

MEDICAL SEARCH ENGINE

Project Report submitted in partial fulfillment of the
requirement for the degree of
Bachelor of Technology.

in

Computer Science & Engineering

under the Supervision of

Ms Ramanpreet Kaur

By

Disha Mehta (111307)

to



Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

CERTIFICATE

This is to certify that project report entitled “Medical Search Engine”, submitted by Disha Mehta in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Supervisor’s Name

Designation

Signature

Date:

ACKNOWLEDGEMENT

The satisfaction that accompanies with the partial completion of any task would be incomplete without mention of people whose ceaseless corporation made it possible, whose constant guidance and encouragement crown all efforts with success.

I take this opportunity to express my sincere gratitude to Head of Department(CSE), Brig. S.P. Ghrera, for co-operation in extension of all necessary facilities for the conduct of my work, Ms Ramanpreet Kaur, my Project Supervisor, for her kind and patient efforts in directing my project.

I would also like to thank all those who have directly or indirectly contributed to my project. Their co-operation and support has been a vital input.

Date:

Name of the student

TABLE OF CONTENTS

Chapter No.	Topic	PageNo.
	Title Page	I
	Certificate from the Supervisor	II
	Acknowledgement	III
	Table of Contents	IV
	List of Abbreviations and Symbols	IX
	List of Figures	X
	List of Tables	XI
	Abstract	1
Chapter-1	Introduction	2
Chapter-2	History of Search Engine	4
2.1	History of Yahoo	5
2.2	History of Ask	5
2.3	History of Google	6
2.4	History of MSN Search	6
Chapter-3	Literature Review	7
3.1	Components of Search Engine	7
3.1.1	Crawler	7
3.1.2	Tokenization Process	8
3.1.3	Indexer	10

3.1.4	Search	10
3.1.5	Ranking of Result	10
3.1.6	Page Rank Algorithm	11
3.1.7	Performance of Search Engine	12
3.2	Types of Search Engine	13
3.2.1	Crawler Based	13
3.2.2	Directories	14
3.2.3	Hybrid Search Engine	14
3.2.4	Miscellaneous Types	14
Chapter-4	Tools and Techniques Used	16
4.1	Java EE Framework	16
4.1.1	Distributed Multi Tier	16
4.1.2	Application Components	17
4.1.3	Java Server Pages	18
4.1.4	Java Servlet	19
4.1.5	Java Features	20
4.2	HTML	20
4.3	MySQL Database	21
4.3.1	MySQL Features	21
4.3.2	MySQL Advantages	22
4.4	PHP	22
4.4.1	Advantages of PHP	22
4.4.2	PHP Usage	23

4.4.3	PHP for Web Development	24
Chapter-5	System Requirements	26
5.1	Requirement Analysis	26
5.1.1	Hardware Requirements	26
5.1.2	Software Requirements	26
5.1.3	Functional Requirements	27
5.1.4	Non Functional Requirements	27
Chapter- 6	Analysis	28
6.1	Introduction	28
6.1.1	Logical Design	28
6.1.2	Physical Design	28
6.2	Use Case Diagram	29
6.2.1	Actors Involved	29
6.3	Sequence Diagram	30
6.3.1	Query Processor	30
6.3.2	Indexer	31
6.4	Activity Diagram	31
6.5	Data Flow Diagram	32
6.5.1	Level-0 DFD	32
6.5.2	Level-1 DFD	33
Chapter- 7	Design and Implementation	34
7.1	Design	34
7.2	Implementation	35

7.2.1	Pseudo Code for Crawler	35
7.2.2	Pseudo Code for Indexer	35
7.2.3	Pseudo Code for Search	36
7.2.4	Pseudo Code for Ranking	36
7.3	Screenshots	37
7.3.1	Search Result	37
7.3.2	Database	37
7.3.3	Page Table	37
7.3.4	Word Table	38
7.3.5	Occurrence Table	38
7.3.6	Crawler Input	39
7.3.7	Crawler Results	39
7.3.8	Page Rank Result	40
Chapter- 8	Results	41
	Conclusion and Future Work	42
	Reference	43
	Appendix	44

ABBREVIATIONS AND SYMBOLS

DFD: Data Flow Diagram

LIST OF FIGURES

SNo.	Title	Page No.
1.	Components of Search Engine	7
2.	Web Crawler	8
3.	Performance Parameters	13
4.	J2EE Applocation Architecture	16
5.	Container Architecture	18
6.	Use Case Diagram	29
7.	Query Processor Diagram	30
8.	Indexer	31
9.	Activity Diagram	32
10.	Level -0 DFD	33
11.	Level- 1 DFD	33
12.	Methodology	34
13.	Search Result for Doctor	37
14.	Database	37
15.	Page Table	37
16.	Word Table	38
17.	Occurrence Table	38

18.	Crawler Input	39
19.	Crawler Result	39
20.	Page Rank Input	40
21.	Page Rank Result	40

LIST OF TABLES

SNo.	Title	Page No.
1.	Stop Words	10
2.	Performance Parameters	41

Abstract

The project aims to develop a Search Engine that will take into account various medical sites and search within them for the keywords entered by the user through the search bar. The result would be displayed taking into account the occurrences of the keywords. A search engine is a software system that is designed to search for particular keywords on the World Wide Web. The search engine returns a list of documents in which they were found.

Through this type of search engine it will be easier for people to search for medical information as it will only be taking into account medical sites, hence it will save time of the users and provide accurate results.

For implementation of my project I have chosen Java as it is easy to learn and start with a language which has integrated database support of the famous database MySQL.

CHAPTER-1

Introduction

A search engine is a software system that is designed to search for particular keywords on the World Wide Web. The search engine returns a list of documents in which they were found, especially a commercial service that scans documents. The results may be a mix of web pages, images, and other types of files. Some search engines also mine data available in databases or open directories. Search engines also maintain real-time information by running an algorithm on a web crawler.

People generally use search engines for research, shopping or entertainment. For instance people who are using a search engine for research purposes are generally looking for answers or at least for data with which to make a decision. They're looking for sites which can fulfil a specific purpose. Search engines are naturally drawn to research-oriented sites and usually consider them more relevant than shopping-oriented websites, because of which a lot of the time, the highest listing for the average query is a Wikipedia page. A smaller percentage of people, use a search engine in order to shop. This is where specialized engines come into the picture. Although you can use a regular search engine to find what it is you're shopping for, some people find it more efficient to use a search engine geared directly towards buying products. Some Web sites out there are actually search engines just for shopping. Amazon, eBay, and Shopping.com are all examples of shopping-only engines. Research and shopping aren't the only reasons to visit a search engine. The Internet is a vast, addictive and reliable source of entertainment, and there are users out there who use the search engines as a means of entertaining themselves. They look up things like videos, movie trailers, games, and social networking sites.

As we can see now that searching for medical information on the Web is a challenging task for ordinary Internet users. Often, users are uncertain about their exact medical situations and are unfamiliar with medical terminology, and hence have difficulty in coming up with the right search keywords. A search engine has no way of knowing who the searcher is - whether it is a doctor, a nurse or someone who is not in the medical field. For a search term, a general search engine presents results that are mixtures of materials meant for the general public and those meant for medical professionals. Moreover busy

medical professionals don't usually have time to devote to disorganized searches on the Internet or searches that don't return the information they need.

A Medical Search Engine is specifically designed to address this challenge as it is a custom search engine that jumps over these layman-focused sites and returns only medical site links on the first results page. This is done by specifically including only credible healthcare sites. It is preferable to use Medical Search Engines for searching medical information as they are selective in contrast to general search engines, they are comprehensive as they store the index of all the words and medical terms corresponding to the selected medical websites so all the medical practitioners, researchers can search full text of all such pages with a single search, moreover they are precise and target searchers as a Medical Search Engine examines the number of times medical terms are mentioned in each entry, and ranks them accordingly, thus making for more precise, targeted searches. It is also a fine tuned search, as the number of search results are limited and is easy and fast for the user to access information. Last but not the least Medical Search Engines are user friendly and patients/ users don't need to spend ages on learning it, hence saves time.

The existing Medical Search Engines are-

OmniMedicalSearch.com

Healthline.com

For implementation of the Medical Search Engine I am using PHP 5.3.5. As PHP is a language that is specifically designed for web programming with built-in integration with the most popular open source database MySQL. Moreover it has many advantages like it is easy to start with, easy to use. It has integrated database support and supports cheap hosting.

CHAPTER-2

History of Search Engine

The goal of all search engines is to find and organize distributed data found on the Internet. Before search engines were developed, the Internet was a collection of File Transfer Protocol (FTP) sites in which users would navigate to find specific shared files. As the central list of web servers joining the Internet grew, and the World Wide Web became the interface of choice for accessing the Internet, the need for finding and organizing the distributed data files on FTP web servers grew. Search engines began due to this need to more easily navigate the web servers and files on the Internet.

The first search engine was developed as a school project by Alan Emtage, a student at McGill University in Montreal. Back in 1990, Alan created Archie, an index (or archives) of computer files stored on anonymous FTP web sites in a given network of computers (“Archie” rather than “Archives” fit name length parameters – thus it became the name of the first search engine). In 1991, Mark McCahill, a student at the University of Minnesota, effectively used a hypertext paradigm to create Gopher, which also searched for plain text references in files.

Archie and Gopher’s searchable database of websites did not have natural language keyword capabilities used in modern search engines. Rather, in 1993 the graphical Mosaic web browser improved upon Gopher’s primarily text-based interface. About the same time, Matthew Gray developed Wandex, the first search engine in the form that we know search engines today. Wandex’s technology was the first to crawl the web indexing and searching the catalog of indexed pages on the web. Another significant development in search engines came in 1994 when WebCrawler’s search engine began indexing the full text of web sites instead of just web page titles.

While both web directories and search engines gained popularity in the 1990s, search engines developed a life of their own becoming the preferred method of Internet search. For example, the major search engines found in use today originated in development between 1993 and 1998.

2.1) History of Yahoo

David Filo and Jerry Yang started Yahoo! in 1994. Originally it was a highly regarded directory of sites that were catalogued by human editors. This directory provided an extensive listing of websites supported by a network of regional directories. In 2001, Yahoo! started charging a fee for inclusion in its directory listing. Yahoo!'s action helped control the number of sites listed and helped cover costs with additional revenue.

Initially, Yahoo! used secondary search engine services to support its directory. Partnerships have included agreements with Inktomi and Google. In October 2002, Yahoo shifted to crawler-based listing for its search results. In 2004, Yahoo! purchased Overture's pay-per-click service, which had only months earlier purchased AltaVista and All the Web, and Inktomi's search database. With these acquisitions, Yahoo! combined these tools to create its own search index. Today, Overture is renamed Yahoo! Search Marketing and provides paid search advertising revenue. The Yahoo! Directory remains one of the top indexes powering search listings.

2.2) History of Ask

Ask was developed in 1996 by Garret Gruener and David Warthen and launched in 1997 as Ask Jeeves. In 2006, the "Jeeves" name was removed; revamping its image after Ask Jeeves was purchased in 2005 by Barry Diller's InterActiveCorp (IAC). Originally as Ask Jeeves, human editors listed the prominent sites along with paid listings and results pulled from partner sites. Following acquisition of Direct Hit in 2000 and Teoma in 2001, Ask commenced developing its own search technology. With financial growth, Ask has acquired other companies including Excite and iWon.

Today, with emphasis on paid inclusion listings, Ask struggles for market share against Google, Yahoo!, and MSN Search.

2.3) History of Google

Google was founded in 1998 as another school project at Stanford University in California. In January 1996, Stanford PhD students Larry Page and Sergey Brin began researching the concept of a search engine based on relevancy ranking. Page and Brin believed that search engines should analyze and rank websites based on the number of times search terms appeared on web pages. Likewise, Page and Brin developed a search engine nicknamed “BackRub.” BackRub checked the number and quality of links coming back to websites in order to estimate the value of a website. Brin and Page’s research eventually led them to develop the trademarked PageRank link analysis algorithm that Google’s search engine would use to assign a numerical weighting to hyperlinked document elements.

In 2000, Google replaced Inktomi as the provider of search results to Yahoo! and later AOL and Netscape. Even though Yahoo! broke away from Google in 2004, its market share has continued to grow to account for about 70 percent of all web searches. Google’s market share has steadily increased over the years.

2.4) History of MSN Search (Now Windows Live)

MSN Search was a service offered as part of Microsoft’s network of web services. The Microsoft Network debuted as an online service and Internet service provider in August 1995. During the 1990s, Microsoft launched Internet Explorer as a bundled part of their operating system and software products. MSN Search first launched in 1998 displaying search results from Inktomi and later blending results with Looksmart. For a short time in 1999, AltaVista search results were used instead of Inktomi. Since 2004, MSN Search began using its own built-in search results. Since this time, MSNBot has continually crawled the web. Today, image search is powered by Picsearch. MSN Search was renamed Windows Live in 2006.

CHAPTER-3

Literature Review

3.1) Components of Search Engine

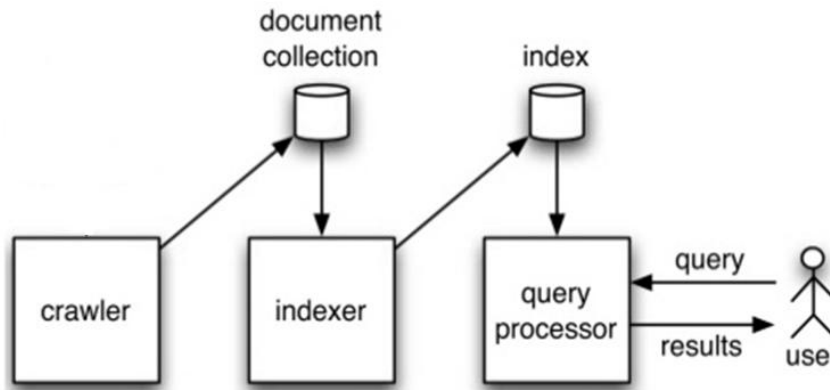


Figure: 1 Components of Search Engine

3.1.1) Crawler

The Process or Program used by search engines to download pages from the web for later processing by a search engine that will index the downloaded pages to provide fast searches. It starts with a list of URLs to visit. As the crawler visits these URLs, it identifies all the hyperlinks in the page and adds them, to the list of visited URL's called the crawl frontier. URLs from the frontier are then visited and indexed.

a) Architecture

- **URL FRONTIER:** Contains URLs yet to be fetched.
- **FETCH:** Generally use the http protocol to fetch the URL
- **PARSE:** The page is parsed. Texts and Links are extracted.
- **REPETITION:** Test whether the webpage with the same content has already been seen at another URL.

Web crawling is a process to collect all the web pages that are interested to search engine. It's a usually a challenging task for general search engine. But web crawling is quite easy for site-specific search engine because the developers of site-specific search engine usually have access to the web pages of the web site. In this project, I downloaded about 10-20 most frequently accessed web pages.

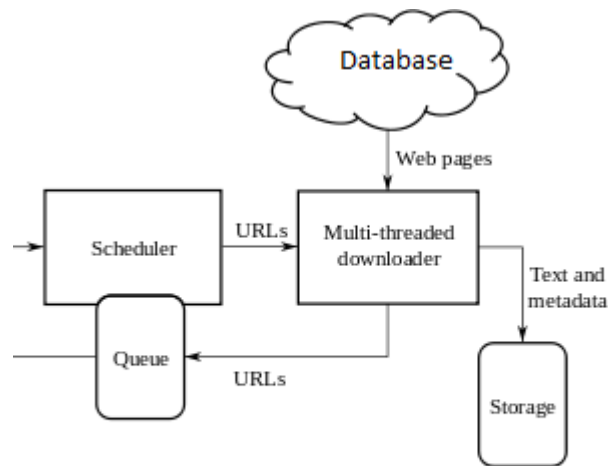


Figure: 2 Web Crawler

3.1.2) Tokenization Process

The process of tokenization is to convert each document to a set of keywords or tokens. A token (case insensitive) is a document word which is supposed to represents the document theme or meaning. In SE, all words of documents, after processing, are considered as tokens. The whole tokenization process can be summarized in following important steps.

a) Text Extraction from HTML File

A corpus contains html documents, document need to be parsed to extract text from html tags. First of all, to make the regular expression matching faster, all the comments (`/*...*/` and `<!--...-->`) are removed to reduce document size. Mainly the text from the body part is considered for token extraction. First of all 'script' (`<script>...</script>`) and 'style'

(<style>...</style>) part have been removed from the body part. After that rest of the html tags (<...>) are replaced by empty string. This process produces html tag free text from body part.

b) Punctuation Removal

We generally use different punctuation symbols in our text. But these symbols themselves don't help in understanding document themes. So during token extraction from texts, these punctuation symbols can be filtered out. In the process it also reduces the number of distinct tokens like the "world" and "world?" end up with producing single token "world" and "?" is filtered out.

c) Stop Word Removal

All words from the text are not considered as token. Usually some words occur frequently in almost all of the documents. Because of this property, their discrimination power is negligible. These types of words are called stop words and these words can be filtered out during tokenization.

Different types of stop word are:

The	All	Am	Are
Before	By	Can	Come
Did	Does	Each	Else
For	From	Go	Got
Has	He	In	Is
Less	Let	Of	Only
Said	So	Still	Take
This	Till	To	Via

Table: 1 Stop words

3.1.3) Indexer

The information the search engine spider found on each page is analyzed, building a list of the words and phrases within the document. An Index is a database where information after being collected, parsed and processed is stored to allow for quick retrieval. If we don't have the index, it will take too much time to search through the whole site to find the document that matched our query. Creating an index means that retrieval process is faster and the accuracy is better. It saves on storage and makes the whole process faster.

3.1.4) Search

A web search query is a query that a user enters into a web search engine to satisfy his or her information needs. Web search queries are distinctive in that they are often plain text or hypertext with optional search-directives (such as "and"/"or" with "-" to exclude).

- **Term frequency:** How frequently a query term appears in a document is one of the most obvious ways of determining a document's relevance to a query.
- **Location of terms:** Many search engines give preference to words found in the title or lead paragraph or in the metadata of a document. Some studies show that the location — in which a term occurs in a document or on a page — indicates its significance to the document.
- **Proximity of query terms:** When the terms in a query occur near to each other within a document, it is more likely that the document is relevant to the query than if the terms occur at greater distance.

3.1.5) Ranking of Results

Search engines use complex, proprietary algorithms to determine website ranking in the organic search results.

- **Content:** Search engines reward websites that contain fresh, current, compelling, and unique content; they penalize websites that aren't updated

regularly. Adding a blog to your website (and contributing to it on a regular basis) is one of the best ways to get your website found in the organic search results.

- **Inbound Links:** Search engines pay considerable attention to whether or not other authoritative websites are linking back to yours. If an authoritative website in your industry is linking back to content found on your website, that's a good indication that your website is relevant.
- **Activity:** Search engines consider the number of visitors to your website, the amount of time they spend, the number of pages they visit, where they click, and where they came from, to be additional measures of your website's relevance.
- **Popularity and Social Sharing:** Search engines are increasingly taking into account whether or not websites are popular. Your presence on social media, whether or not you've published videos, the existence of online reviews.

3.1.6) Page Rank Algorithm

Let us suppose page A has pages T1...Tn which point to it (i.e., are citations). The parameter d is a damping factor which can be set between 0 and 1. We usually set d to 0.85. There are more details about d in the next section. Also C(A) is defined as the number of links going out of page A. The PageRank of a page A is given as follows:

$$PR(A) = (1-d) + d (PR(T1)/C(T1) + \dots + PR(Tn)/C(Tn))$$

Assume a small universe of four web pages: A, B, C and D. The initial approximation of PageRank would be evenly divided between these four documents. Hence, each document would begin with an estimated PageRank of 0.25. In the original form of PageRank initial values were simply 1. This meant that the sum of all pages was the total number of pages on the web. Later versions of PageRank (see the below formulas) would assume a probability distribution between 0 and 1. Here we're going to simply use a probability distribution hence the initial value of 0.25. If pages B, C, and D

each only link to A, they would each confer 0.25 PageRank to A. All PageRank PR() in this simplistic system would thus gather to A because all links would be pointing to A.

$$PR(A) = PR(B) + PR(C) + PR(D)$$

But then suppose page B also has a link to page C, and page D has links to all three pages. The value of the link-votes is divided among all the outbound links on a page. Thus, page B gives a vote worth 0.125 to page A and a vote worth 0.125 to page C. Only one third of D's PageRank is counted for A's PageRank (approximately 0.083).

$$PR(A) = PR(B)/2 + PR(C)/1 + PR(D)/3$$

In other words, the PageRank conferred by an outbound link L() is equal to the document's own PageRank score divided by the normalized number of outbound links (it is assumed that links to specific URLs only count once per document).

3.1.7) Performance of a Search Engine

a) Response Time

This is the time period that starts from the point of query submission until user gets a huge list of responded results. Response time is directly related to activeness of the search engine and is kept as minimum as possible. Time saving mechanism such as search result caching and index tiering is heavily exploited, despite the risk that such approaches may cause relevant content to be missed.

b) Total Number of Results

To find the relevant result, it is essential to cover the whole area of the web and select the best among them. User does not prefer a huge list of results. It is better for quality pages that fulfill the need of the user's query.

c) Precision and Recall

- **Recall** is the fraction of relevant documents retrieved from all relevant documents in the collection.

For example if there are 100 documents in the collection, 10 are relevant, 12 are retrieved and out of those only 3 are relevant so the recall is 0.3 or 30%

- **Precision** is the fraction of retrieved documents that are relevant.

For example if there are 100 documents in the collection, 10 are relevant, 12 are retrieved and out of those only 3 are relevant so the precision is 0.25 or 25%

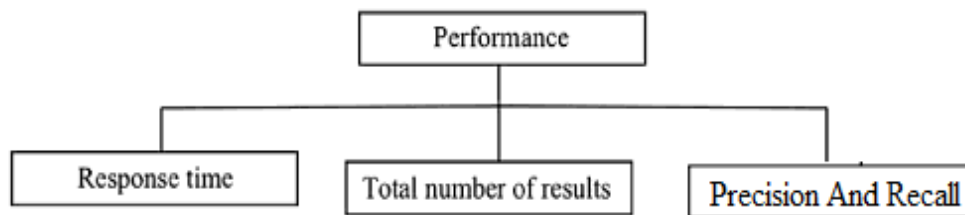


Figure: 3 Performance Parameter

3.2) Types of Search Engine

Although the term "search engine" is often used indiscriminately to describe crawler based search engines, human-powered directories, and everything in between, they are not all the same. Each type of "search engine" gathers and ranks listings in radically different ways.

3.2.1) Crawler Based

Crawler-based search engines, compile their listings automatically. They "crawl" or "spider" the web and people search through their listings. These

listings are what make up the search engine's index or catalogue. You can think of the index as a massive electronic filing cabinet containing a copy of every web page the spider finds. Because spiders scour the web on a regular basis, any changes you make to a web site may affect your search engine ranking.

3.2.2) Directories

Directories such as Open Directory depend on human editors to compile their listings. Webmasters submit an address, title, and a brief description of their site, and then editors review the submission. Unless you sign up for a paid inclusion program, it may take months for your web site to be reviewed. Even then, there's no guarantee that your web site will be accepted. After a web site makes it into a directory however, it is generally very difficult to change its search engine ranking.

3.2.3) Mixed Results/ Hybrid Search Engine

Some search engines offer both crawler-based results and human compiled listings. These hybrid search engines will typically favour one type of listing over the other however. Many search engines today combine a spider engine with a directory service. The directory normally contains pages that have already been reviewed and accessed.

3.2.4) Miscellaneous Types

The search engine can be categorized as follows based on the application for which they are used:

- a) Primary Search Engines:** They scan entire sections of the World Wide Web and produce their results from databases of Web page content, automatically created by computers.
- b) Subject Guides:** They are like indexes in the back of a book. They involve human intervention in selecting and organizing resources, so they cover fewer resources and topics but provide more focus and guidance.
- c) People Search Engines:** They search for names, addresses, telephone numbers and e mail address.

- d) Business and Services Search Engines:** They essentially national yellow page directories.
- e) Employment and Job Search Engines:** They either provide potential employers access to resumes of people interested in working for them or provides prospective employees with information on job availability.
- f) Finance-Oriented Search Engines:** They facilitate searches for specific information about companies (officers, annual reports, SEC filings, etc.)
- g) News Search Engines:** They search newspaper and news web site archives for the selected information
- h) Image Search Engines:** They which help you search the WWW for images of all kinds.
- i) Specialized Search Engines:** They that will search specialized databases, allow you to enter your search terms in a particularly easy way, look for low prices on items you are interested in purchasing, and even give you access to real, live human beings to answer your questions.

CHAPTER-4

Tools and Techniques Used

4.1) Java EE Framework

The Java2 Platform enterprise Edition (J2EE) technology provides a component based approach to the design, development, assembly and deployment of enterprise applications. The J2EE platform provides a multi tier distributed application model. The ability to reuse components, a unified security model and a flexible transaction control. Not only can you deliver innovative customer solutions to your clients, faster than ever but your platform independent J2EE component based solutions are not tied to the products and API's of any one wonder.

4.1.1) Distributed Multi-Tiered Applications

The J2EE Platform uses a multi-tiered distributed application model. This means application logic is divided into components according to the functions and the various application components that make up a J2EE application are installed on different machines depending on which tier in the multi-tiered J2EE environment the application component belongs. Figure below shows two multi-tiered J2EE applications divided into tiers described in the bullet list.

- Client Tier Component run on the client machine
- Web Tier Component run on the client machine
- Business Tier Component run on the J2EE server

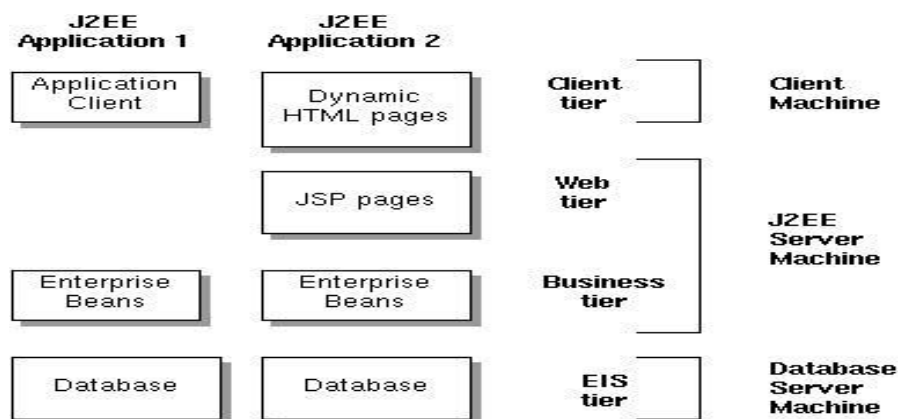


Figure: 4 J2EE Application Architecture

4.1.2) J2EE Application Components

A J2EE component is a self- contained functional software unit that is assembled into a J2EE application with its related classes and files and communicates with other components. The J2EE specifications defines the following J2EE components.

- Application Clients and Applets are client components
- Java servlet and JavaServerPages (JSP) Technology Components are web components
- EnterpriseJavaBean (EJB) components are Business components

J2EE components are written in java programming language and compiled in the same way as any java programming language program. The difference when you work with J2EE platform, is J2EE components are assembled into J2EE applications, verified that they are well formed and in compliance with the J2EE specifications and deployed to production when they are run and managed by J2EE server.

The deployment process installs J2EE application components in the following types of J2EE containers.

- An EnterpriseJavaBean (EJB) container manages the execution of all enterprise beans for one J2EE application. Enterprise beans and their container run on the J2EE server
- A Web container manages the execution of all JSP pages and servlet components for one J2EE application. Web components and their container run on the J2EE server.
- An Application client container manages the execution of all application client components for one J2EE application. Application clients and their container run on the client machine.
- An applet container is the web browser and java plug-in combination running on client machine.

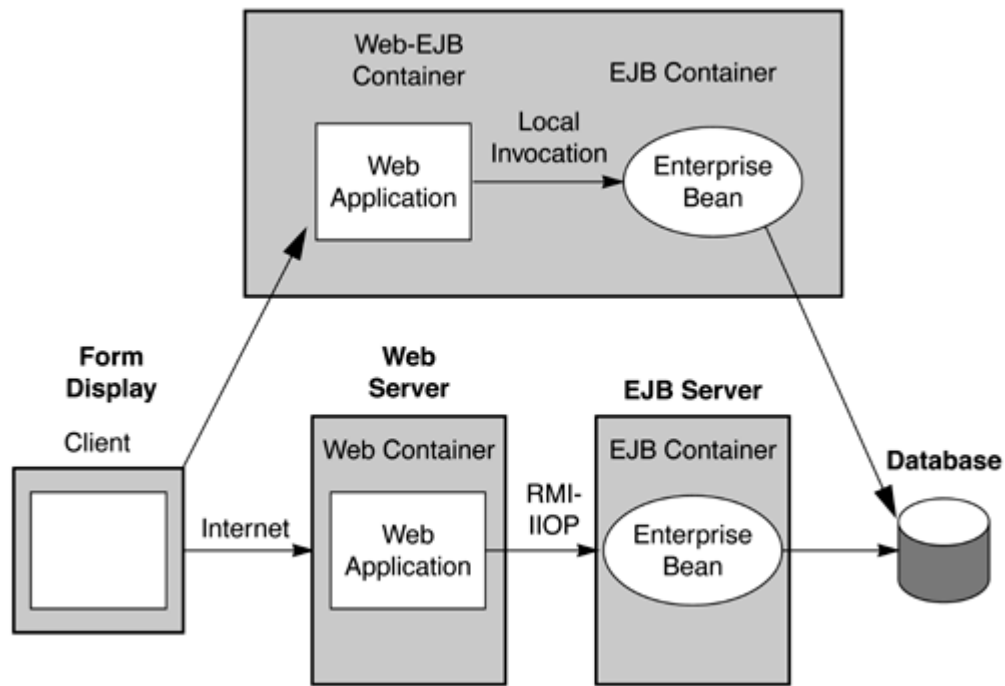


Figure: 5 Container Architecture J2EE

4.1.3) Java Server Pages

Architecturally, JSP may be viewed as a high-level abstraction of Java servlets. JSP pages are loaded in the server and operated from a structured special installed Java server packet called a Java EE Web Application, often packaged as a .war or .ear file archive.

JSP allows Java code and certain pre-defined actions to be interleaved with static web markup content, with the resulting page being compiled and executed on the server to deliver a document. The compiled pages, as well as any dependent Java libraries, use Java bytecode rather than a native software format. Like any other Java program, they must be executed within a Java virtual machine (JVM) that integrates with the server's host operating system to provide an abstract platform-neutral environment.

Java syntax is a fluid mix of two basic content form : Scriplet elements and markups

Markup is typically standard HTML or XML, while scriplet element are delimited block of Java code which may be intermixed with the markup. When the page is requested the javacode is executed and its output is added, in situ, with the surrounding markup to create the final page. Because Java is a compiled language , not a scripting language, JSP pages must be compiled to Java byte Code classes before

they can be executed, but such compilations is needed only when a change to the source JSP file has occurred.

Java code is not required to be complete(Self contained) within its scriptlet element block, but can straddle markup content providing the page as a whole is syntactically correct(For example, any Java if/for/while blocks opened in one scriptlet element must be correctly closed in a later element for a page to successfully compile).This system of split inline coding section is called step over scripting because it can wrap around the static markup by stepping over it. Markup which falls inside a split block of code is subject to that code , so markup inside an if block will only appear in the output when the if conditions evaluated to true, likewise markup inside a loop construct may appear multiple times in the output depending upon how many times the loop body runs.

The JSP syntax adds additional XML-like tags , called JSP actions, to invoke ,build in functionality .Additionally , the technology allows for the creation of JSP tag libraries that act as extension to the standard HTML or XML tags. JVM operated tag libraries provide a platform independent way of extending the capabilities of a web server. Note that not all commercial Java servers are JAVA EE specification compliant.

Starting with the version 1.2 of JSP specification, Java Server Pages have been developed under the Java Community Process. JSR 53 defines both the JSP 1.2 and Servlet 2.3 specification and JSR 152 defines the JSP 2.0 specification.

The new version of JSP specification includes new features meant to improve programmer productivity. Namely

- An Expression Language (EL) which allows developers to create Velocity-style templates (among other things)
- A faster/Easier way to display parameter values
- A clearer way to navigate nested beans.

4.1.4) Java Servlet

A Servlet is a Java class which conforms to the Java Servlet API, a protocol by which a Java class may respond to HTTP requests. Thus, a software developer may use a servlet to add dynamic contents to a Web server using Java Platforms. The Generated contents is commonly HTML, but may be other data

such as XML. Servlets are the JAVA counterpart to Non-java dynamic web content technologies such as CGI and ASP.NET. Servlet can maintain state in a session variables across many server transaction using HTTP cookies, or URL rewriting.

The Servlet API, contained in the Java Package hierarchy javax.servlet, defines the expected interactions of a web container and servlet. A web container is essentially the component of a web server that interacts with the servlets. The Web container is responsible for managing the lifecycle of servlets , mapping a URL to a particular servlet and ensuring that the URL requester has the correct access rights.

4.1.5) Java Features

Some of the important features of java are:

- Simplicity
- Object Oriented
- Platform Independent
- Security
- Robust
- High Performance
- Multi Threading
- Dynamic Linking
- Garbage Collection

One of the most important feature of java is Platform independence which makes it famous and suitable language for World Wide Web.

4.2) HTML

Hyper Text Markup Language (HTML) is a language for describing how pages of text , graphics , and other information are organised. Hypertext means text stored in electronic form with cross-reference links between pages. An HTML page contains HTML tags,which are embedded commands that supply information about the page's structure ,appearance ,and contents. Web browser use this information to determine how to display the page. The

purpose of a web browser is to read HTML documents and compose them into visual or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured document by denoting structural semantic for text such as headings, paragraph list, links, quotes and other items.

4.3) MySQL Database

MySQL is a open source Relational Database Management System. MySQL is very fast, reliable and flexible Database Management System. It provides a high performance and it is multi threaded and multi user Relational Database Management System.

MySQL is one of the most popular relational database Management System on the web. The MySQL Database has become the world's most popular open source Database, because it is free and available on almost all the platforms. MYSQLE can run on UNIX, Windows and MAC OS. MYSQLE is used for internet applications as it provides good speed and is very secure. MYSQLE has developed to manage large volumes of data at very high speed to overcome the problems of existing solutions.

4.3.1) MySQL Features

- MYSQLE is very fast and much reliable for any type of application
- MYSQLE is a very light weight application
- MYSQLE command line tool is very powerful and can be used to run SQL queries against database
- MYSQLE supports indexing and binary objects
- It allows changes to structure of tables while server is running
- MYSQLE has a wide user base
- MYSQLE code can be tested with multiple compilers
- MYSQLE is available as a separate program for use in a client/server network environment

4.3.2) MySQL Advantages

- Reliability and Performance

MYSQL is very reliable and high performance relational database management system. It can be used to store many GB's of data in the database

- Availability of Source

MYSQL source code is available that is why you can now recompile the source code

- Cross Platform Support

MYSQL supports more than twenty different platforms including major Linux distribution.

- Large pool of trained and certified developers

MYSQL is very popular and it is the world's most popular open source database, so it is easy to find high quality staff around the world.

4.4) PHP

4.4.1) Advantages of PHP

PHP is a language that is specifically designed for web programming with built-in integration with the most popular open source database MySQL.

a) Easy to start with

As a beginner it is easy to start with PHP. The user just have to add a few PHP-tags with e.g. a for-loop in it's existing HTML-files and then upload it to the server and see the result or an error message. Dynamic typing and associative arrays makes it also easier to start using PHP.

b) Easy to use

Compared to most solutions like e.g. Java, PHP doesn't need to be compiled, so it's just to write the script and then upload it to the server and then update the browser.

c) Integrated database support

PHP has (mostly) built-in support for the most popular databases like e.g. MySQL, that means it is easy to start using databases, no additional drivers needs to be installed, just to use the mysql-functions. The easy to use web based admin tool PHPMyAdmin (released 1998) is also important to the PHP's success in combination with MySQL.

d) Old language (since 1995) with a big user base

PHP became popular early (1995) since it was designed for web programming. Since then the user base has grown and now there is many web-oriented frameworks and libraries available. Some examples are blogg-systems and e-shopping-platforms.

e) Cheap hosting

Since PHP has existed for long time and works good on both Linux and Windows, and many web servers have support for it. There is no problem to find hosting with PHP pre-installed.

4.4.2) PHP Usage

There are three main areas where PHP scripts are used.

a) Server-side scripting

This is the most traditional and main target field for PHP. You need three things to make this work. The PHP parser (CGI or server module), a web server and a web browser. You need to run the web server, with a connected PHP installation. You can access the PHP program output with a web browser, viewing the PHP page through the server.

b) Command line scripting

You can make a PHP script to run it without any server or browser. You only need the PHP parser to use it this way. This type of usage is ideal for scripts

regularly executed using cron (on *nix or Linux) or Task Scheduler (on Windows). These scripts can also be used for simple text processing tasks.

PHP can be used on all major operating systems, including Linux, many Unix variants (including HP-UX, Solaris and OpenBSD), Microsoft Windows, Mac OS X, RISC OS, and probably others. PHP has also support for most of the web servers today. This includes Apache, IIS, and many others. And this includes any web server that can utilize the FastCGI PHP binary, like lighttpd and nginx. PHP works as either a module, or as a CGI processor.

So with PHP, you have the freedom of choosing an operating system and a web server. Furthermore, you also have the choice of using procedural programming or object oriented programming (OOP), or a mixture of them both.

With PHP you are not limited to output HTML. PHP's abilities include outputting images, PDF files and even Flash movies. You can also output easily any text, such as XHTML and any other XML file. PHP can auto generates these files, and save them in the file system, instead of printing it out, forming a server-side cache for your dynamic content.

One of the strongest and most significant features in PHP is its support for a wide range of databases. Writing a database-enabled web page is incredibly simple using one of the database specific extensions (e.g., for mysql), or using an abstraction layer like PDO, or connect to any database supporting the Open Database Connection standard via the ODBC extension. Other databases may utilize cURL or sockets, like CouchDB.

PHP has useful text processing features, which includes the Perl compatible regular expressions (PCRE), and many extensions and tools to parse and access XML documents.

4.4.3) PHP for Web Development

a) Perfect Database Interaction

PHP is an excellent language choice when it comes to building Dynamic Websites that interact with Databases, as it can exchange all sorts of information with ease.

b) Cost

Another reason why PHP website development and web development is admired by developers is that PHP programs run on Linux, which is free. Also, the database connectivity is less expensive as compared to that of other programs such as ASP which is based on MS_SQL, a Microsoft product that needs to be purchased. However, web development with PHP through MySQL is free to use.

c) Incredible Speed

PHP has an upper hand when it comes to speed. This is mainly because the PHP code runs faster as it runs in its own memory space.

CHAPTER-5

System Requirements

5.1) Requirement Analysis

Information gathering is usually the first step of a project. The purpose of this phase is to identify and document the exact requirements for the system. The user's request identifies the need for a new information system. The objective is to determine whether the request is valid and feasible before a recommendation is made to build a new or an existing manual system.

The major steps are-

- Defining the user requirements
- Studying the present system to verify the problem

5.1.1) Hardware Requirements

There are no limits on the type of hardware used to host an installation. However, as your needs grow, so will the need for additional resources. A simple installation requires about 50M of free disk space and 32M of free system memory. However, a minimum of 128M free memory available to PHP is recommended. Disk space should be allocated according to usage.

5.1.2) Software Requirements

The installation requires an environment running the following:

- EclipseN
- PHP version 5.1+ (PHP 5.2+ recommended)
- Mysql Server 4+ (Mysql 5 is strongly recommended)
- Web server (Apache 2+ recommended)
- Write access from the web server to some folders inside the efront installation

5.1.3) Functional Requirements

Function requirements mean the physical modules, which are going to be produced in the proposed systems. The functional requirements of the proposed system are described below:

- Web Crawling
- Words Extracting
- Indexing
- Ranking
- Search
- Search Results Ordering

5.1.4) Non- Functional Requirements

Non-Function requirements mean the characteristics that are not related to system's physical functions.

- **User Friendly**- The system should be easy to use, i.e. user friendly, for both administrator and external users. This can be done for providing function descriptions.
- **Secure**. The system should be secure. If not, hackers can access the database and undergo destruction.
- **Reliable**- The system must be resistant to failure.
- **Maintainability**- The website must be easy to maintain by the administrator.
- **Availability**- The services should be available 24X7.
- The user interface screen should be loaded immediately.
- Queries shall return results as soon as possible.

CHAPTER-6

Analysis

6.1) Introduction

The objective of the system design is to deliver the requirements as specified. System design involves first logical design and then physical construction of the system. The logical design describes structure and characteristics of features, such as outputs, inputs, files, databases and procedures. The physical construction produces actual program software, files and a working system.

System design goes through two phases of development-

6.1.1) Logical Design

We know that a data flow diagram shows the logical flow of the system and defines the boundary of the system. Logical design specifies the user need at a level of details that virtually determine the information flow into and out of the system and the required data resources. Logical design describes the input, output, database and procedures, all in a format that meets user requirements.

6.1.2) Physical Design

It provides the working system by designing the design specification that tells programmers what exactly the system must do. In short it can state that physical design is the implementation of the logical design.

Physical system design must consist of the following-

a) Design the physical system

- Specify input, output media
- Design the database and specify the backup procedures
- Design physical information through the system

b) Plan system implementation

6.2) Use Case Diagram

Use Case Diagram is the basic analysis diagram. It is the first analysis diagram. It gives interaction of the system with entities outside the system. These entities are called actors. Actors are the users of the system or other systems who give input to the system and take output from the system. The various scenarios in the system are called use case. They are denoted by oval. The complete system has been explained using a basic use case diagram which shows the interaction between the main actors and a set of use case scenarios.

The detailed use case diagrams following the basic use case diagram gives the detailed interaction of each actor with the system.

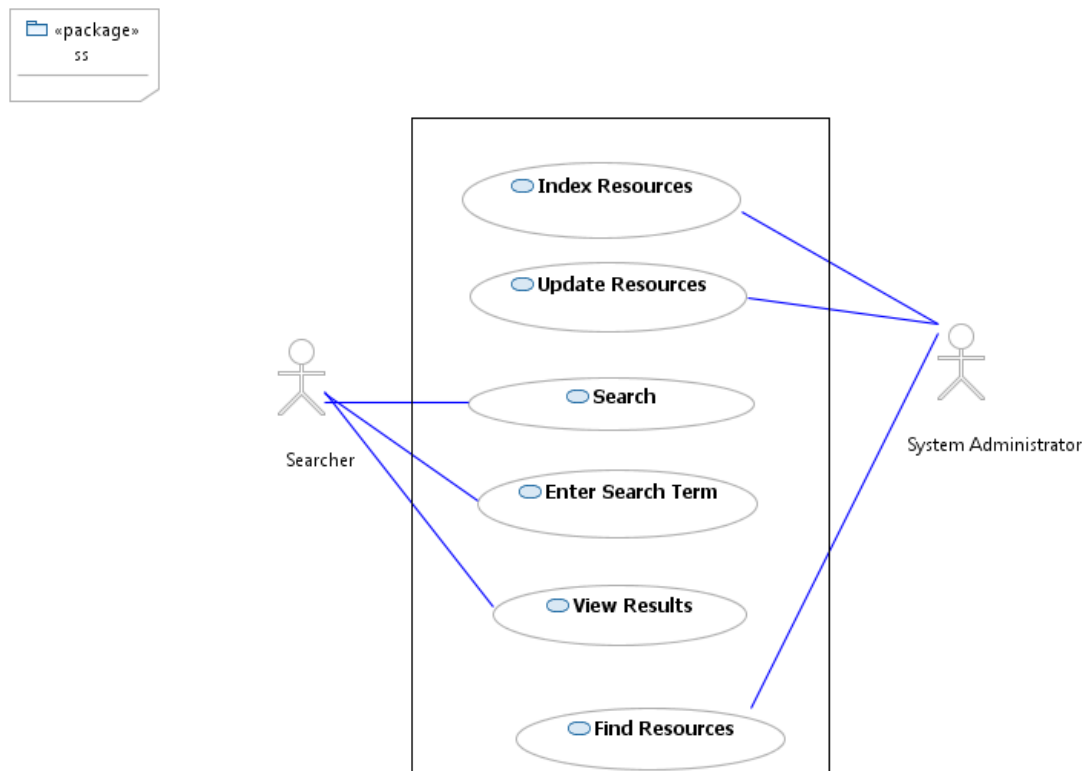


Figure: 6 Use Case Diagram

6.2.1) Actors Involved

a) Searcher

- Search
- Enter search item
- View Result

b) System Administrator

- Index resources
- Updates resources
- Find resources

6.3) Sequence Diagram

It is the analysis diagram which gives the sequence of events taking place in the system. The diagram gives the objects in the system and their interaction among each other. The diagram also shows the messages passed between the objects.

6.3.1) Query Processor

The main view of the system is query processing. This is the main function of the search engine with respect to the user. The main objects involved in this function are user, presenter, searcher and store server. The user submits a query which is given to the presenter which in turn submits the query to the searcher. The searcher searches for the keywords and creates a document list which gives the address of all the URL's related to the mentioned keywords. The presenter then requests the store server to return document at each URL. It then creates a summary for each URL and presents it to the user in the form of a list of URL's along with their summary.

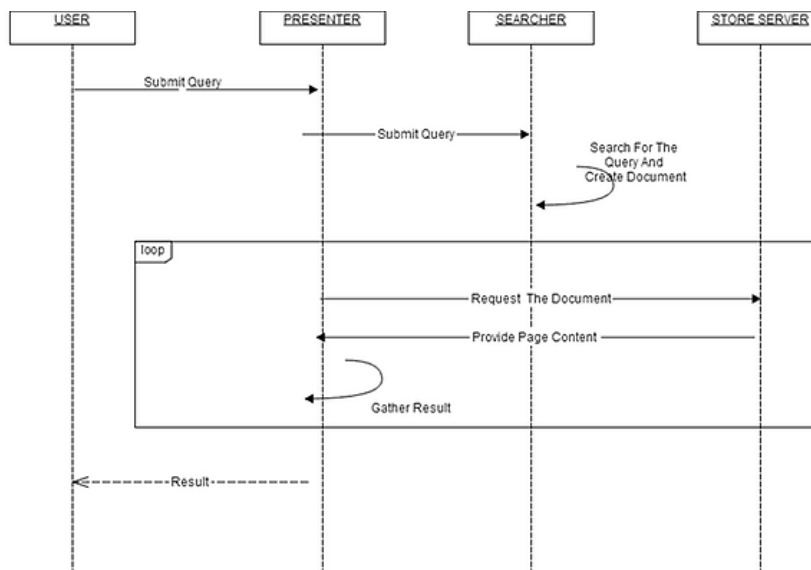


Figure: 7 Query Processing Sequence Diagram

6.3.2) Indexer

The indexer is another continuously running search engine program which has the functionality after the crawler. Its main task is to index the crawled web pages. The objects involved are indexer, store server and URL server. The indexer requests a new page from the store server. It then scans the page and prepares forward index for each page.

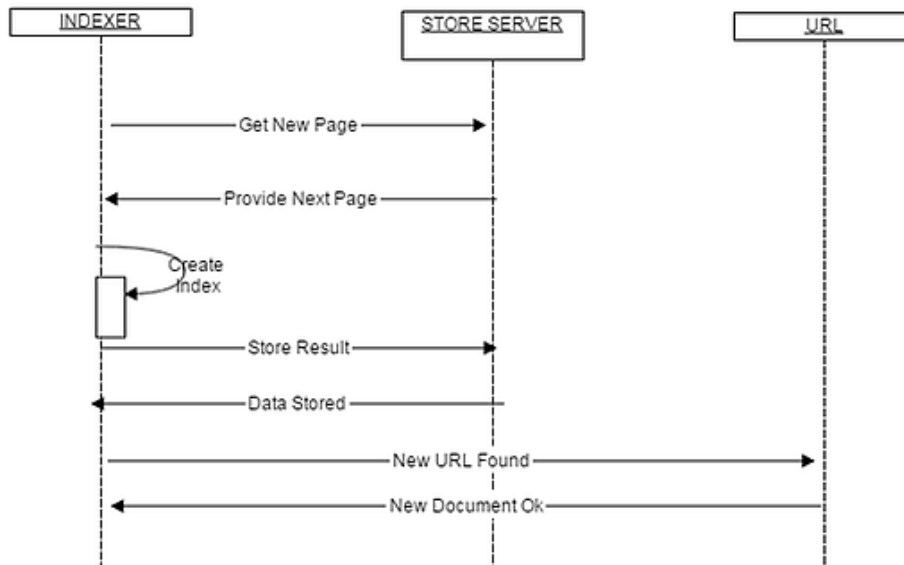


Figure: 8 Indexer Sequence Diagram

6.4) Activity Diagram

This is another UML diagram used in object oriented analysis. This diagram gives the list of activities being performed in the system. It starts with a filled circle and ends with bull's eye. A condition can also be shown in activity diagram using a diamond having more than one output. Activity is a particular operation of the system. Activity diagrams are used for visualizing dynamic nature of a system. The specific usage is to model the control flow from one activity to another. This control flow does not include messages.

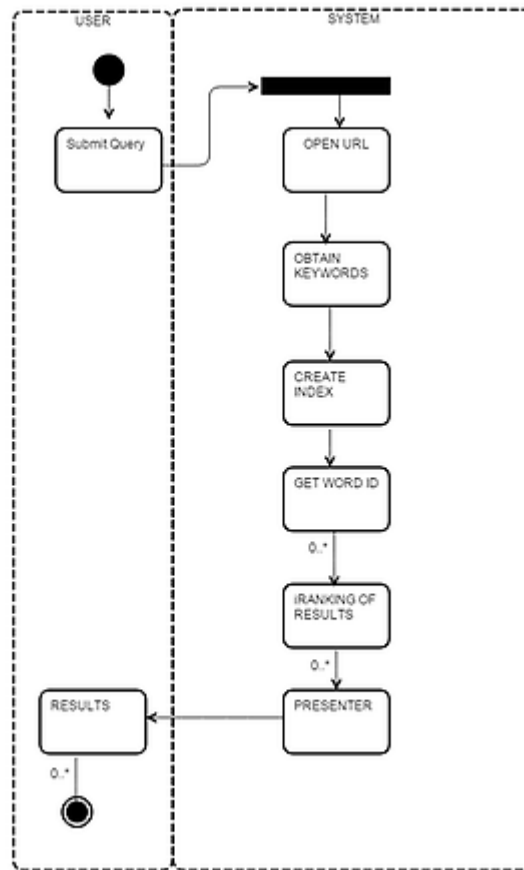


Figure: 9 Activity Diagram

6.5) Data Flow Diagram

Information is transformed as it flows through a computer based system. The system accepts input in a variety of forms and produces output in varied forms. For depicting this information flow in functional modelling, Data Flow Diagram (DFDs) are used. A Data Flow Diagram is a graphical representation that depicts information flow and the transforms that are applied as data move from input to output. The data flow diagram may be used to represent the system or software at any level of abstraction. DFDs may be partitioned into levels that represent increasing information flow and functional details. The DFD provides mechanism for functional modelling as well as information flow modelling.

6.5.1) Level – 0 DFD

A level 0 DFD, also called a fundamental system model or a context model, represents the entire software element as a single bubble with input and output data indicated by incoming and outgoing arrows, respectively

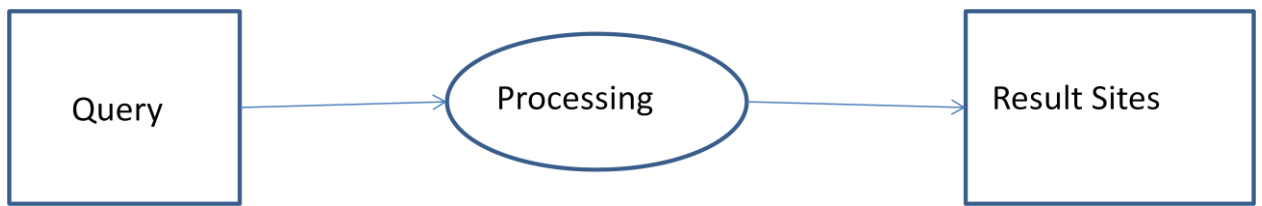


Figure: 10 Level-0 DFD

6.5.2) Level – 1 DFD

Additional processes (bubbles) and information flow paths are represented as level 0 DFD is partitioned to reveal more detail. The input and output remains the same but the process is broken down.

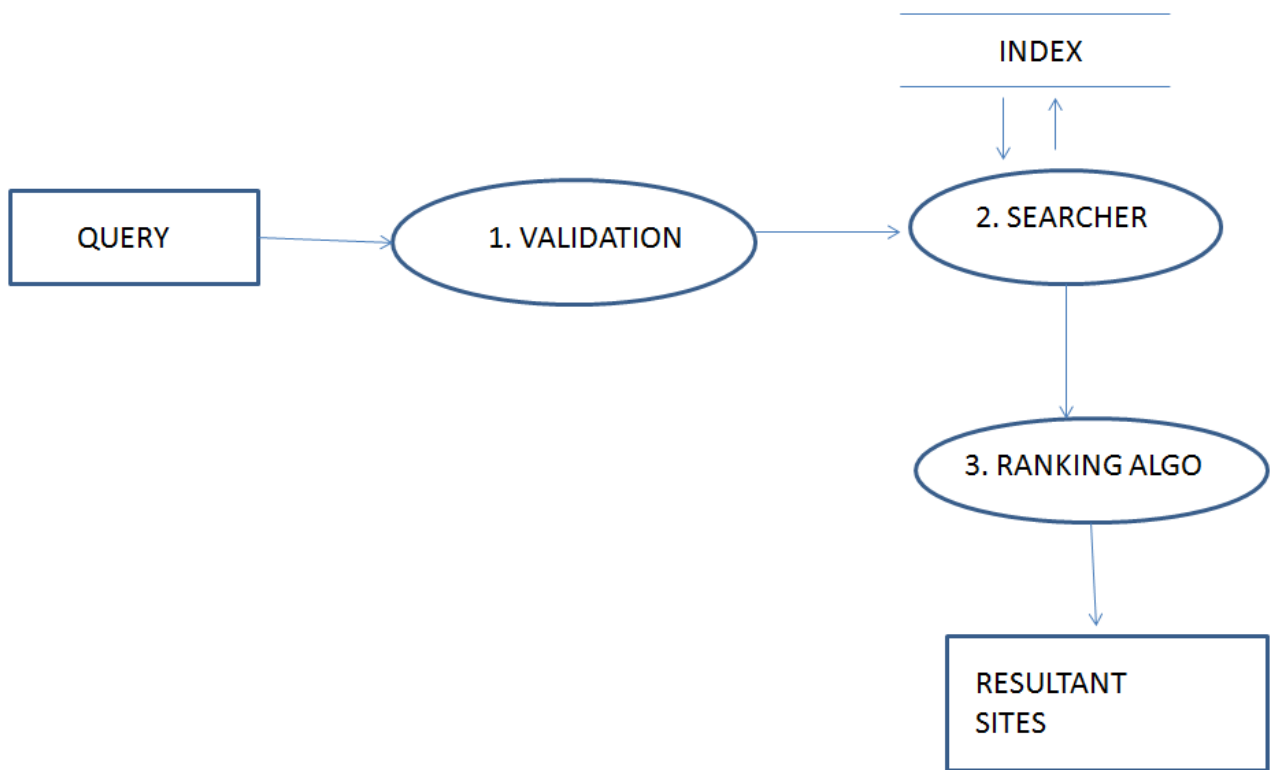


Figure: 11 Level -1 DFD

Chapter-7

Design and Implementation

7.1) Design

The design tells us how the search engine will be implemented. The following diagram shows the design of the search engine.

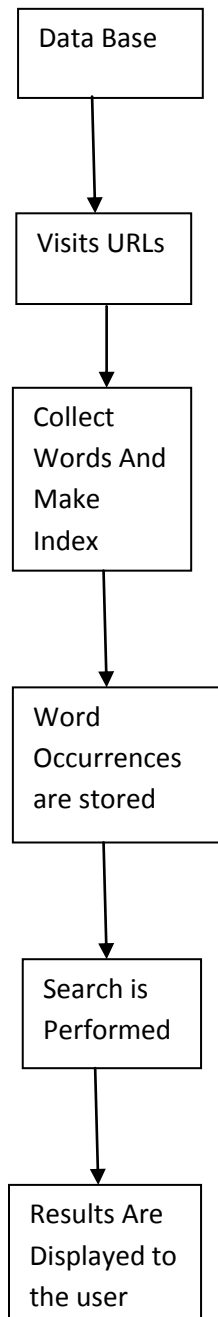


Figure: 12Methodology

7.2) Implementation

7.2.1) Pseudo Code for Crawler

```
Connect to database
if connected
do
    Get URL from form
    Truncate Records
    Processpage()
do
    Insert URL into database
    Open the URL
    GetLink()
    Processpage(link)
end
DisplayURL
End
```

7.2.2) Pseudo Code for Indexer

```
Connect to database
if connected
do
    Get URL from form
    if ( URL )
do
    Get the web page
    Remove HTML syntax
    Strip HTML tags, scripts, and styles
    Process the page's words
    Split the text into a word list
    For Each Word
do
    Convert to lower case
    Remove stop words
```

```
                Insert into table
            end
        end
    end
```

7.2.3) Pseudo Code for Search

```
    Connect to database
    Get the query
    For query
    do
        Break the query into words using whitespace as the delimiter
        For Each Website in database
        do
            For each word in the query
            do
                Count the no of occurrences of the word
                Add count to existing count variable
            end
            Display the Website and no of occurrences
        end
    end
end
```

7.2.4) Pseudo Code for Ranking

```
Begin
    Initialise the nodes for graph.
    Initialise the PageRank as 1/n
    Distribute the PR of each page
    Calculate the new PR using the damping factor
    Normalise the page ranks so it's all a proportion 0-1
End
```

7.3) Screen Shots

7.3.1) Search Result for Doctor

doctors0. <http://www.bestdoctors.com/> (occurrences: 156)

doctors0. <http://www.drugs.com/> (occurrences: 24)

doctors0. <http://www.medicinenet.com/script/main/hp.asp#> (occurrences: 4)

Figure: 13 Search Results

7.3.2) Database

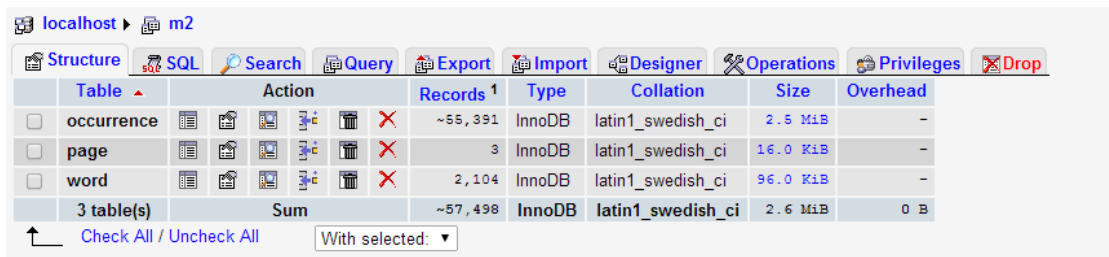
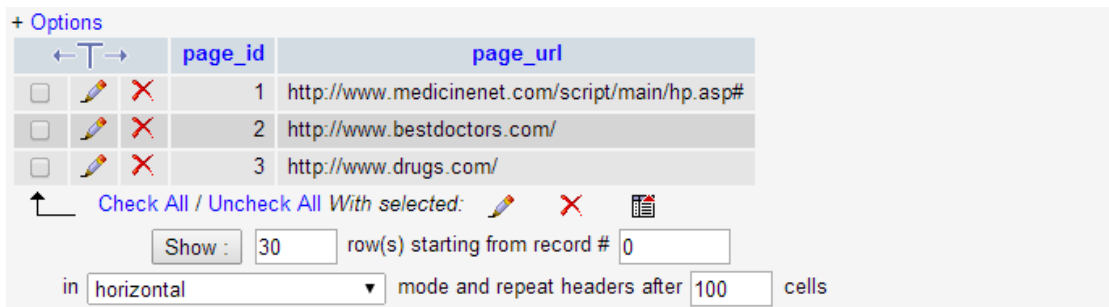


Table	Action	Records	Type	Collation	Size	Overhead
occurrence		~55,891	InnoDB	latin1_swedish_ci	2.5 MiB	-
page		3	InnoDB	latin1_swedish_ci	16.0 KiB	-
word		2,104	InnoDB	latin1_swedish_ci	96.0 KiB	-
3 table(s)	Sum	~57,498	InnoDB	latin1_swedish_ci	2.6 MiB	0 B

Figure: 14 Database

7.3.3) Page Table

Page holds all indexed web pages.



page_id	page_url
1	http://www.medicinenet.com/script/main/hp.asp#
2	http://www.bestdoctors.com/
3	http://www.drugs.com/

Figure: 15 Page Table

7.3.4) The Word Table

Word holds all of the words found on the indexed pages.

























←T→			word_id	word_word
<input type="checkbox"/>			631	locations
<input type="checkbox"/>			632	now
<input type="checkbox"/>			633	drug
<input type="checkbox"/>			634	interaction
<input type="checkbox"/>			635	potential
<input type="checkbox"/>			636	interactions
<input type="checkbox"/>			637	check
<input type="checkbox"/>			638	121
<input type="checkbox"/>			639	baby
<input type="checkbox"/>			640	catch
<input type="checkbox"/>			641	tools
<input type="checkbox"/>			642	help

Figure: 16 Word Table

7.3.5) The occurrence Table

Occurrence with page and word, we can determine which pages contain a word, as well as how many times the word occurs.



























←T→			occurrence_id	word_id	page_id
<input type="checkbox"/>			121	10	1
<input type="checkbox"/>			122	41	1
<input type="checkbox"/>			123	2	1
<input type="checkbox"/>			124	10	1
<input type="checkbox"/>			125	41	1
<input type="checkbox"/>			126	2	1
<input type="checkbox"/>			127	10	1
<input type="checkbox"/>			128	42	1
<input type="checkbox"/>			129	43	1
<input type="checkbox"/>			130	10	1
<input type="checkbox"/>			131	42	1
<input type="checkbox"/>			132	43	1
<input type="checkbox"/>			133	44	1

Figure: 17 Occurrence Table

7.3.6) Crawler Input



Figure 18: Crawler Input

7.3.7) Crawler Results

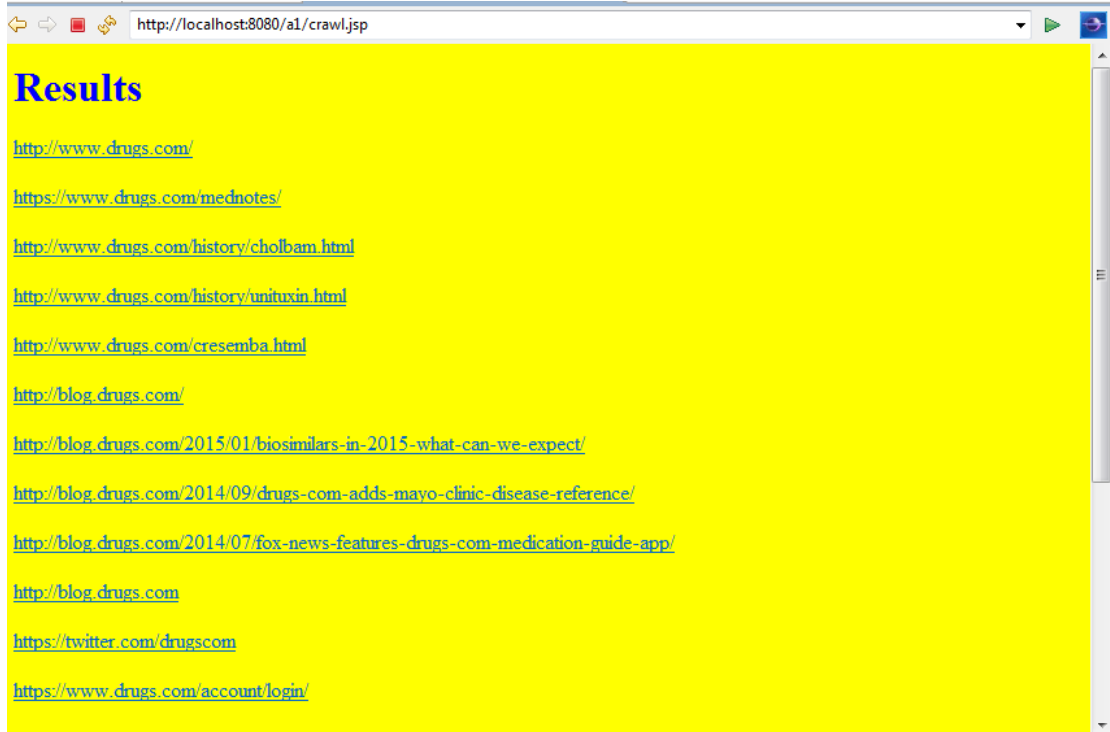


Figure 19: Crawler Result

7.3.8) Page Rank Result

```
$links = array(  
    1 => array(5),  
    2 => array(4, 7, 8),  
    3 => array(1, 3, 4, 7, 9),  
    4 => array(1, 2, 4, 8),  
    5 => array(1, 6, 7, 9),  
    6 => array(1, 5, 8),  
    8 => array(3, 4),  
    9 => array(1, 4, 6, 8)  
);
```

Figure: 20 Page Rank Input

```
array(8) {  
    [1]=>  
    float(0.17084756816354)  
    [2]=>  
    float(0.060138597093041)  
    [3]=>  
    float(0.095270626028832)  
    [4]=>  
    float(0.17308244947008)  
    [5]=>  
    float(0.20422266588834)  
    [6]=>  
    float(0.086831014221959)  
    [8]=>  
    float(0.12476184607085)  
    [9]=>  
    float(0.084845233063364)  
    ?>
```

Figure: 21 Page Rank Result

CHAPTER-8

Results

Performance of Search Engine is according to the following factors:

- Number of Search Results
- Response Time

Number of Search Results	Depends upon the number of websites Crawled and Indexed
Response Time	0.1-0.2sec

Table:2 Performance Parameters

Conclusion and Future Work

With the fulfilment of the project “Medical Search Engine” the user can now search for medical related queries by visiting our web page and the results are displayed according to the occurrences of the keywords. The user can enter its query and keywords would be extracted from the query. A number of medical websites in my database, which would be searched, indexed and the result would be the websites which will have combined occurrences of the keywords, but for the websites present in the database. Further I have also developed a crawler, which is visiting the sites and also the links inside those websites.

In Future the project can be extended when the results are being displayed according to the PageRank algorithm and a comparative study can be done of the two algorithms.

References

Research Papers

- [1] Cody Hansen, and Feifei Li “**ColumbuScout: towards building local search engines over large databases**” in SIGMOD Conference, page 617-620. ACM, (2012)
- [2] Sergey Brin and Lawrence Page “**The Anatomy of a Large-Scale Hypertextual Web Search Engine**” in Proceedings of the Seventh International World Wide Web Conference Volume 30, Issues 1–7, April 1998, Pages 107–117
- [3] Suppawong Tuarob, Prasenjit Mitra, C. Lee Giles “**Building a Search Engine for Algorithms**” in SIGWEB Newsletter Winter 2014
- [4] Steven Bird, James R. Curran “**Building a Search Engine to Drive Problem-Based Learning**” in ITiCSE’06, June 26–28, 2006, Bologna, Italy.

Reference Links

- [1] Centre for Natural Language Processing, How a Search Engine Works [online] 2001 <http://www.infotoday.com/searcher/may01/liddy.htm>(Accessed :12th August 2014)
- [2] Web Search Engine, How a Search Engine Works [online]2006 <http://www.webopedia.com/DidYouKnow/Internet/HowWebSearchEnginesWork.asp> (Accessed: 17th August 2014)
- [3]Searching the World Wide Web, Search Engines [online] 1998 <http://www.exploratorium.edu/lc/search/searchengine.html> (Accessed: 20th September 2014)
- [4] Performance Parameters, Comparing the Performance of Search Engines [online]1998 <http://moz.com/blog/comparing-search-engine-performance-how-does-cuill-stack-up-to-google-yahoo-live-ask>(Accessed: 4th October 2014)

APPENDIX

Source Code for Crawler

```
<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" %>

<% @ page import="dbConnect.DB" %>
<% @ page import="java.sql.*" %>
<% @ page import="java.io.*" %>
<% @ page import="java.lang.*" %>
<% @ page import="org.jsoup.*" %>
<% @ page import="org.jsoup.helper.*" %>
<% @ page import="org.jsoup.select.*" %>
<% @ page import = "org.jsoup.Jsoup" %>
<% @ page import= "org.jsoup.nodes.Document" %>
<% @ page import = "org.jsoup.nodes.Element" %>
<% @ page import = "org.jsoup.select.Elements" %>
<% @ page import = "org.apache.jasper.JasperException" %>
<% @ page import="org.jsoup.nodes.*" %>

<html>
<body style="background-color:yellow;">
<h1 style="color:blue;">Results</h1>
<%! static DB db = new DB();
static String url=null;
static String words=null;
static String URL1[]=new String[1000];
static int i=0;
%>
<% url=request.getParameter("url"); %>
<% if((words=request.getParameter("words"))==null)
words="medical"; %>
<% db.runSql2("TRUNCATE Record;");
```

```

processPage("http://" +url); %>
<%! public static void processPage(String URL)
{
try{
String sql = "select * from record where URL = '"+URL+"'";
ResultSet rs = db.runSql(sql);
if(rs.next()){
}else{
sql = "INSERT INTO `m2`.`page` " + "("page_url`) VALUES " + "(?)";
PreparedStatement stmt = db.conn.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);
stmt.setString(1, URL);
stmt.execute();
//get useful information
Document doc = Jsoup.connect("http://" +url+ "/").get();
if(doc.text().contains(words)){
URL1[i]=URL; i++; %>
<%! System.out.println(URL);
}
Elements questions = doc.select("a[href]");
for(Element link: questions){
if(link.attr("href").contains(".net") ||
link.attr("href").contains(".com")||link.attr("href").contains(".edu")||link.attr("href").co
ntains(".in"))
{
processPage(link.attr("abs:href"));
}
}
}
}
catch(Exception e)
{
System.out.println("exception occured"+e);
}

```



```
}  
%>  
</body>  
</html>
```

Source Code for Search

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"  
    pageEncoding="ISO-8859-1"%>  
  
<%@ page import="dbConnect.DB" %>  
  
<%@ page import="java.sql.*" %>  
  
<%@ page import="java.io.*" %>  
  
<%@ page import="java.lang.*" %>  
  
<%@ page import="org.jsoup.*" %>  
  
<%@ page import="org.jsoup.helper.*" %>  
  
<%@ page import="org.jsoup.select.*" %>  
  
<%@ page import = "org.jsoup.Jsoup" %>  
  
<%@ page import= "org.jsoup.nodes.Document" %>  
  
<%@ page import = "org.jsoup.nodes.Element" %>  
  
<%@ page import = "org.jsoup.select.Elements" %>  
  
<%@ page import = "org.apache.jasper.JasperException" %>  
  
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  
  
<html>  
  
<body>  
  
<h1 style="color:blue;">Results</h1>  
  
<%! static DB db = new DB();
```

```

String nword=null;

static String key=null;

String purl;

String[] nkey;

static String URL1[]=new String[1000];

int counter=0;

static int i=0;

int j=0;

int pid=0;

int k;

int l=0;

int m=0;

int occ=0;

%>

<% key=request.getParameter("keyword");

%>

<%

if(key!=null)

{

out.println("keyword is :"+key);

String sql = "select count(*) from page ";

ResultSet rs = db.runSql(sql);

if(rs.next())

```

```

{
    pid = rs.getInt(1);
}

for(k=1;k<=pid;k++)
{
    counter=0;

    for (String ret: key.split(" "))
    {
        String sql2 = "SELECT p.page_url AS url,COUNT(*) AS occurrences FROM page
        p, word w, occurrence o where p.page_id = o.page_id AND w.word_id = o.word_id
        AND w.word_word = '" + ret + "' AND p.page_id = '" + k + "' GROUP BY p.page_id";

        ResultSet rs2 = db.runSql(sql2);

        if(rs2.next())
        {
            occ = rs2.getInt("occurrences");

            purl=rs2.getString(1);
        }

        counter=counter+occ;
    }

    if(counter!=0)
    {
        out.println(purl);

        out.println("occurrences : "+counter);
    }
}

```

```
}  
  
counter=0;  
  
}  
  
}  
  
%>  
  
</body>  
  
</html>
```

Source Code for Indexer

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1 "  
    pageEncoding="ISO-8859-1"%>  
  
<%@ page import="dbConnect.DB" %>  
  
<%@ page import="java.sql.*" %>  
  
<%@ page import="java.io.*" %>  
  
<%@ page import="java.lang.*" %>  
  
<%@ page import="org.jsoup.*" %>  
  
<%@ page import="org.jsoup.helper.*" %>  
  
<%@ page import="org.jsoup.select.*" %>  
  
<%@ page import = "org.jsoup.Jsoup" %>  
  
<%@ page import= "org.jsoup.nodes.Document" %>  
  
<%@ page import = "org.jsoup.nodes.Element" %>  
  
<%@ page import = "org.jsoup.select.Elements" %>  
  
<%@ page import = "org.apache.jasper.JasperException" %>  
  
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" >
```

<html>

<body>

<%

String stopwords= "a about above across after again against all almost alone along
already also although always among an and another any anybody anyone anything
anywhere are area are as around as ask asked asking asks at away b back backed
backing backs be became because become becomes been before began behind being
beings best better between big both but by c came can cannot case cases certain
certainly clear clearly come could d div did differ different differently do does done
down down downed downing downs during e each early either end ended ending ends
enough even evenly ever every everybody everyone everything everywhere f face
faces fact facts far felt few find finds first for four from full fully further furthered
furthering furthers g gave general generally get gets give given gives go going good
goods got great greater greatest group grouped grouping groups h had has have
having he her here herself high high high higher highest him hi homepagecarouselx
content homepagecarouselx image homepagecarouselx list homepagecarouselx image
homepagecarouselx barousel nav homepagecarouselx thslide myself his how however i
if important in interest interested interesting interests into is it its itself j just k keep
keeps kind knew know known knows l large largely last later latest least less let lets
like likely long longer longest m made make making man many may me member
members men might more most mostly mr mrs much must my myself n necessary
need needed needing needs never new new newer newest next no nobody non noone
not nothing now nowhere number numbers o of off often old older oldest on once one
only open opened opening opens or order ordered ordering orders other others our out
over p part parted parting parts per perhaps place places point pointed pointing points
possible present presented presenting presents problem problems put puts q quite r
rather really right right room rooms s said same saw say says second seconds see
seem seemed seeming seems sees several shall she should show showed showing
shows side sides since small smaller smallest so some somebody someone something
somewhere state states still still such sure t take taken than that the their them then
there therefore these they thing things think thinks this those though thought thoughts
three through thus to today together too took toward turn turned turning turns two u

under until up upon us use used uses v very var w want wanted wanting wants was
way ways we well wells went were what when where whether which while who
whole whose why will with within without work worked working works would x y
year years yet you young younger youngest your yours z ";

```
%>
```

```
<h1 style="color:blue;">Results</h1>
```

```
<%! static DB db = new DB();
```

```
String nword=null;
```

```
static String url1=null;
```

```
int i=0;
```

```
static int wid;
```

```
static int pid;
```

```
int j=0;
```

```
int k=0;
```

```
int l=0;
```

```
int m=0;
```

```
%>
```

```
<% url1=request.getParameter("url"); %>
```

```
<%
```

```
if(url1==null)
```

```
out.println("you need to define a URL");
```

```
else
```

```
if(url1.substring(0,7)!="http://")
```

```
url1 = "http://" + url1;
```

```

%>

<% out.println(url1); %>

<%

String sql = "select page_id from page where page_url = '"+url1+"'";

ResultSet rs = db.runSql(sql);

if(rs.next()){

    pid = rs.getInt("page_id");

}

else{

sql = "INSERT INTO `m2`.`page` " + "("page_url`) VALUES " + "(?)";

PreparedStatement stmt = db.conn.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);

stmt.setString(1, url1);

stmt.execute();

String sql2 = "select page_id from page where page_url = '"+url1+"'";

ResultSet rs3 = db.runSql(sql2);

if(rs3.next()){

    pid = rs3.getInt("page_id");

}

}

%>

<%

Document doc = Jsoup.connect(url1+"/").get();

```

```

String str = doc.text().replaceAll("[^a-zA-Z ]", "");

%>

<%

for (String ret2: stopwords.split(" "))

{

if(str.contains(ret2))

{

str = str.replaceAll(" "+ ret2 + " ", " ");

}

}

%>

<%

for (String ret: str.split(" "))

{

nword=ret;

String sql3 = "select word_id from word where word_word = '"+nword+"'";

ResultSet rs2 = db.runSql3(sql3);

if(rs2.next()){

wid = rs2.getInt("word_id");

}

else{

sql3 = "INSERT INTO `m2`.`word` " + "("+word_word`) VALUES " + "(?)";

```



```

PreparedStatement stmt = db.conn.prepareStatement(sql3,
Statement.RETURN_GENERATED_KEYS);

stmt.setString(1,nword);

stmt.execute();

String sql4 = "select word_id from word where word_word = '"+nword+"'";

ResultSet rs4 = db.runSql3(sql4);

if(rs4.next()){

wid = rs4.getInt("word_id");

}

}

sql = "INSERT INTO occurrence (word_id,page_id) "

+ "VALUES (?, ?) ";

PreparedStatement stmt = db.conn.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);

stmt.setInt(1,wid);

stmt.setInt(2,pid);

stmt.execute();

}

%>

</body>

</html>

```

Source Code for Website

```

<!Doctype html>
<head>

```

```

<title>Demo</title>
<link rel="stylesheet" href="css/ex.css" type="text/css">
<link rel="stylesheet" type="text/css" href="style.css" media="screen" />
<body>
<div id="main_container">
<div class="header">
<div id="logo">
</div>
<div style="position:absolute;top:4%;left:35%;">
<h1 style="color:black;text-align:center;font-size: 65px;">MEDICINE SEARCH
</h1>
</div>
</div>
<frameset cols="25%,50%,25%">
<frame>

</frame>
<frame >

</frame>

</frame>
</frameset>
<frameset cols="25%,75%">
<frame>
</frame>
<frame >
</frame>
</frameset>
<div style="position:absolute;top:60%;left:35%;">
<form method="POST" action="search.php">
<div
<table align="center">

```

```

<tr>
<td colspan="2" align="center">
<input type="text" name="keyword" id="keyword" size="60" />
</td>
</tr>
<td>
<input type="submit" value="search" />
</td>
<td>
<input type="reset" value="Clear"/>
</td>
</table>
</form>
</div>
<br><br><br><br><br><br><br><br>
<div style="position:absolute;top:60%;left:35%;">
<h2 style="color:black;text-align:center;font-size: 20px;">
<a href="in.html" target="_blank">Indexer</a>
<br>
<a href="pagelinks.php" target="_blank">Pages Processed</a>
<br>
</h2></div>
</div>
</body>
</html>

```

Source Code for PageRank

```

<?php
$links = array(
    1 => array(5),
    2 => array(4, 7, 8),

```

```

3 => array(1, 3, 4, 7, 9),
4 => array(1, 2, 4, 8),
5 => array(1, 6, 7, 9),
6 => array(1, 5, 8),
8 => array(3, 4),
9 => array(1, 4, 6, 8)
);
?>
<?php
function calculatePageRank($linkGraph, $dampingFactor = 0.15) {
    $pageRank = array();
    $tempRank = array();
    $nodeCount = count($linkGraph);
    foreach($linkGraph as $node => $outbound) {
        $pageRank[$node] = 1/$nodeCount;
        $tempRank[$node] = 0;
    }
    $change = 1;
    $i = 0;
    while($change > 0.00005 && $i < 100) {
        $change = 0;
        $i++;
        foreach($linkGraph as $node => $outbound) {

```

```

    $outboundCount = count($outbound);

    foreach($outbound as $link) {

        $tempRank[$link] += $pageRank[$node] / $outboundCount;

    }

}

$total = 0;

foreach($linkGraph as $node => $outbound) {

    $tempRank[$node] = ($dampingFactor / $nodeCount)

        + (1-$dampingFactor) * $tempRank[$node];

    $change += abs($pageRank[$node] - $tempRank[$node]);

    $pageRank[$node] = $tempRank[$node];

    $tempRank[$node] = 0;

    $total += $pageRank[$node];

}

foreach($pageRank as $node => $score) {

    $pageRank[$node] /= $total;

}

}

return $pageRank;

}

?>

```