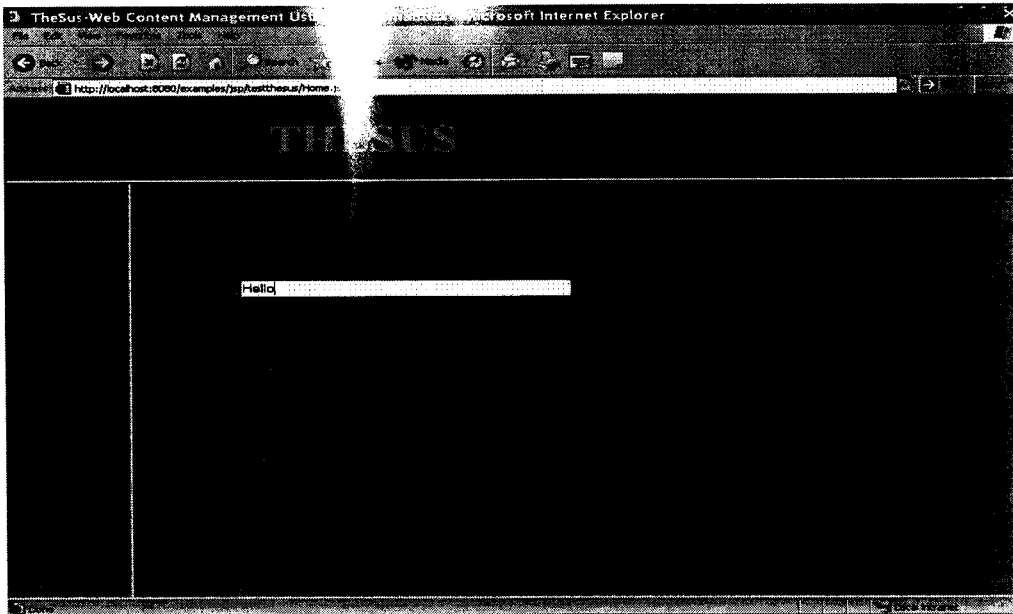


```
        form.submit();
    } }
</script>
</head>
<body MS_POSITIONING="GridLayout" bgColor="#6699cc">
<form method="post" name="Search" action="InfoAccqui.jsp">
<input type="hidden" value="1" name="TransId">
<input type="button" value="Go" name="cmdSearch"
onClick="validate(this.form);">
</form></body></html>
```

10.2 Sample Output:







REFERENCES

11.REFERENCES

1. "Mining link structure of world wide web", Computer, vol 32, no.8, pp.60-67, Aug 1999
2. GreenLaw and Hepp, 'Inline and Online Internet', Tata McGraw Hill Publication
3. M.Henzinger, "Hyperlink Analysis for the Web," IEEE Internet Computing, vol 5, no.1, pp.45-50, 2001
4. Herbert Schildt (2000), 'Java-Complete Reference', Tata McGraw Hill Publication, Fifth Edition
5. Ivar Horton, 'Beginning SDK 1.3', Wrox Publications
6. Iraklis Warlamis, Michalis vazirgiannis, Maria Halkidi, "THESUS, a closer view on Web Content Management Enhanced with Link Semantics", IEEE Transactions on Knowledge and Data Engineering, vol.16, no.6, June 2004
7. Jeff Frentzen & Henry Sabotka (1999), 'JavaScript, Annotated Archives', Tata McGraw Hill
8. Jiawei Han & Micheline Kamber (2001), 'Data Mining-Concepts & Techniques', Morgan Kaufmann Publishers
9. Margret Young (1999), 'The Internet - Complete Reference', Tata McGraw Hill Publication
10. Natlaya F. Noy and Deborah L. McGuinness, "Ontology Development 101: A Guide to creating your First Ontology", Stanford University

Online Reference:

1. <http://www.computer.org>
2. <http://www.searchenginesystems.net>
3. www.iprcom.com/papers/pagerank
4. <http://www.db-aueb.org>
5. <http://www.digital-web.com/articles/smartercontentpublishing>
6. <http://www.daml.org/ontologies/>
7. <http://www.db-net.aueb.gr/thesis>
8. <http://directory.google.com>
9. www.northernlights.com (Search Engine Sample)
10. www.vivisimo.com (Search Engine Sample)
11. www.citeseer.com
12. Word Net website – <http://www.cogsci.princeton.edu/~wn>
13. ODP – open directory Project, <http://dmoz.org/>

