Project Entitled

DESIGN AND IMPLEMENTATION OF WEB-BASED SEARCH ENGINE - XPLORE

MAKHIJANI ROSHAN

MISHRA SANJAYKUMAR

MOULICK PRATHAMESH

UNDER THE GUIDANCE OF

MS. KUMKUM SAXENA

(Assistant Professor, Department of Information Technology)

DEPARTMENT OF INFORMATION TECHNOLOGY
THADOMAL SHAHANI ENGINEERING COLLEGE
UNIVERSITY OF MUMBAI
2010-2011

DESIGN AND IMPLEMENTATION OF WEB-BASED SEARCH ENGINE - XPLORE

PROJECT REPORT 'B'

Submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING

in

INFORMATION TECHNOLOGY

by

Group No:16

Roll No.	Seat No	Name of Student
46		MAKHIJANI ROSHAN
47		MISHRA SANJAYKUMAR
48		MOULICK PRATHAMESH

Under the Guidance of

MS. KUMKUM SAXENA

(Assistant Professor, Department of Information Technology)

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Department of Information Technology
Thadomal Shahani Engineering College
University Of Mumbai

2010-2011.

CERTIFICATE

This is to certify that Roshan Makhijani having Roll No. 46 has satisfactorily carried out the Project-B work entitled, 'DESIGN AND IMPLEMENTATION OF WEB- BASED SEARCH ENGINE – XPLORE' as a part of the curriculum for the degree of Bachelor of Engineering in Information Technology under University of Mumbai.

_____	_____
Ms. Kumkum Saxena (Project Guide)	Mr. Arun B. Kulkarni Assistant professor and Head, Department of IT, TSEC

DESIGN AND IMPLEMENTATION OF WEB-BASED SEARCH ENGINE - XPLORE

PROJECT REPORT 'B'

Submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING

in

INFORMATION TECHNOLOGY

by

Group No

Seat No. Name

Makhijani Roshan

Mishra Sanjaykumar

Moulick Prathamesh

_____ _____

Ms. Kumkum Saxena (External Examiner)
(Project Guide)

_____

(Internal Examiner)

_____ _____

Prof. Arun Kulkarni Dr.G.T.Thampi
(Head , Department of I.T.) (Principal)

TABLE OF CONTENTS

ABSTRACT

The project attempts to integrate the traditional approach of web search with a novel technique of process based search. Since users can belong to diverse backgrounds and have varied expectations from a query, we propose to improve the traditional search as well, using personalization based on enriched fuzzy concept networks. These networks are built based on user's profile using ontology. The main idea of the project is to provide the user with an option of traditional search, personalized to suit his/her preferences and an option of process based search. Unlike traditional web search that displays a list of ranked hyperlinks, a process based system searches and displays results based on the process. In traditional search the users may or may not get the information they want in a synopsis. A process-based search attempts to compose and frame the exact requirement of a user by organizing the results in a stepwise manner.

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

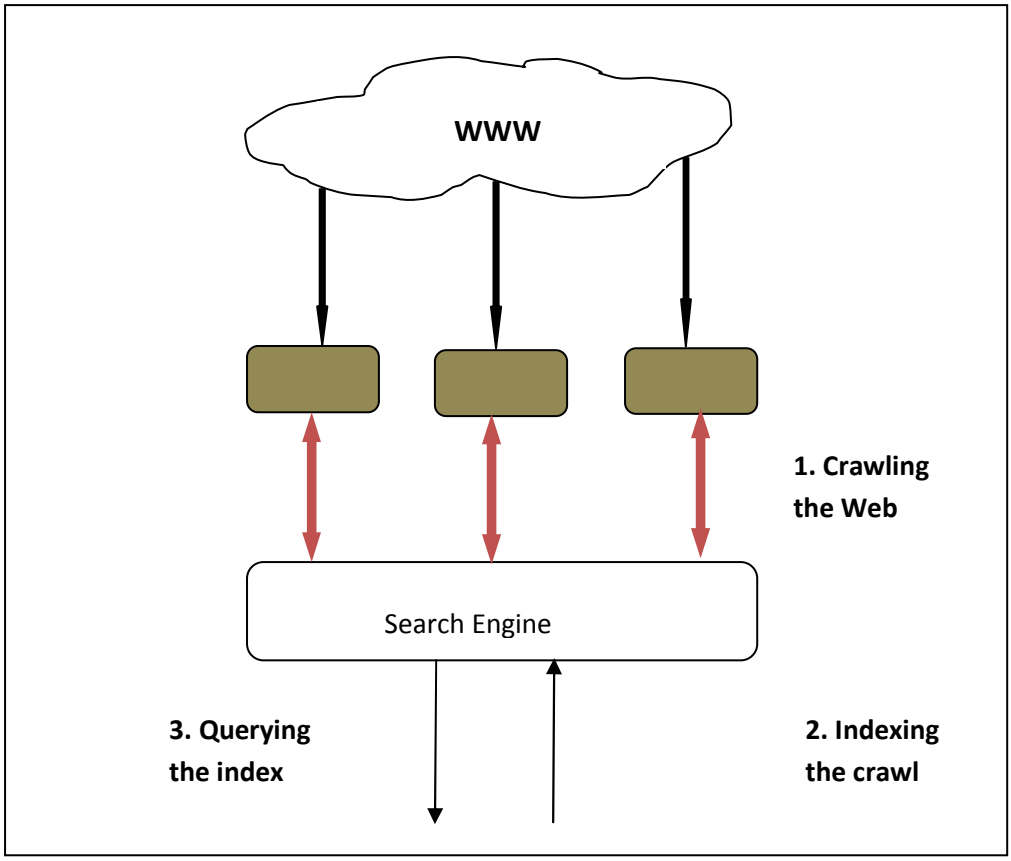

Figure 1. Search Engine Block Diagram.

A web search engine is designed to search for information on the World Wide Web. The information may consist of web pages, images, information and other types of files [5]. Unlike Web directories, which are maintained by human editors, search engines operate algorithmically or are a mixture of algorithmic and human input.

A web search engine, as shown in Figure 1 operates, in the following order:

1. Web crawling

2. Indexing

The usefulness of a search engine depends on the relevance of the result set it gives back. While there may be millions of web pages that include a particular word or phrase, some pages may be more relevant, popular, or authoritative than others. Most search engines

employ methods to rank the results to provide the "best" results first. Even today, conducting a search on any of the major search engines can be classified as an "enter your query and hope for the best" experience.

A. Semantic Web

The Semantic Web is a World Wide Web Consortium (W3C) driven major development aiming at improving the state of the World Wide Web through the use of semantic technology. The basic idea of semantic web is to enrich the current Web with machine cognitive information about the semantics of information content [7]. Semantic search supplies the different results to the keyword search.

Ontology for Semantic Web

An ontology is a set of assertions specifying the concepts involved in the domain. By using ontology to organize text documents, knowledge can be represented in a way that facilitates understanding between computers and humans. The Semantic Web usually refers to a knowledge-based system as the addition of information about the terminology, providing concept and properties descriptions, and assertions about particular instances of classes. Ontology is used to formally specify this information [3].

B. Crawling

A Web crawler is a computer program that browses the World Wide Web in a methodical, automated manner or in an orderly fashion as shown in Figure 2. This process is called Web crawling or spidering [4]. Many sites, in particular search engines, use spidering as a means of providing up-to-date data. Web crawlers are mainly used to create a copy of all the visited pages for later processing by a search engine that will index the downloaded pages to provide fast searches.

C. Personalization

The personalization process extracts users' preference and builds a profile for every individual user, and then it provides immediate response by filtering and re-ranking the results respect to individual user's interests. The imprecise and uncertain information comes from three major aspects in an information retrieval system environment including the representations of users' queries, the representations of documents, and the relevance relationships between users' queries and documents. In order to represent and process the

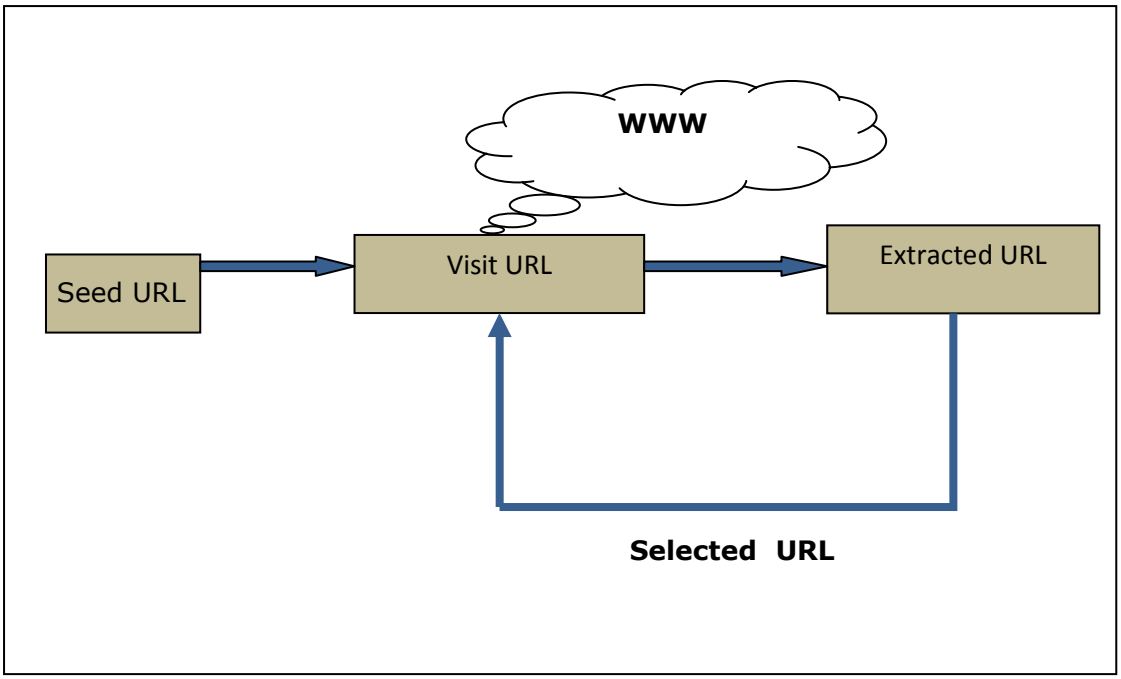

Figure 2. Working of a Crawler.

imprecise and uncertain information in information retrieval systems, the fuzzy set theory has been applied to information retrieval systems. In this context, fuzzy concept networks are very used in personalizing search engines. Our main idea is to employ the concepts of ontology in order to enrich the common fuzzy concept networks built based on user's profile.

D. Process Based Search

Consider a user who intends to search on the Web about how to achieve a goal, such as “Move to another city”, “Drive a car”, or “Attend school abroad”. Current search engines may return a list of related articles discussing the goal, if the user is lucky. Next, the user will have to spend considerable time to read, compare, and to investigate several individual Web pages to understand exactly how many steps and in what order these steps should be performed for achieving the goal. These steps could be frustrating, especially when the user is busy.

In a Process Based Search system [2], an example scenario of a common search is as follows:

(1) Michael just moved to the city of Lawrence. He needs to be able to drive a car to move around this city without a public transportation system. He is also totally new to this country and has no friends yet. He opens a Web browser to go to the Process Based Search system website. He types in “How to drive a car” in the Web page and presses the Search button.

(2) The system displays the matched processes as follows:

Process 1: How to drive a car?
Step 1: Go to a driving school
<links>
Step 2: Take the driver’s license test
<links>
Step 3: Buy a car
<links>
and so on..
(3) Michael chooses the first process, checks and schedules every step, and then fills in some online applications and reservation forms. He saves the process he chose and also the progress, in order to be able to continue next time.

1.1 Problem Definition

The field of information retrieval has become an integral part of the life of an average internet user. Almost every person that uses the internet today uses a search engine for information. Therefore the information on the World Wide Web needs to be reorganized and presented in a more useful manner to the user.

In our project a meta-search engine is developed that queries a well established search engine infrastructure. A Graphical User Interface (GUI) is developed for querying the Web. In the GUI we give the user the option of querying traditionally or using personalization based on his/her profile or searching for a process.

In traditional search, the results are displayed according to the rank of the document in accordance with the query. In today’s day and age where there is an explosion of information there is a need to add a semantic dimension to the web. With personalization, the traditional search is modified according to the search requirements gleaned from the user profile.

Therefore personalization helps users find information relevant to their needs. Using the personalization process we build a user profile for every interested user and then filter and re-rank the search results according to his/her preferences.

In process-based search the user queries the search engine normally, but the results of the query are displayed in the form of a process i.e. as a series of steps. If a query suitable for process-based search is fed to the traditional engine then the user will have to spend a considerable amount of time to read, compare and coagulate all the information necessary to understand the process and the steps required for its completion. The processes and their corresponding information are stored in the process database. The process data can be collected in two ways. The first way is to manually feed the process and its steps, along with the URLs for the steps. Any user can add, modify or delete a process or its steps, subject to supervision by the administrators. A Graphical User Interface is developed for manual control of the process repository. Another way is to crawl through the WWW and automatically extract processes and their steps, but processes may be sparsely distributed on the Web. Therefore in order to extract implicit processes; a topical crawler is used that only crawls websites which specialize in process related content.

1.2 Relevance of the Project

Thousands of users issue queries to the web search engines to find their required information. Since the users may have diverse backgrounds and may have different expectations for a given query, some search engines try to personalize their results to better match the overall interests of an individual user. Therefore the personalization of search engines can efficiently help users to find information to their relevant needs.

In business society, process has always been an important part of enterprise knowledge. As human society becomes more and more complicatedly managed, not just in the business world, part of everyday life has also gradually begun to follow certain processes. Most of the information about processes exists on the Internet; though they may not be stored in the way that computers can deal with them directly. And many of the processes have their services or part of them available online. People can search, browse, or directly process them through the Internet. New ways of doing things are continually invented by service vendors, allowing users to take advantage of these opportunities provided by the Internet. However, up till now

no search engine has been focused on managing and utilizing the knowledge of processes which exist on the Web. The situation leaves the users with the responsibility to provide search engines with more information than necessary or possible, if the users are intending to find out how to solve a problem which may involve a series of activities.

A system of *Process-Based Search Engine* reduces searching time and improves searching quality for such situations. Unlike traditional Web search engines showing a list of ranked hyperlinks as results, the proposed system searches and displays results based on the process.

1.3 Project Scope

In our project 'Design and Implementation of a Web-Based Search Engine - Xplore' we initially develop a meta-search engine that queries the Google search engine. A GUI is also developed for displaying the search results from the engine. We need to develop a database where user profiles and their preferences are stored and managed. An add-on is developed for the Mozilla Firefox browser that helps in capturing the user profile. This is used for the personalization of the search results. Then we develop a process database for storing the process related content i.e. the process repository. We also need to develop the GUI for manually controlling the process database. Access to the database is given to almost every user, subject to authority of the administrators, much like the Wiki concept. The process crawler is also developed which is basically a topical crawler that crawls process related websites. The query fired in the process-based search engine is first fed to the traditional search engine, the query is then analyzed using the process repository and search results are displayed by filtering and grouping them according to the analysis. Finally a GUI is developed to display the results of the process based search engine.

1.4 Motivation

The arena of search engines is highly developed and advanced. However there are possibilities of improvement where relevance and conciseness of the results are concerned. The attempt is to improve the results, so that the user spends less time navigating through the plethora of results and finding what is useful.

There are two ways in which the relevance can be improved – either through tweaking results based on the previous browsing patterns of the user or by mining and utilizing the inherent process based nature of many queries. The user should be presented with results that are uncluttered and easy to comprehend. Also the speed of a search engine not only depends on the time it takes to return results but also the time it takes for the user to browse through those results and be satisfied with the information.

2. REVIEW OF LITERATURE

As of the year 2010, the World Wide Web around contains 14.68 billion pages; it is widely believed that “indexing the whole of the Web” is already impractical or would soon become so due to its exponential growth. But currently, the GYM search engines—Google, Yahoo!, and Microsoft—are indexing a huge amount of data and between them providing reliable sub second responses to around a billion queries a day in a plethora of languages. If this were not enough, the major engines now provide much higher quality answers. For most searchers, these engines do a better job of ranking and presenting results, respond more quickly to changes in interesting content, and more effectively eliminate dead links, duplicate pages, and off-topic spam. Search engines cannot and should not index every page on the Web. After all, due to dynamic Web page generators such as automatic calendars, the number of pages is infinite. Basically a search engine consists of front-end process and back-end process. As shown in Figure 3, there are three software components in the back-end process,

Crawler/Spider: A crawler crawls around the web, visits web pages through their linkages, and downloads the page content to the local database of the search engine [8] [13].

- Indexer: An indexer transfers a page into various components, typically keywords, and analyzes them. Entities such as, titles, headings, links, text, constructs, bold, italic, and other style portions of a page are ripped apart and analyzed [5] [12].
- Index file database: The index file database is a category containing a copy of every single page that the crawler scans and the corresponding index information generated by the indexer. When a crawler returns to your website and scans your website, this catalog will be updated [5]. There are also three software components in the front-end process,
- Query parser: This is the software used to analyze the queries input from users by transferring the query into the same representation as document in the indexer. In this way, a comparison can be done between the query and each document.

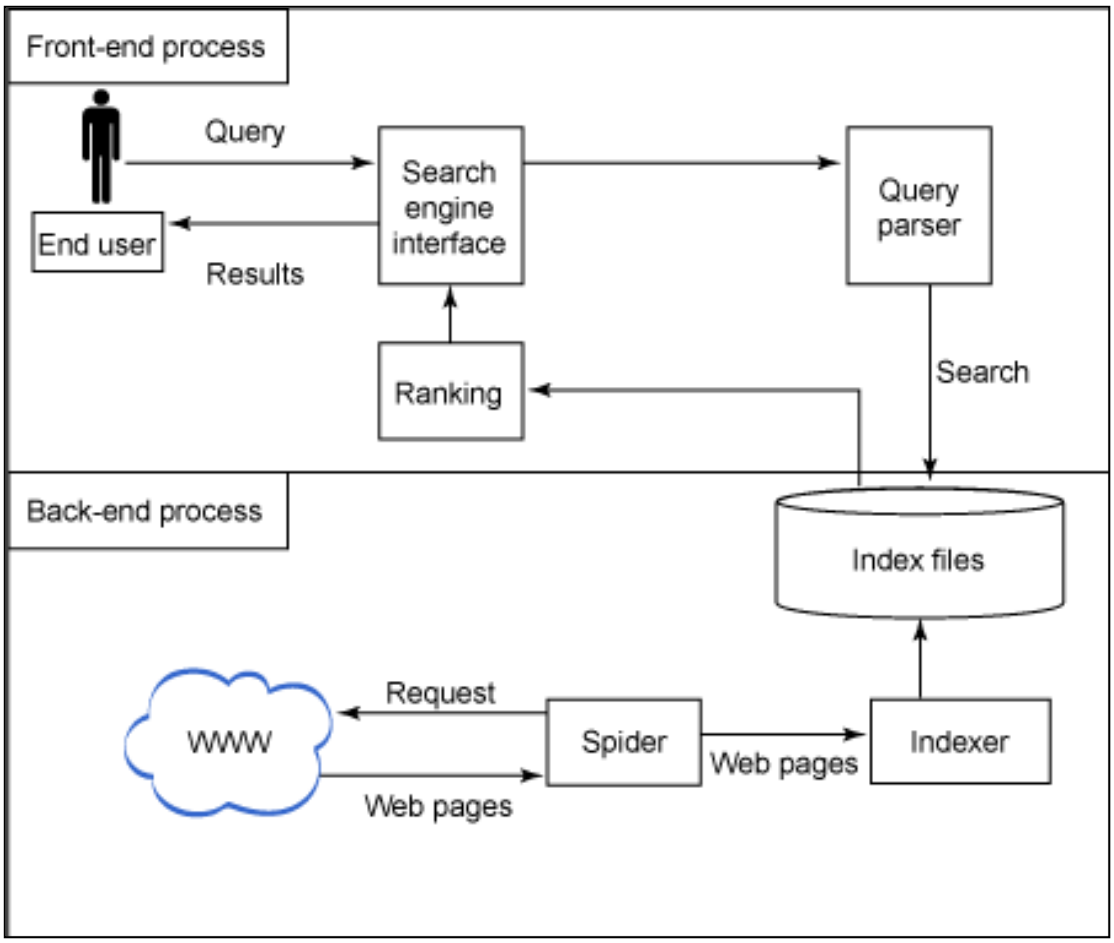

Figure 3. Generic Search Engine Architecture.

- Ranking mechanism: This is the algorithm that matches the users' keywords with the pages in the database and comes up with matches. These matches are then displayed on the users' screen in the decreasing order of their relevance to the users' input query.
- Search engine interface: This is the portion of a search engine a user interacts with when he/she performs a search and gets the returned results.

Based on these components, the search engine basically compiles responses to user queries by accessing a local repository that mirrors the Web. When a user submits a query, usually in the form of a list of textual terms, an internal scoring function is applied by the indexer to each webpage in the repository. Applying the scoring function to a webpage produces a numerical score, representing the best available estimate of the usefulness of the webpage to the user who submitted the query [6]. Query results are usually presented in the form of a sequential list of links to web pages arranged in descending order of score.

2.1 Meta-Search Engines

A meta-search engine is a search tool that sends user requests to several other search engines and/or databases and aggregates the results into a single list or displays them according to their source. Metasearch engines enable users to enter search criteria once and access several search engines simultaneously. Metasearch engines operate on the premise that the Web is too large for any one search engine to index it all and that more comprehensive search results can be obtained by combining the results from several search engines. This also may save the user from having to use multiple search engines separately [15].

Metasearch engines create what is known as a virtual database. They do not compile a physical database or catalogue of the web. Instead, they take a user's request, pass it to several other heterogeneous databases and then compile the results in a homogeneous manner based on a specific algorithm.

2.1.1 Google SOAP API

The SOAP Search API was created for developers and researchers interested in using Google Search as a resource in their applications. Developers write software programs that connect remotely to the Google SOAP Search API service. Communication is performed via SOAP, an XML-based mechanism for exchanging typed information [16].

Developers can issue search requests to Google's index of billions of web pages and receive results as structured data, access information in the Google cache, and check the spelling of words. The SOAP Search API is limited to applications not for commercial use. Google provides each developer who registers to use the Google SOAP Search API service a limit of 1,000 queries per day.

2.1.2 JSON API

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language. JSON is a text format that is completely language independent [17].

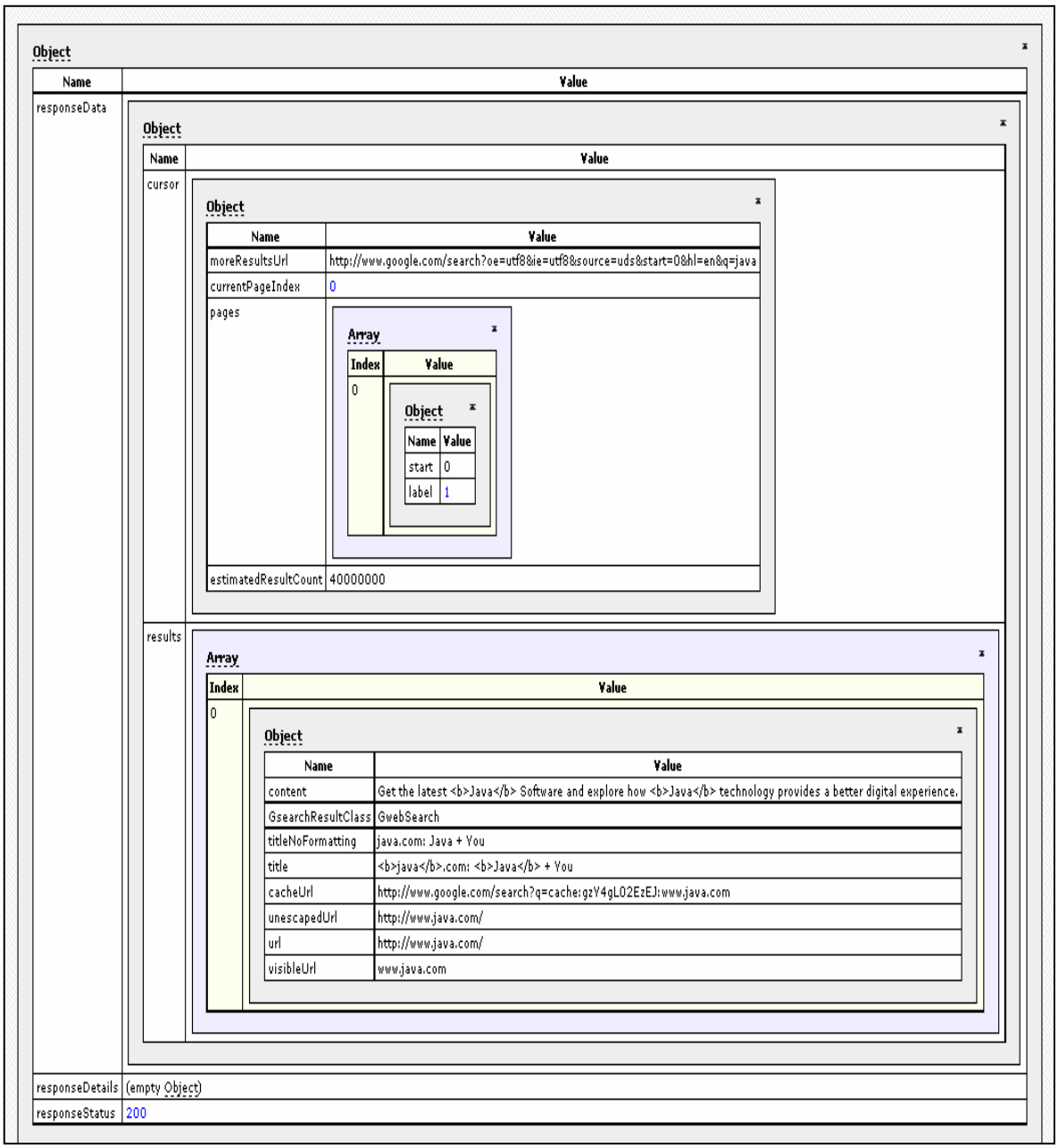

Figure 4. JSON to HTML for the query 'Java'.

JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.

The search results for the query ‘Java’ from the Google SOAP Search API are fed to the JSON object where each result is formatted as shown in Figure 4 [18].

2.2 Process Base Development

The designing and programming phases of the Process Based Search Engine (PBSE) system are the beginning of its development lifecycle. After it is released to public, it will rely on Internet users’ participation (co-development) to become mature. And it is true that only a small percentage of users are willing and able to contribute to a Web application. But the very few people built Linux, and people built Wikipedia. People search for everything on the Internet. People are more willing to actively participate on the Web than to spend time passively doing other tasks such as watching television. Therefore, we believe that the goal of our system is achievable and it will be useful to the public.

2.2.1 The Wiki Concept

The Wiki concept was invented by Ward Cunningham. It is a collection of ideas and technologies for building Web-based collaborative discussion servers. It is based on the simple idea of “open editing”, allowing everyday users to create and edit any pages on a Wiki Web site. The most well-known Wiki Web site is the free online encyclopedia: Wikipedia. In spite of its simplicity, the fact that it has been widely accepted has attracted academic interests in Wiki technology, especially Wikipedia.

It is difficult to extract process data from Wikipedia or other existing online sources. We could not identify an existing Wiki Website which collects process data. The wikiHow is the closest to the Process Base introduced, but it was designed for collecting how to do things types of articles, most of which do not necessarily relate to online information or actions.

2.2.2 eHow

The “Process Base” of the proposed system is close to the Website of eHow [2] [14]. Its extension, called wikiHow is a Wiki website. The Website of wikiHow accumulates “How to” articles from Internet contributors, and then provide keyword searching about “How to” do things. Different from eHow and wikiHow, our goal is not to build a free encyclopedia of processes. We target to assist users searching and gathering information or achieving their goals online. Unlike these two sites which present users with static data of how to do things, we search the Internet and provide users more trustful and updated information for every step in a process. PBSE is better suited to solve the type of problems which involve online actions or information.

2.3 Natural Language Processing

Natural Language Processing (NLP) is the computerized approach to analyzing text that is based on both a set of theories and a set of technologies. And, being a very active area of research and development, there is not a single agreed-upon definition that would satisfy everyone, but there are some aspects, which would be part of any knowledgeable person’s definition [19].

Natural Language Processing is a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications.

2.3.1 OpenNLP

Apache OpenNLP is a machine learning based toolkit for the processing of natural language text. These tasks are usually required to build more advanced text processing services. OpenNLP also hosts a variety of java-based NLP tools which perform sentence detection, tokenization, pos-tagging, chunking and parsing, named-entity detection, and co-reference using the OpenNLP Maxent machine learning package [20].

A. Tokenizer

The OpenNLP Tokenizers segment an input character sequence into tokens. Tokens are usually words, punctuation, numbers, etc.

OpenNLP offers multiple tokenizer implementations:

- Whitespace Tokenizer - A white space tokenizer, non white space sequences are identified as tokens.
- Simple Tokenizer - A character class tokenizer, sequences of the same character class are tokens.
- Learnable Tokenizer - A maximum entropy tokenizer detects token boundaries based on probability model.

With OpenNLP, tokenization is a two-stage process: first, sentence boundaries are identified, and then tokens within each sentence are identified [21].

B. Part of Speech Tagger

The Part of Speech (POS) Tagger marks tokens with their corresponding word type based on the token itself and the context of the token. A token can have multiple pos tags depending on the token and the context. The OpenNLP POS Tagger uses a probability model to guess the correct pos tag out of the tag set. To limit the possible tags for a token a tag dictionary can be used which increases the tagging and runtime performance of the tagger [22].

2.4 Fuzzy Concept Networks

Imprecise and uncertain information comes from three major aspects in an information retrieval system environment including the representations of users' queries, the representations of documents, and the relevance relationships between users' queries and documents. In order to represent and process the imprecise and uncertain information in information retrieval systems, the fuzzy set theory has been applied to information retrieval

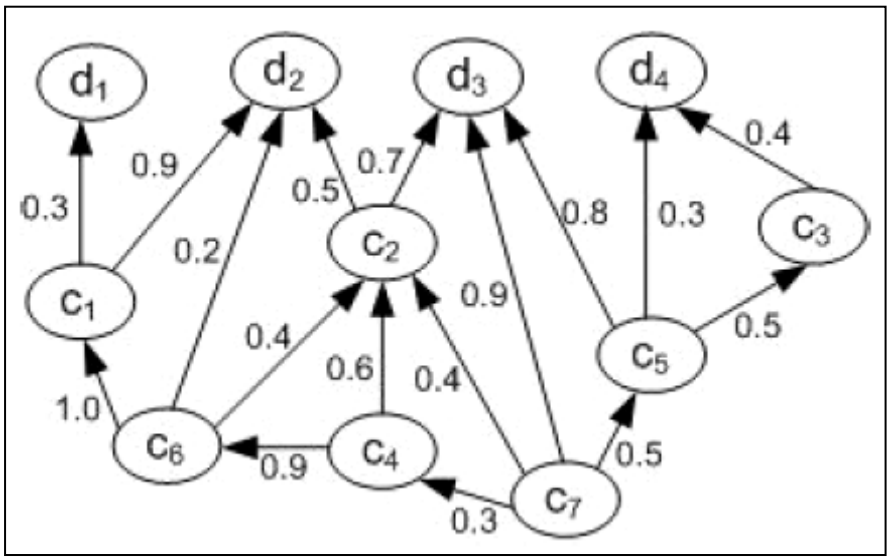

Figure 5. Fuzzy Concept Network.

systems. In this context, fuzzy concept networks are very used in personalizing search engines [1]. A fuzzy concept network used in information retrieval systems includes nodes and directed links, where each node represents a concept $c_i \in$ C or a document $d_j \in$ D, each directed link connects two concepts or connects one concept c_i to one document d_j and is labeled with a real value between zero and one. Figure 5 shows an example of fuzzy concept network. Such a fuzzy concept network helps us to understand the degree of relevance between two concepts and also the relevance degree of each document with respect to a specific concept.

If $c_i \xrightarrow{\mu} c_j$, then degree of relevance from concept c_i to concept c_j is μ. If $c_i \xrightarrow{\mu} d_j$, then μ indicates that the degree of relevance of document d_j with respect to concept c_i is μ, where $\mu \in [0,1]$ as shown in Eq.2.1. Expression $c_i \xrightarrow{\mu} c_j$ is represented with $f(c_i, c_j) = \mu$ and expression $c_i \xrightarrow{\mu} d_j$ is represented with $g(c_i, d_j) = \mu$ where f and g are mapping functions. In a concept network a document has a different relevance value with respect to each concept. For each document d_j, the document descriptor I_{dj} on the basis on the binary indexing relation I, is a fuzzy subset of C defined as follows:

$$I = \begin{array}{l} \{\mu_I(d,c),(d,c) \mid d \in D, c \in C\} \\ where\ \mu_I : D \times C \rightarrow [0,1] \end{array} \qquad \textbf{Eq. 2.1}$$

where μ_I is a membership function and indicates the degree of relevance of document d with respect to concept c [1]. Therefore the document descriptor matrix H is defined as follows:

$$\begin{bmatrix} h_{11} & h_{12} & \Lambda & h_{1n} \\ h_{21} & h_{22} & \Lambda & h_{2n} \\ \mathrm{M} & \mathrm{M} & \Lambda & \mathrm{M} \\ h_{m1} & h_{m2} & \Lambda & h_{mn} \end{bmatrix}$$
$$h_{ij} = I_{dj}(c_i), 1 \le i \le m, 1 \le j \le n$$

Eq. 2.2

C= { c_1, c_2, …, c_n } is a set of concepts. A fuzzy concept matrix K is a matrix where in K_{ij} $\in$ [0, 1]. The (i,j) element of K represents the degree of relevance from concept c_i to concept c_j.

K^2 = K$\otimes$ K is the multiplication of the concept matrix.

$$K^2{}_{ij} = \bigvee_{l=1}^{n} (k_{il} \wedge k_{lj}), \quad 1 \le \mathrm{i}, \mathrm{j} \le \mathrm{n}$$

Eq. 2.3

$\vee$ and $\wedge$ represent the max operation and the min operation, respectively. Then, there exists an integer $\rho \le$ n-1, such that $K^{\rho} = K^{\rho+1} = K^{\rho+2} = \ldots$. Let $K^* = K^{\rho}$. K* is called the transitive closure of the concept matrix K. Missed information of fuzzy concept network can be inferred from the transitive closure of itself.

The relevance degree of each document, with respect to a specific concept, can be improved by computing the multiplication of the document descriptor matrix H and the transitive closure of the concept matrix K as follows:

$$H^* = H \otimes K^*$$

Eq. 2.4

H* is called the expanded document descriptor matrix.

2.5 Ontology

An ontology is a set of assertions specifying the concepts involved in the domain. By using ontology to organize text documents, knowledge can be represented in a way that facilitates understanding between computers and humans [3]. The Semantic Web usually refers to a knowledge-based system as the addition of information about the terminology, providing concept and properties descriptions, and assertions about particular instances of classes. Ontology is used to formally specify this information.

2.5.1 WordNet

Word Net is a large lexical database of the English language where words (separated into the parts of speech: nouns, verbs, adjectives, and adverbs) are linked via semantic relationships. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations [26].

2.5.2 Java API for WordNet Searching (JAWS)

The Java API for WordNet Searching (JAWS) is an API that provides Java applications with the ability to retrieve data from the WordNet database [23]. It is a simple and fast API that is compatible with both the 2.1 and 3.0 versions of the WordNet database files and can be used with Java 1.4 and later [24].

2.6 Add-on development

Extensions are packaged and distributed in ZIP files or Bundles, with the XPI (pronounced “zippy”) file extension [25].

The content of the Interceptor extension is a typical XPI file with the following structure:

Interceptor\
    install.rdf
    chrome.manifest

```
chrome\
  locale\
    browserOverlay.dtd
    browserOverlay.properties
  content\
    browserOverlay.xul
    browserOverlay.js
  skin\
    browserOverlay.css
```

2.6.1 JavaScript

JavaScript is an implementation of the ECMAScript language standard and is primarily used in the form of client-side JavaScript, implemented as part of a web browser in order to provide enhanced user interfaces and dynamic websites. This enables programmatic access to computational objects within a host environment [27]. A web browser is by far the most common host environment for JavaScript. Web browsers typically use the public API to create "host objects" responsible for reflecting the DOM into JavaScript.

2.6.2 XUL

XUL (XML User Interface Language) is Mozilla's XML-based language that helps build feature-rich cross platform applications that can run connected or disconnected from the Internet [28]. Programmers typically define a XUL interface as three discrete sets of components:

1. *content*: the XUL document(s), whose elements define the layout of the user interface
2. *skin*: the CSS and image files, which define the appearance of an application
3. *locale*: the files containing user-visible strings for easy software localization

XUL serves primarily for constructing Mozilla applications and their extensions.

2.7 Existing Systems and Comparisons

Modern web search engines are complex software systems using the technology that has evolved over the years to give relevant results to users. Although the exact details of search engines' methods are commercially hidden (especially concerning results ranking), their mode of operation is broadly known.

2.7.1 Existing Systems

We discuss 3 search giants: *Google, Yahoo & MSN* [9-11]. The backbone of all major search engines is the same involving three vital stages: crawling, indexing and searching. However the various search techniques adopted by different search engines differ in the way these 3 stages are implemented.

A. Google

Google Search or Google Web Search is a web search engine owned by Google Inc. and is the most-used search engine on the Web. Google receives several hundred million queries each day through its various services. The main purpose of Google Search is to hunt for text in web pages, as opposed to other data, such as with Google Image Search. Google search was originally developed by Larry Page and Sergey Brin in 1997. Google search consists of a series of localized websites. The largest of those, the google.com site, is the top most-visited website in the world. Some of its features include a definition link for most searches including dictionary words, the number of results you got on your search, links to other searches (e.g. for words that Google believes to be misspelled, it provides a link to the search results using its proposed spelling), and many more.

B. Yahoo

Yahoo! Search is a web search engine, owned by Yahoo! Inc. and was as of December 2009, the 2nd largest search engine on the web by query volume, at 6.29%, after its competitor Google at 85.35% and before Bing at 3.27%, according to Net Applications.

Originally, Yahoo! Search started as a web directory of other websites, organized in a hierarchy, as opposed to a searchable index of pages. In the late 1990s, Yahoo! evolved into a full-fledged portal with a search interface and, by 2007, a limited version of selection-based search.

Yahoo! Search, originally referred to as *Yahoo! provided Search interface*, would send queries to a searchable index of pages supplemented with its directory of sites. The results were presented to the user under the Yahoo! brand. Originally, none of the actual web crawling and storage/retrieval of data was done by Yahoo! itself. In 2001 the searchable index was powered by Inktomi and later was powered by Google until 2004, when Yahoo! Search became independent.

C. MSN

MSN Search was a search engine by Microsoft that comprised a search engine, index, and web crawler. MSN Search first launched in the third quarter of 1998 and used search results from Inktomi. In early 1999, MSN Search launched a version which displayed listings from Looksmart blended with results from Inktomi except for a short time in 1999 when results from AltaVista were used instead. Since then Microsoft upgraded MSN Search to provide its own self-built search engine results, the index of which was updated weekly and sometimes daily. The upgrade started as a beta program in November 2004, and came out of beta in February 2005. Image search was powered by a third party, Picsearch. The service also started providing its search results to other search engine portals in an effort to better compete in the market.

2.7.2 Comparison

A. **Ranking algorithm:**

Google determined relevancy primarily on PageRank algorithm. PageRank essentially says that a site that has more inbound links than their competitors is likely a better site, therefore should rank higher.

Yahoo! Search’s Algorithm is not far from Google Algorithm but different at some points, Yahoo gives much interest in taking its web directory as part of its Ranking Algorithm.

MSN's Live Search has a poor relevancy algorithm. New sites that are generally untrusted in other systems can rank quickly in MSN Search.

B. Hit Count Estimation:

The hit count estimate of Google, Yahoo! and MSN Live search correlate significantly, with Google and Live Search having a particularly high value but with Yahoo! correlating less with both Google and Live Search. The reason for the different Yahoo! results was that Yahoo! automatically corrected some apparent user errors.

C. Page Content:

Google heavily biases search results toward informational resources.

Yahoo! offers a paid inclusion program, so when Yahoo! Search users click on high ranked paid inclusion results in the organic search results Yahoo! Profits. Their results are more biased toward commerce.

MSN places too much weight on page content. Their poor relevancy algorithms cause a heavy bias towards commercial results.

D. Crawling ability:

Google is most efficient at crawling ability than competing engines.

Yahoo! is pretty good at crawling sites deeply so long as they have sufficient link popularity to get all their pages indexed.

MSN is nowhere near as comprehensive as Yahoo! or Google at crawling deeply through large sites.

E. Query processing:

Google is much better than Yahoo! or MSN at determining the true intent of a query. It does concept matching. Google's search results are biased toward informational websites.

Yahoo! seems to be more about text matching when compared to Google.

MSN might be a bit better than Yahoo! at processing queries for meaning instead of taking them quite so literally.

F. Link reputation & site age:

Google is much better as being able to determine the difference between real, editorial citations and low quality, spammed, bought, or artificial links. Older sites are trusted more.

In Yahoo! it's still easy to manipulate using low to mid quality links and somewhat to aggressively focused anchor text. It places some weight on older sites.

MSN is as good as other major search engines at telling the difference between real organic citations and low quality links. MSN ranks new sites higher due to link bursts.

3. DESCRIPTION

Today, with the continued rapid growth in web information volume, information access and knowledge management has become challenging. That is why personalization is essential. The personalization process extracts users' preference and builds a profile for every individual user, and then it provides immediate response by filtering and re-ranking the results with respect to individual user's interests.

The web searching technologies have changed from using existing techniques of information retrieval to Web characterized searching. Information extracted from hypertext has improved the quality of the large-scale title searching. The goal of search engines is finding dozens of links and hoping some of them would satisfy the users.

However, as the Internet keeps changing people's life styles, there has been a need for more sophisticated searching than broad topic search. Process-based search is one of these important information seeking needs. Being aware of the abundance of Web information, searching users generally believe that the information they need exists somewhere on the Web. Search engines expect longer and more specific queries if users are trying to find information about a relatively complicated process.

But it is not reasonable to expect the users to know prepared information before searching. The goal of our system is to address these searching issues.

3.1 Proposed System

In our project 'Design and Implementation of a Web Based Search Engine - Xplore', a search engine is developed which uses the results of Google SOAP API in conjunction with JSON. A GUI is also developed for displaying the search results from the engine. An add-on named Interceptor for the Mozilla Firefox browser has been created to build user profiles. A database is developed where the user profiles and their preferences are stored and managed. This is used for the personalization of the search results. Then we develop a process database for storing the process related content i.e. the process repository. A GUI has been designed for manually controlling the process database. Access to the database is given to almost every user, subject to authority of the administrators, much like the Wiki concept. The process

extractor is also developed which is basically a topical crawler that crawls process related websites (eHow). The query fired in the process-based search engine is first fed to the traditional search engine, the query is then analyzed using the process repository and search results are displayed by filtering and grouping them according to the analysis. Finally a GUI is developed to display the results of the process based search engine.

3.2 Methodology and Analysis

The entire implementation (future work and that already implemented) can be divided in the following parts:

A. Personalization based Search Engine
B. Process Based Search Engine

3.2.1 Personalization Based Search Engine

A. Creating and Maintenance of User Profile

We used the automatic fuzzy concept networks for personalizing search results. The proposed method consists of two stages that are described in following. In the first stage, the user profile including web pages that already visited, are collected. After preprocessing on the profile, a fuzzy concept network is automatically generated for it according to predefined concepts vector V. The fuzzy concept network represents the degree of relevance between the concepts considering user's profile. The degree of relevance between concepts c_i and c_j is computed by Eq. 3.1 as follows [1]:

$$\mu(c_i, c_j) = \frac{f_{c_i} + f_{c_j} - | f_{c_i} - f_{c_j} |}{N} \qquad \textbf{Eq. 3.1}$$

Where f_{c_i} is the number of times the concept c_i appears in the user's profile and f_{c_j} is the number of times the concept c_j appears in it. N is the number of total concepts in the profile. $\mu(c_i, c_j)$ indicates ratio of the number of joint times the concepts c_i and c_j

appear in user's profile that is degree of relevance between each pair of concepts considering the profile. For enriching the fuzzy concept network, we aim to import the WordNet ontology and used it for getting stem and synonyms of each word in user's profile.

We enriched the user's profile and computed new profile using concepts of ontology as follows:

$$w_i' = \{ w_i \cup \text{synset}(w_i) \}$$
$$\text{enriched profile} = \cup w_i' \qquad \textbf{Eq. 3.2}$$

Then we generated the fuzzy concept network for enriched profile using Eq. 3.2. Here, the fuzzy concept network generated based on enriched profile is called enriched fuzzy concept network.

In the next stage, a user's query is posed to the search engine and search engine returns a list of results including the retrieved web pages. The document descriptor matrix is generated for search engine results. This matrix represents the degree of relevance between concepts and retrieved documents and is calculated using tf-idf (term frequency inverse document frequency) for each pair of concept and document as follows:

$$\text{tf-idf}(c_i, d_j) = \frac{\text{number of times the concept } c_i \text{ appears in the document } d_j}{\text{number of times the concept } c_i \text{ appears in total retrieved documents}} \qquad \textbf{Eq. 3.3}$$

Then ranking algorithm computes the transitive closure of the enriched fuzzy concept network and multiplies it by the enriched document descriptor matrix. The ranking algorithm also sums the elements at each row of the result matrix to obtain the ranking of each document and reorders documents based on their rankings.

B. Personalized Search and Re-ranking

In the next stage, a user's query is posed to the search engine and search engine returns a list of results including the retrieved web pages. The document descriptor matrix is

generated for search engine results. This matrix represents the degree of relevance between concepts and retrieved documents and is calculated using tf-idf (term frequency inverse document frequency) for each pair of concept and document. Then ranking algorithm computes the transitive closure of the enriched fuzzy concept network and multiplies it by the enriched document descriptor matrix. The ranking algorithm sums the elements at each row of the result matrix to obtain the ranking of each document and reorders documents based on their rankings.

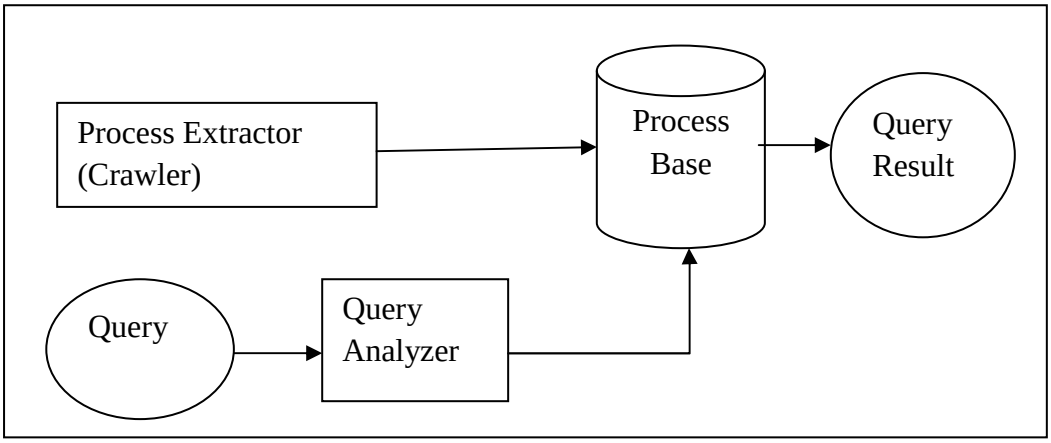

Figure 6. Prototype Process Based Search Engine.

3.2.2 Process Based Search Engine

A process-based search engine uses process data in a process repository to transform a query into a sequence of sub-queries, and then uses a traditional search engine to search each sub-query, and finally integrates results and presents them to the user in the format of a process breakdown [2].

A. Process Extractor

The documents collected by the crawler are sent to the process extractor. The process extractor already has manual rules fed into it, in order to extract processes from particular websites. The current version of the process extractor only extracts processes from the eHow website.

B. Process Base

A database of processes and a group of programs which provide means to collect and access the process data. The obtained processes are then annotated and stored in a relational database, called Process Base.

C. Query Analyzer

Given a query, a Query Analyzer first checks where the query is to be sent: traditional search engine or the process based search engine as illustrated in Figure 8. For the latter, the query is transformed into a sequence of sub-queries by matching the

```
Function: query_operations
Input: Query q
Output: A sequence of sub-queries if q is goal-oriented, NULL
otherwise
Begin
annotationq = query_analyzer(q)
if annotationq is NULL then
return NULL
else then
process_list =
process_searcher(q, annotationq)
if process_list is NULL then
return NULL
else then
p = one process in process_list
sub_queries = query_transformer(q,annotationq, p)
return sub_queries
End
```

Figure 7. Query Operations for Process Based Searches.

semantics of the query with the process annotations in the process base.

Obtained sub-queries are then sent to the process-based search engine as shown in Figure 9. Many queries submitted to a search engine are topic searches, which do not require process-based searches.

Users can be expected to be aware of the existence of a process-based search engine and manually select it. In addition, the choice to switch to a traditional search is given to the user if the process-based search has failed. This method starts from words in a sentence to induce the grammar structure of the sentence. An open-source package (OpenNLP)

has been adopted to implement the query analyzer. The OpenNLP parser abides by the Penn Treebank Style. The three styles of SBAR (WHADVP…), SBARQ (WHADVP …), and VP are filtered by the query analyzer.

Function: query_analyzer
Input: Query q
Output: <predicate, object> **if** q is goal-oriented, NULL
otherwise
Begin
vp = NULL
if q is SBAR (WHADVP…) or SBARQ (WHADVP …) **then**
vp = the verb phrase in WHADVP
else if q is a VP (verb phrase) **then**
vp = the VP
if (vp is not NULL) **then**
<predicate, object> = <the head of vp, the noun in the noun phrase of vp>
return <predicate, object>
else then
return NULL
End

Figure 8. Query Analyzer.

Function: process_searcher
Input: Query q and annotation <predicateq, objectq>
Output: A process relevant to q.
Begin
1. Select all processes from the process repository where the following condition is true (<predicatep, objectp> represents the annotation of process p): (predicatep == %predicateq%) AND (objectp == %objectq%).

Figure 9. Searching Processes.

Examples of these three styles are shown as follows:

SBAR (WHADVP…): How to drive a car in Mumbai.

SBARQ (WHADVP…): How do I drive a car in Mumbai.

VP: Driving a car in Mumbai.

D. Query Transformer

The annotation of a query q (denoted by <predicateq, objectq>) is matched with the annotation of a process p (denoted by <predicatep, objectp>) to decide if p is relevant to q. If a process is found that is in the database then the steps are retrieved and the sub

queries are fired to the traditional search engine. If not then the original query is fired to the traditional search engine.

3.3 Design Approach

3.3.1 Personalization Based Search Engine

The personalization design creates user profile for all participating users. The user need to login to utilize the feature of personalized search. Creation of user profile can be based on

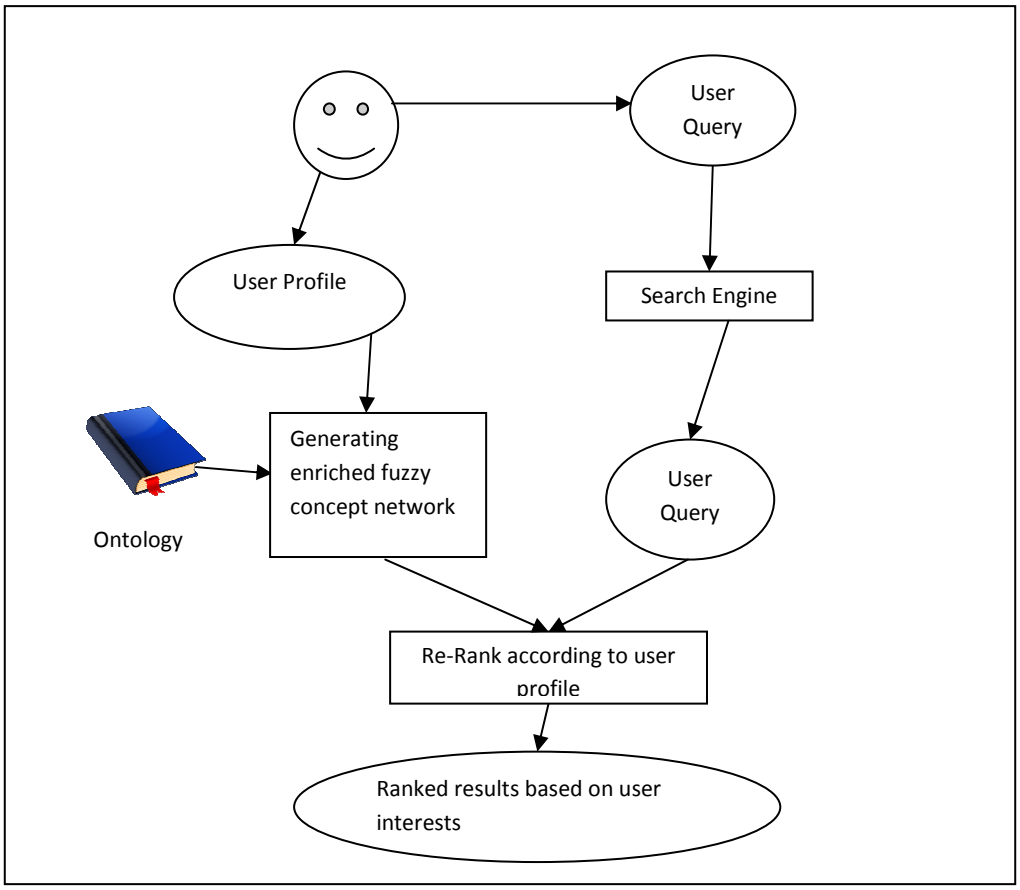

Figure 10. Personalization Implementation Design.

feedback. It can also be done using assessing the cookies and caches of users. When the user submits its query, the query is given to the search engine and results are retrieved normally. In parallel to this, the user profile created, previously, is accessed and fuzzy

concept network (FCN) is created. The FCN is enriched using the ontology to generate Enriched Fuzzy Concept Networks (EFCN). The retrieved pages, along with, the EFCN is given to the ranker. The final result is thus according to user interests [1].

3.3.2 Process Based Search Engine

Based on the proposed model, we have designed the Process-Based Search Engine (PBSE). Assuming that the answer to a query is to combine multiple Web pages, in query pre-processing phase, the initial query that the user provides is rewritten into multiple sub-queries, each of which can then be searched for separately. In order to achieve this, the Process-Based Search Engine is composed of two sub-systems, as shown in Figure 11 where the dashed line represents the proposed system.

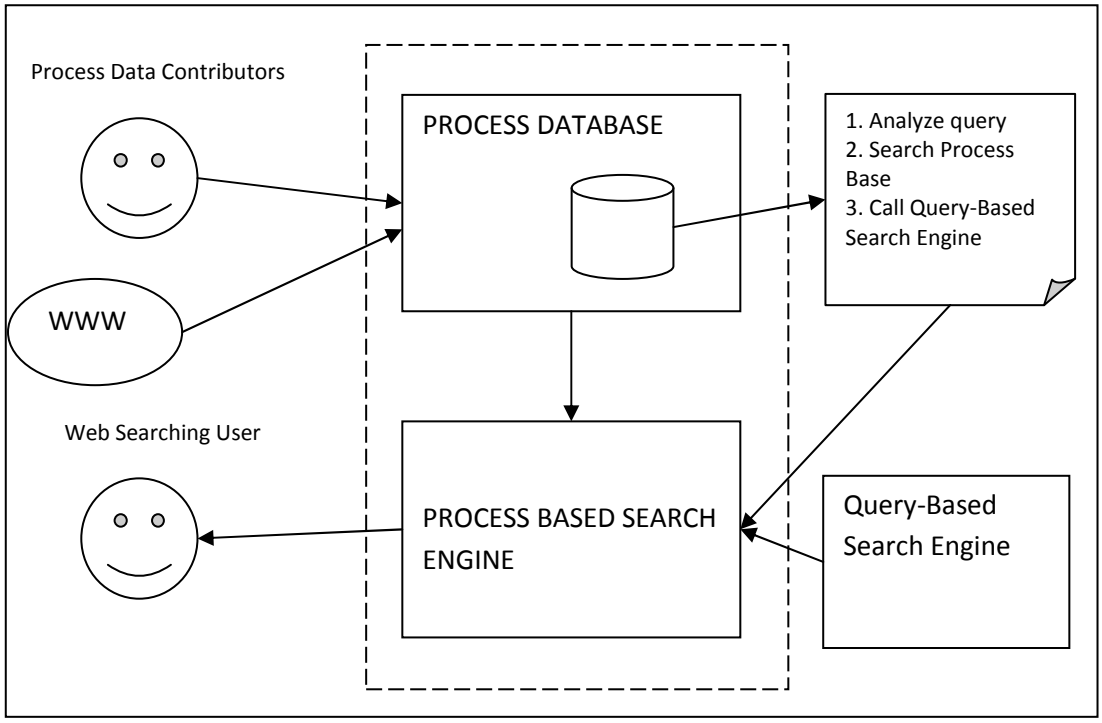

Figure 11. High Level Design of Process Based Search.

The sub-systems of PBSE are:

(1) *The Process Base*: Collects and manages data of processes, which are used by the Process-Based Search Engine to expand the initial query into multiple sub-queries.

(2) *The Process-Based Search Engine*: Transforms the initial query into multiple sub-queries using the Process Base. It uses these sub-queries to search a set of ordered documents as the answer to the initial query.

The Process Base is populated by two sources: Web users' contributions and the data automatically crawled and extracted from the Web. We store the obtained process data in a relational database. Processes are classified for easily searching and browsing. The Process Base provides information to the Process-Based Search Engine sub-system. Figure 11 shows the block diagram of PBSE relying on Web users' contributions. The Process-Based Search Engine searches the Web based on the information of processes, which is stored in the process base. Given a query, assuming that it is not a title search and it is searching for how to achieve a certain goal, the system first searches the Process Base to locate matched processes, and ranks them. Then, the initial query is transformed into multiple sub-queries by combining part of the initial query and each step in the matched process.

3.3.3 User Interface Design

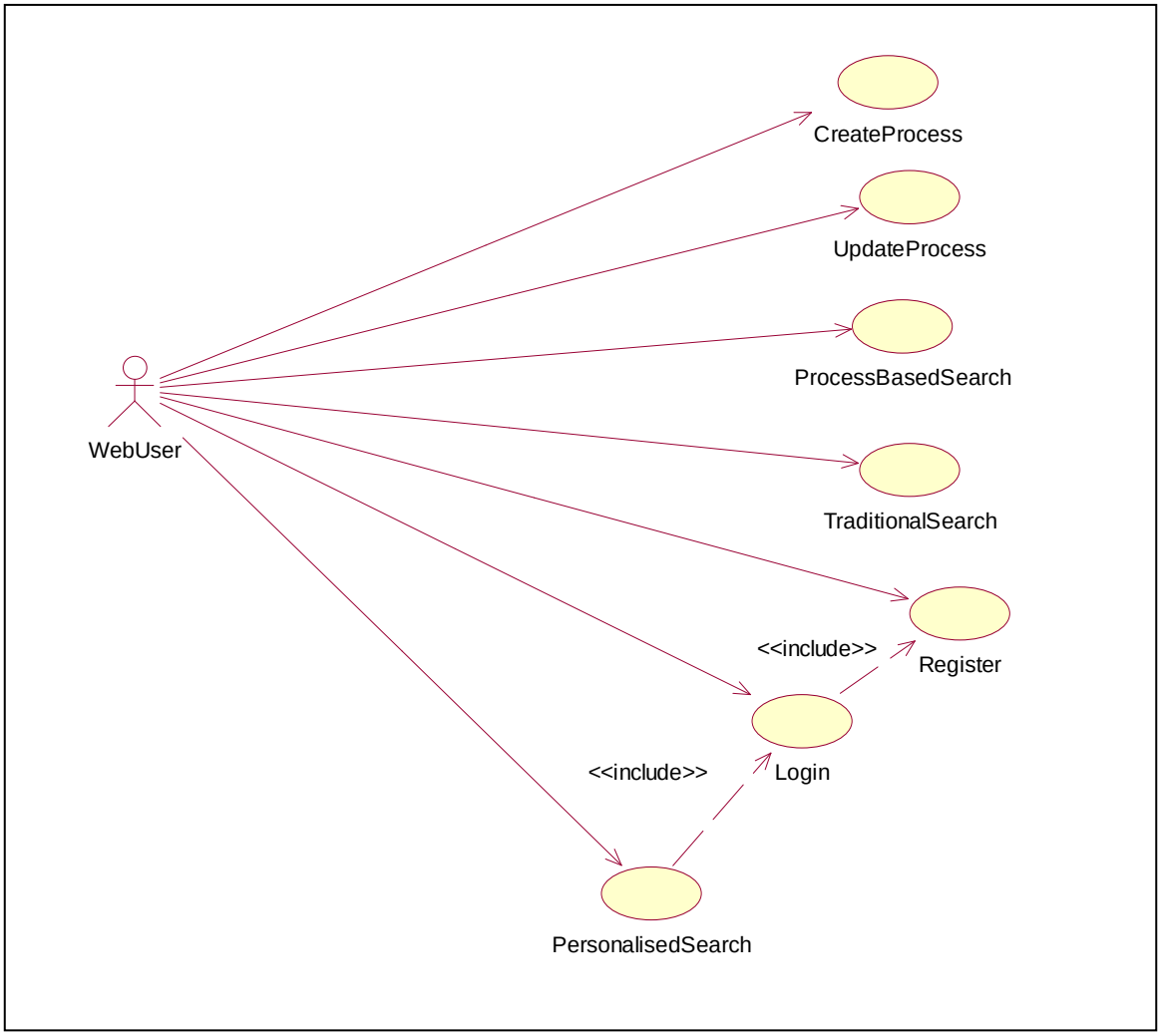

Figure 12. Web User Interaction Use Case Diagram.

The design is basically a Model-View-Controller framework. The view is the front end GUI of the search engine. In this the user can query the search engine traditionally, or query for a process or login into the system and use personalized search results. The Web user interface of process contribution is made as simple as possible.

The basic use case diagram in Figure 12 shows the various interactions and interdependencies between the search engine user and the system.

We have developed a Web user interface based on the concept of Wiki to provide users with full access to the Process Base. There are many Wiki applications implemented in various languages. But these existing Wiki applications are designed to mainly manage plain text. Due to the structural data in the Process Base, it is difficult to adopt an existing Wiki implementation. It may be difficult to decide how Wiki the Web UI of the Process Base is, but the Process Base UI fully follows the philosophy of open-editing and easy use by non-technical users introduced in the Wiki concept. This component is a Servlet Web application. Each action (e.g., create a process) invokes the corresponding method in the servlet.

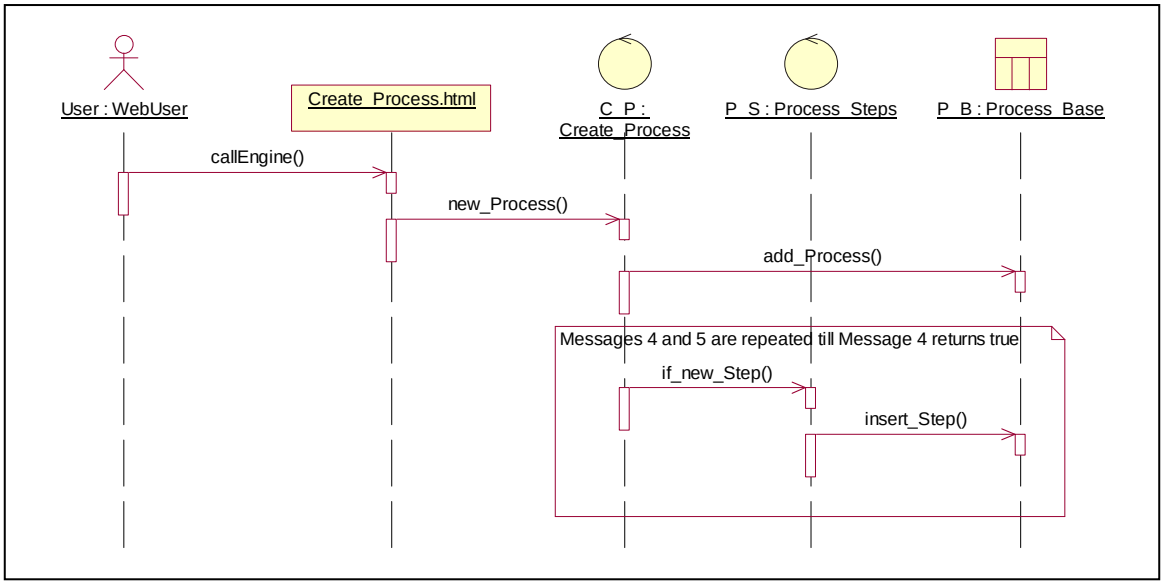

Figure 13 Sequence Diagram for Process Creation.

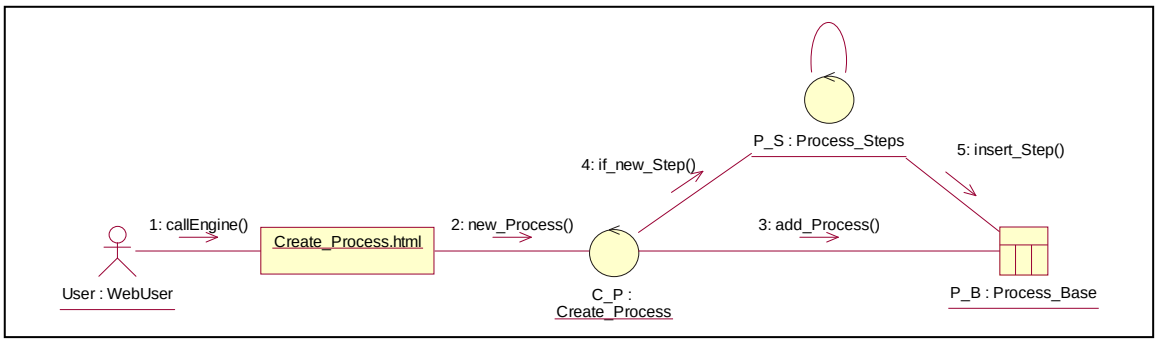

Figure 14. Collaboration Diagram for Process Creation.

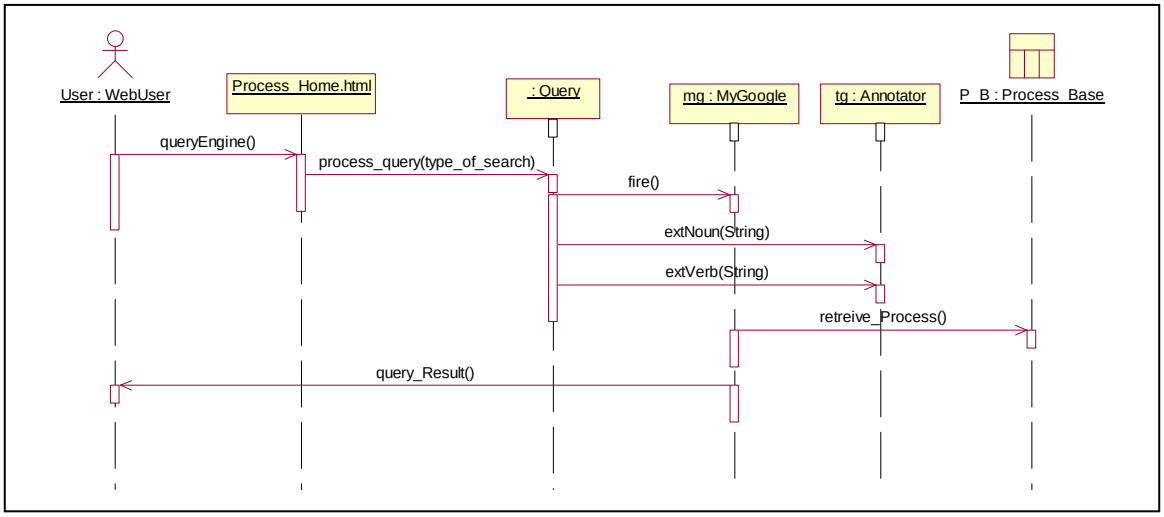

Figure 15. Sequence Diagram for Process Based Search.

The sequence for the interactions for creating a process indicates the stepwise interaction and procedure for the creation of a process to be added into the process repository as shown in Figure 13. The user first needs to call the home page, and then he needs to navigate to add the process by entering the process and its steps into the user interface provided. On completion of the details the details are added to the database. Subsequently steps can be added to the database. The collaboration diagram for process creation indicates the manner in which, the various classes and actors interact which each other for this particular functionality. Figure 14 shows the messages sequenced in the order in which they are sent.

The PBSE is also implemented as a Servlet Web UI to get the query that the user has entered and present the user with process-based search results. Figure 15 illustrates the sequence diagram of how process-based search is realized.

The sequence of messages and method calls required to search for a process in the database is outlined in the sequence diagram. Figure 15 illustrates the sequence of method calls necessary to execute a process based search.

Similarly the collaboration diagram in Figure 16 for the process based search outlines the exact order of messages and method calls required to successfully execute a process based query.

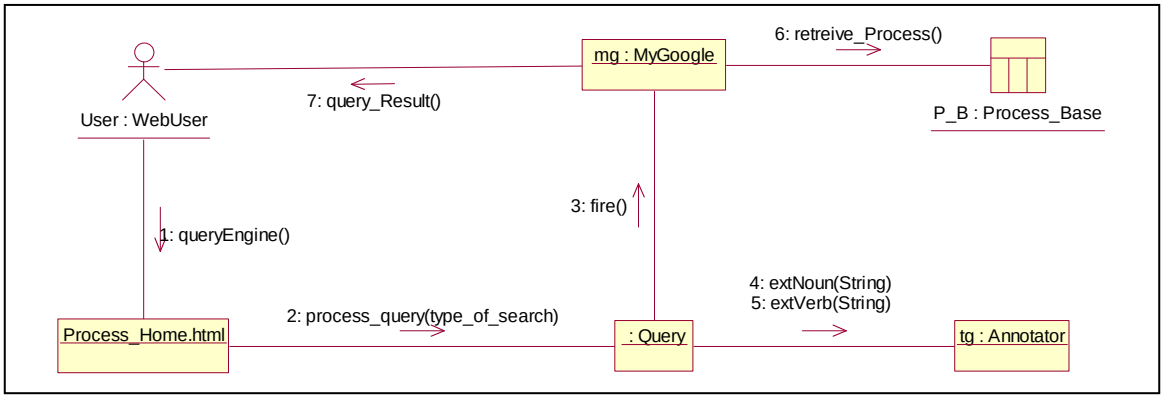

Figure 16. Collaboration Diagram for Process Based Search.

The personalization module of Xplore has the functionality of logging in, to provide users with the option of personalized search according to the user profile captured during the

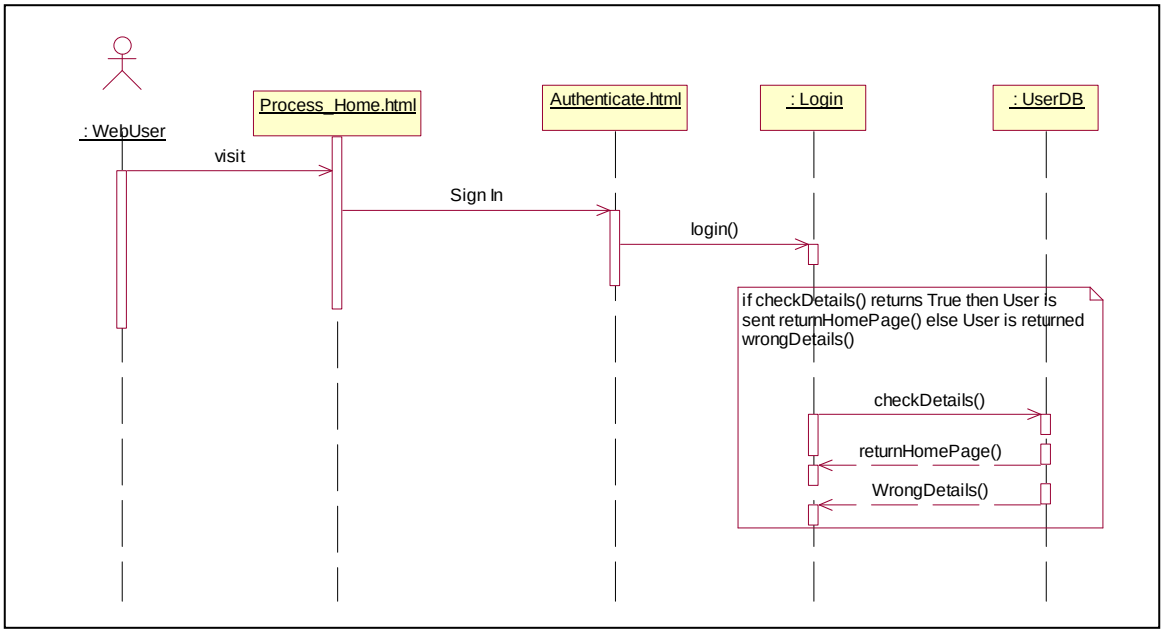

Figure 17. Sequence Diagram for Logging into the Personalized Search Engine.

phase of profile collection. The sequence of messages passed is as shown in Figure 17. The user first accesses the processhome.html page from where he is redirected to the authenticate.html page for the login process. Thereafter the details are checked and verified with those present in the database and if the details such as username and password are found authentic the user is logged. The collaboration diagram in Figure 18 specifies the ordering of events as they happen during the event of log in.

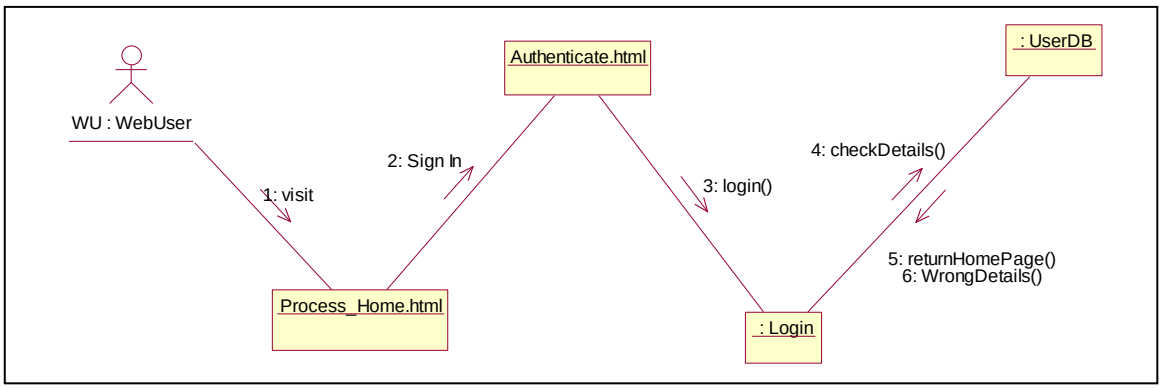

Figure 18. Collaboration Diagram for Logging into the Personalized Search Engine.

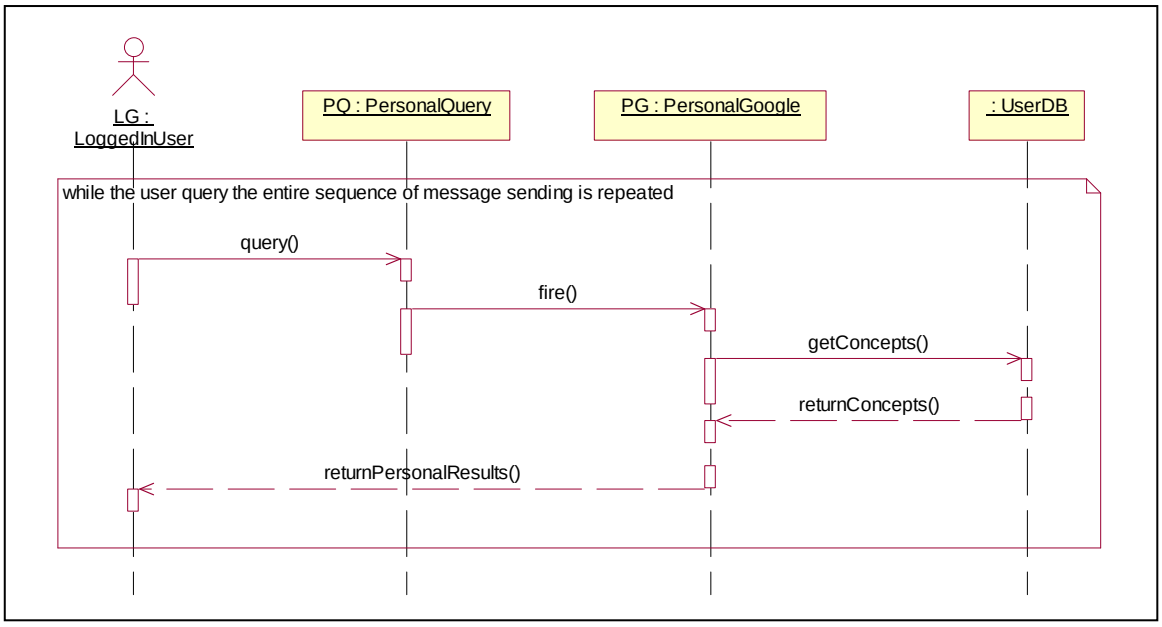

Figure 19. Sequence Diagram for Personalized Search.

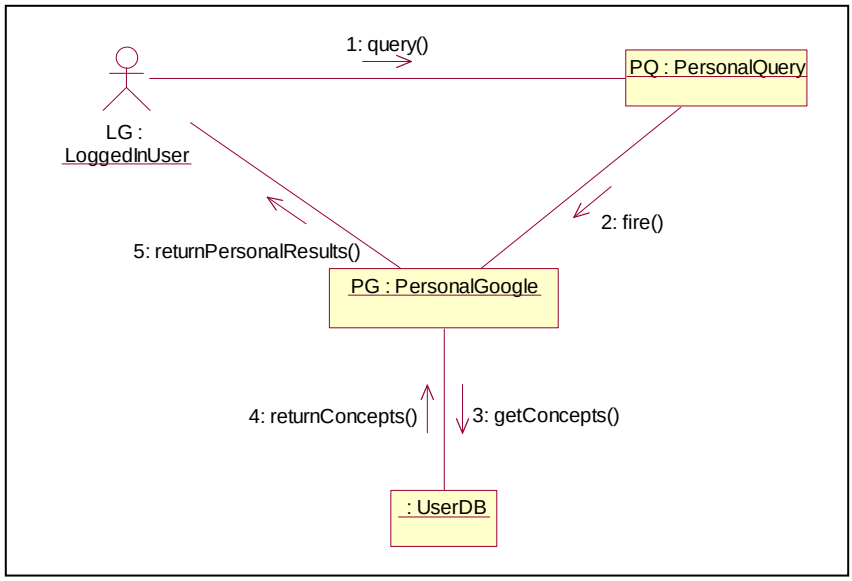

Figure 20. Collaboration Diagram for Personalized Search.

The procedure for personalized search requires that the user first be logged in. Once logged in, then his query is fired to the search engine. According to the user profile that was generated based on the user pattern captured during the data collection phase, the results are re-ranked according to the preferences generated by retrieving the concepts from the database. These concepts were enhanced using the WordNet ontology. The entire sequence of messages is listed in Figure 19. The collaboration diagram shown in Figure 20 draws out the ordering of events that happen when a logged in user searches for content in the database. The entire chain of events is repeated till the user finishes querying the search engine.

4. IMPLEMENTATION

4.1 Software requirements

- Java Version: jdk 1.5 onwards is required
- OS Version: Windows XP 32bit
- Eclipse Galileo
- OpenNLP API
- JSON API
- Google SOAP API
- Mozilla Firefox
- KomodoEdit 6.1.1

4.2 Hardware requirements

- RAM: 512 DDR is the minimum system requirement for uninterrupted working of our system.
- Hard disk: Our system requires minimum of 40 GB HDD to exhibit proper functionality.
- CPU: Intel x86 P4 2.8Ghz.
- LAN card: LAN card is required to connect with the internet.

4.3 Database

Oracle 10g database has been used to store the process base. The user profiles and their browsing patterns are also stored in the database.

4.3.1 Process Base

Definition of process: A sequence of steps or activities for achieving a specific goal.

ID	Title	Category	Description

Table 1. Process Layout.

P_ID	Object	Predicate

Table 2. Annotation Format.

ID	Title	URL	Des	Process_ID

Table 3. Steps Layout.

userName	userID	name	Concepts

Table 4. User profile.

Definition of process base: A database of processes and a group of programs which provide means to collect and access the process data.

Definition of process-based search: An extension to traditional Web search. Given a query q provided to the Web search engine, discover the implied sub- concepts, c_1, c_2, …, c_n, of this initial query q if they exist, using the process base. Perform traditional Web search on each sub-concept c_i, i = 1, 2, …, n. Let r_i denote the obtained URL list by searching concept c_i, then $< r_1, r_2, \ldots, r_n >$ represents the result of the process-based search on the initial query q.

4.3.2 Database for Personalization

Each user is given a unique user id. After the user registers for the first time, concepts are generated in the form of a blob and added to the database. Subsequent interactions of the user with the personalization module of the database scan through this table to generate personalized results.

The results for the user are re-ranked according to the concepts that are present in his profile. These concepts are generated through the user browsing patterns and enriched with the help of the WordNet ontology.

4.4 Implementation Scheme

4.4.1 Process Based Search Engine

The procedure of Process-Based search is shown in Figure. 21. The Process-Based Search Engine has the following main functions.

1. **Analyzing query:** Deciding if the query should be rewritten and searched based on process.

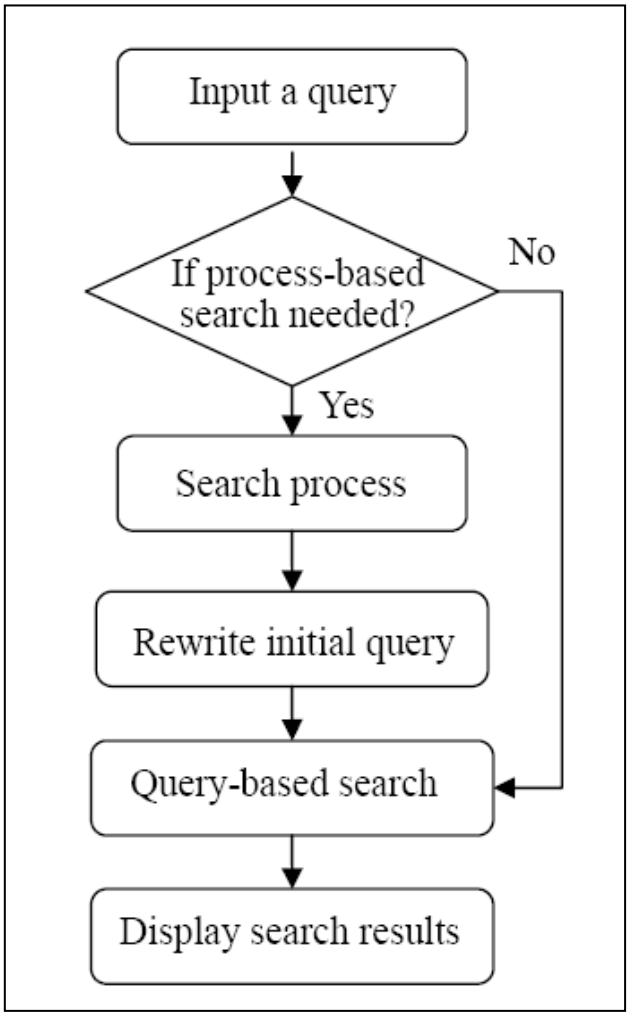

Figure 21. Flow Chart of Process-Based Search.

2. **Searching the Process Base:** Given the query, searching the Process Base and returning a list of matched processes.

3. **Searching the Web:** Rewriting the query into multiple sub-queries, using the information of matched process, and then sending the rewritten queries to a query-based search engine to search the Web for every step in the process.

4. **Displaying search result**: Combining answer sets from all the steps and displaying in the step-by-step format of process.

4.4.2 Personalization Based Search

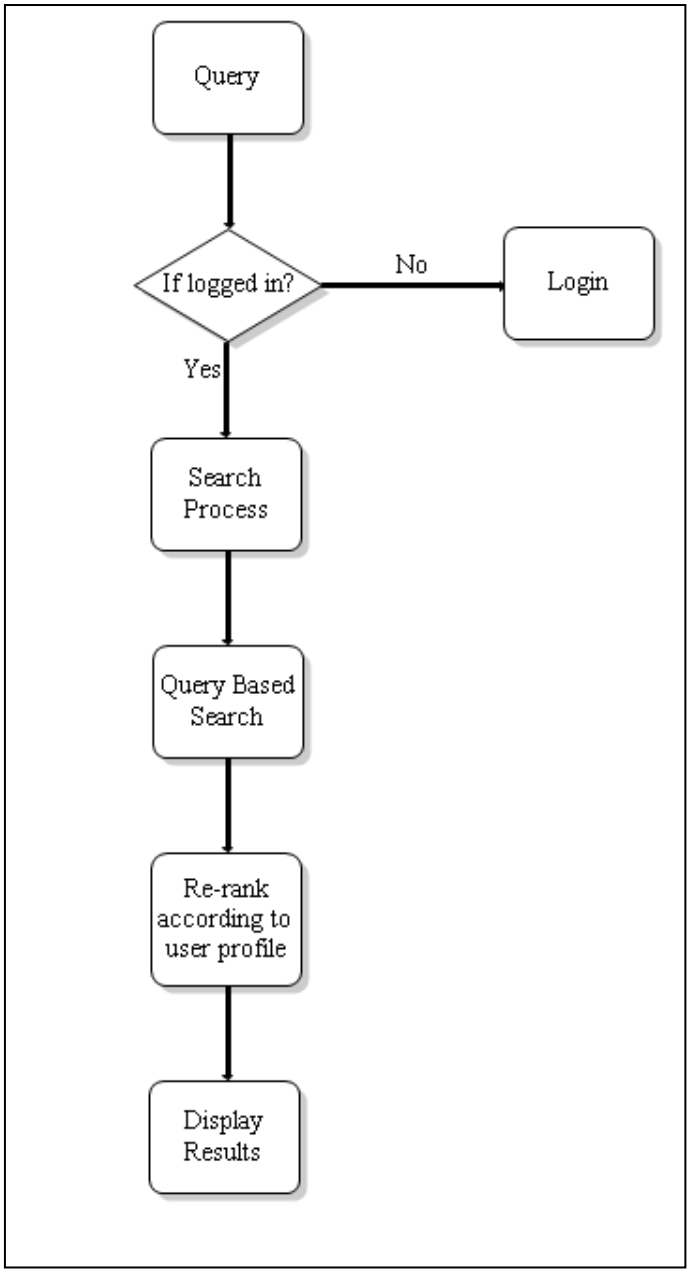

Figure 22. Flow Chart of Personalization.

As shown in Fig. 22 the procedure of Personalization based results has the following main functions:

1. **Logging in**: The user has to log in to be able to see the results of the query according to the preferences captured. If the user is not logged in he gets the results from the traditional search engine.

2. **Query:** If the user is logged in then he can fire his query into the search engine.

3. **Query Based Search:** Traditional query based search is carried out.

4. **Re-rank according to user profile**: The results from the traditional engine are re-ranked according to the user's profile in the database.

5. **Display Results:** Here the results after re-ranking are displayed to the logged in user. The results have been modified to suit the subjective needs of the user according to the browsing pattern.

5. RESULTS

5.1 Snapshots

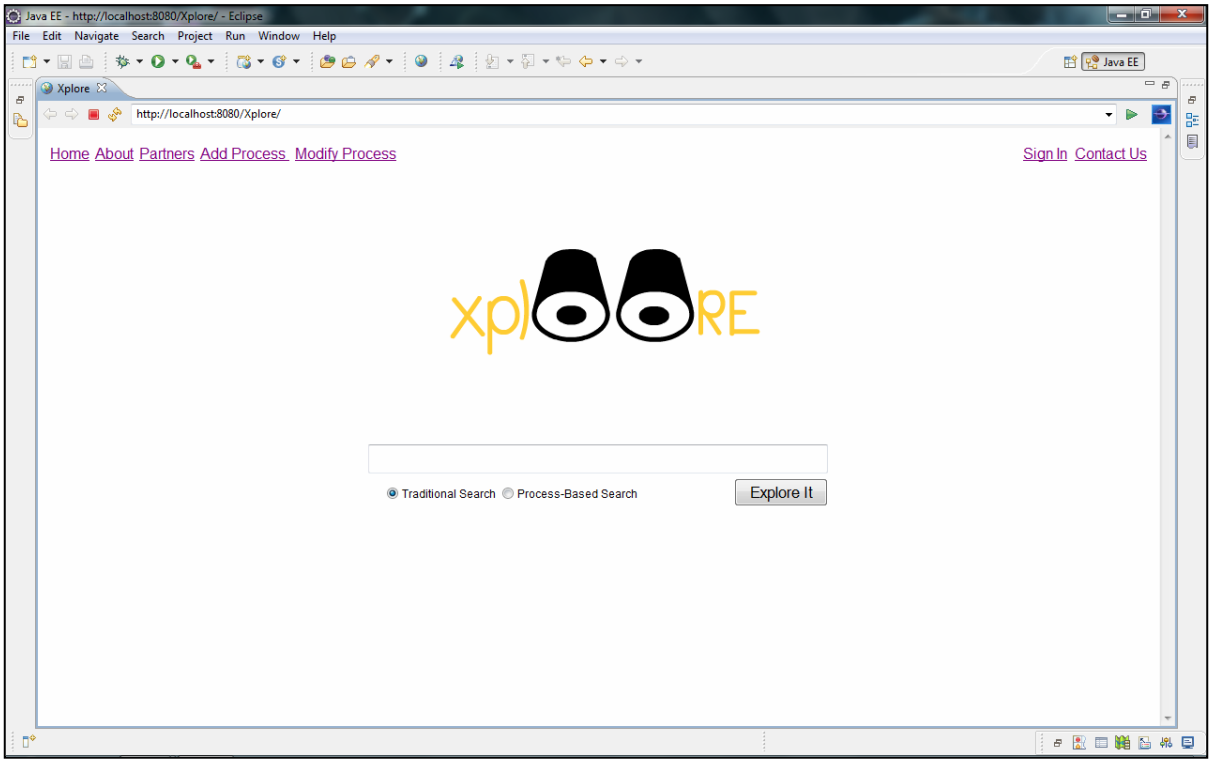

Figure 23. Home Page- Xplore.

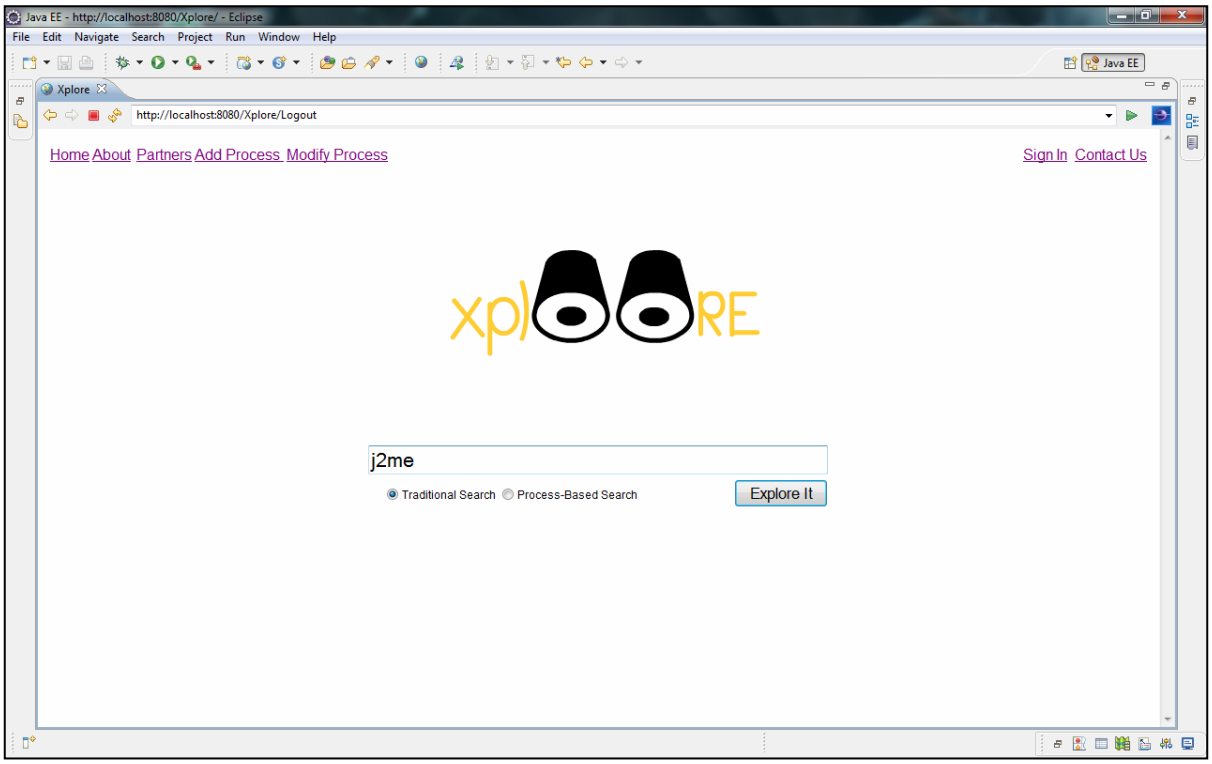

Figure 24. Traditional Search for the query 'j2me'.

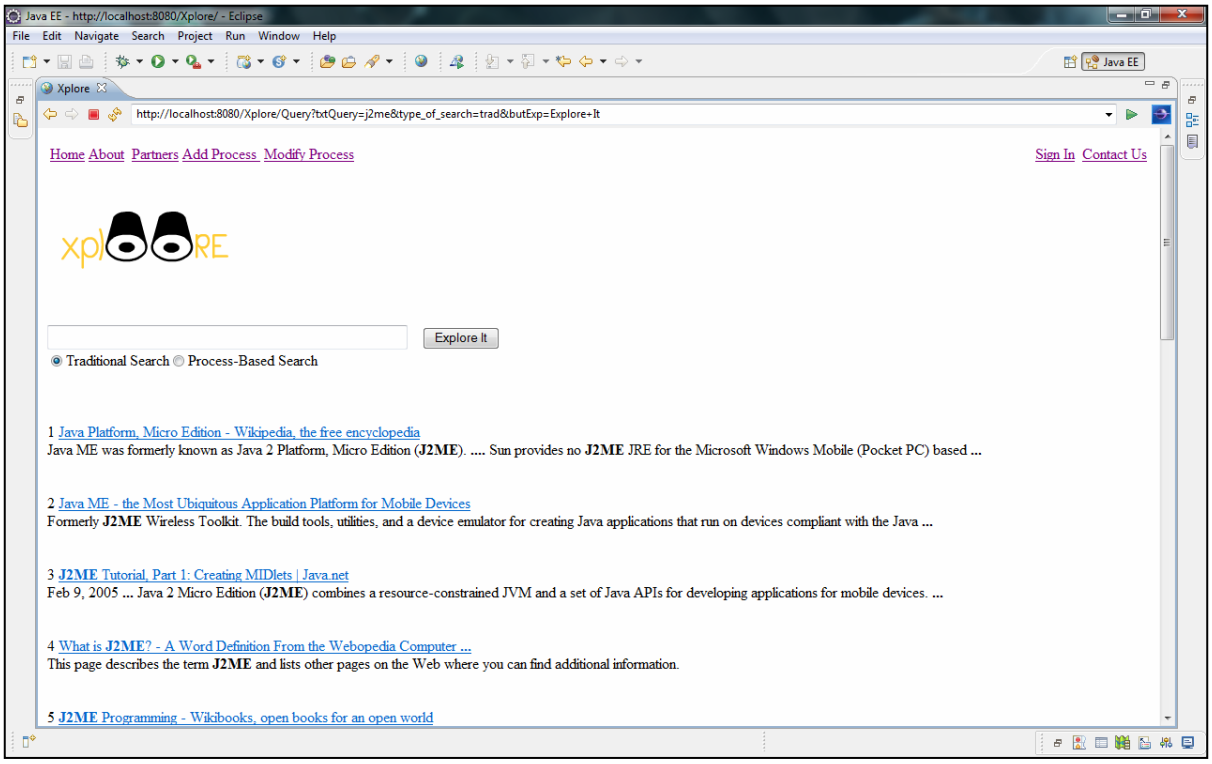

Figure 25. Traditional search results for the query 'j2me'.

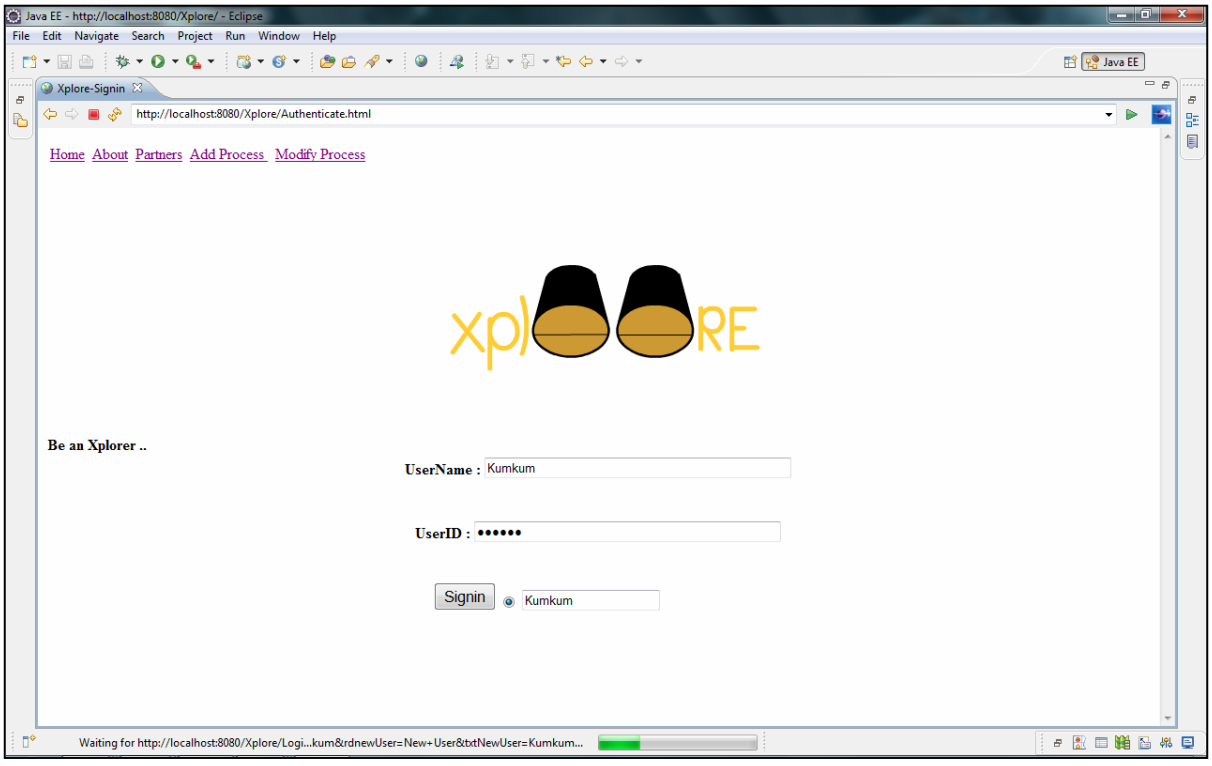

Figure 26. Login for the user.

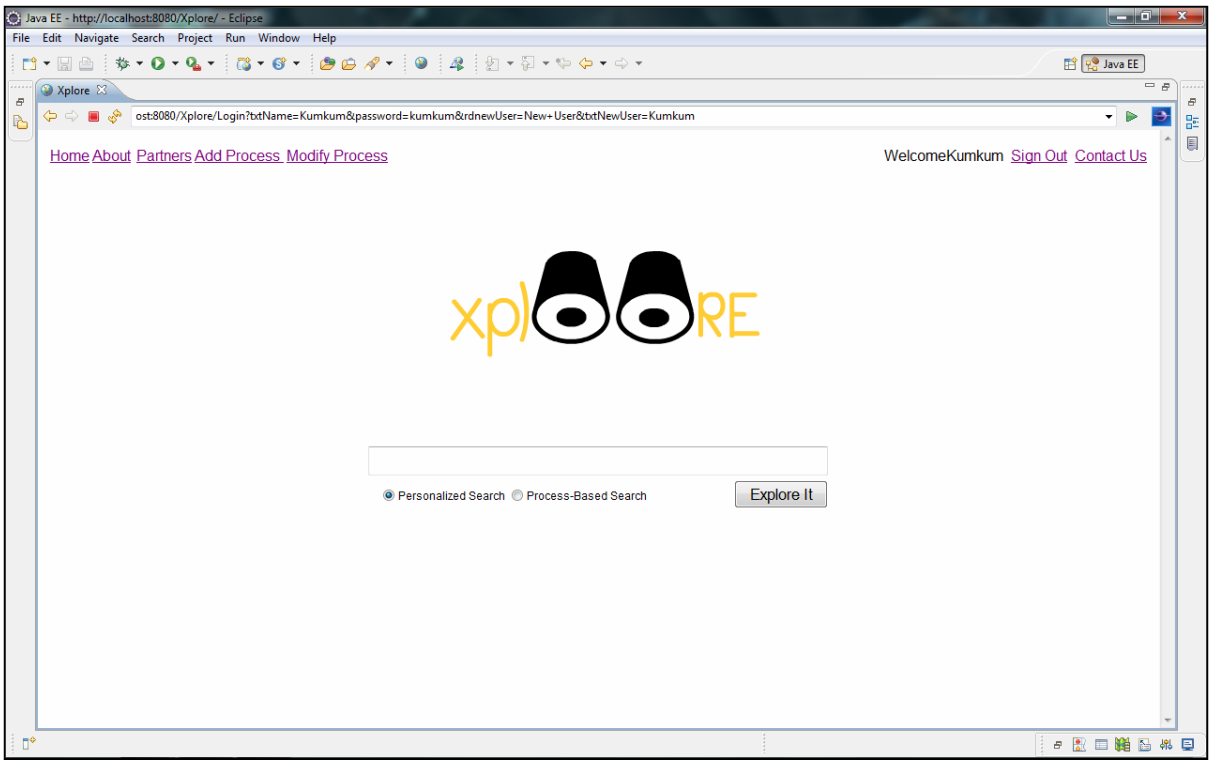

Figure 27. User Logged In.

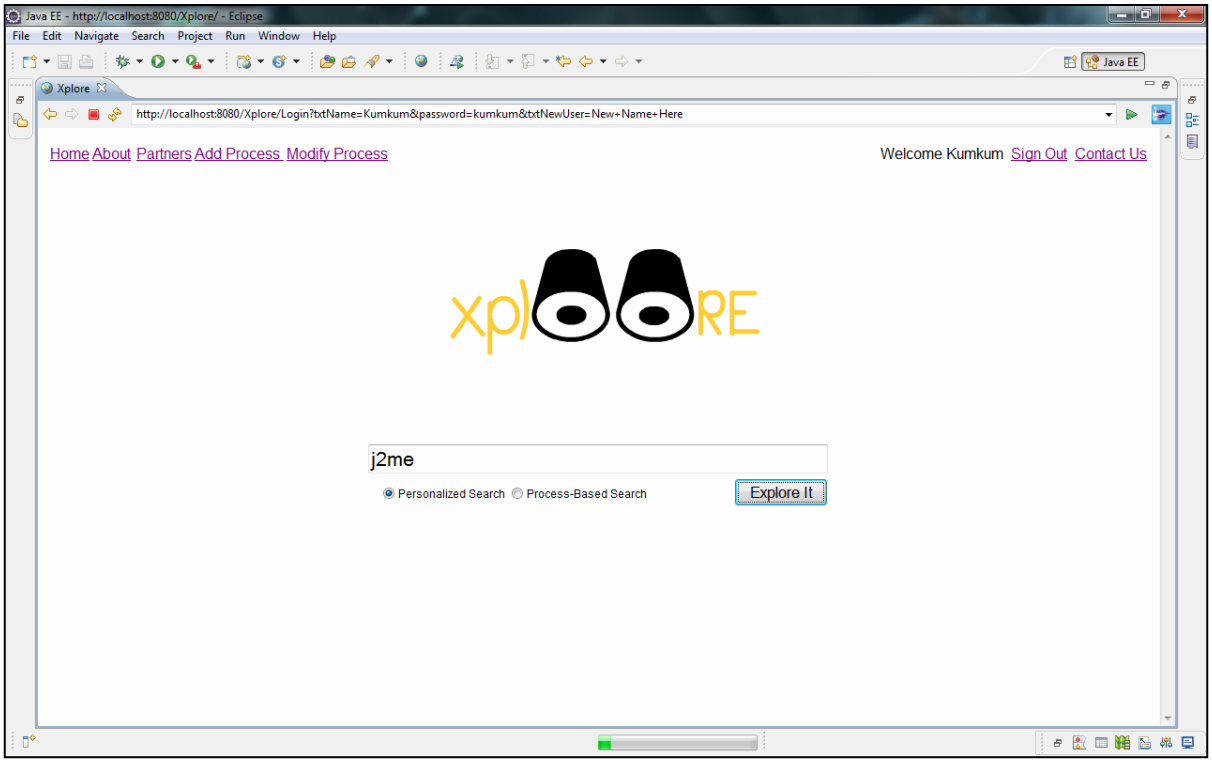

Figure 28. Personalized Search.

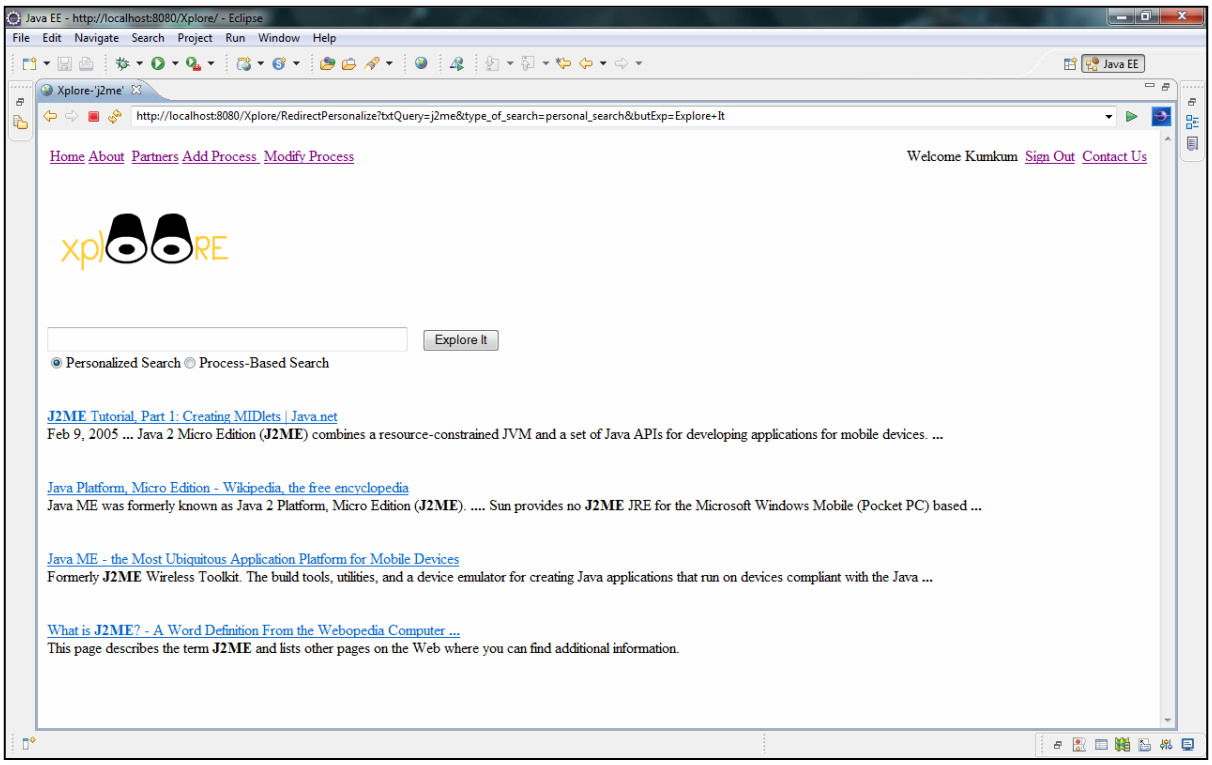

Figure 29. Personalized Results for the query 'j2me'.

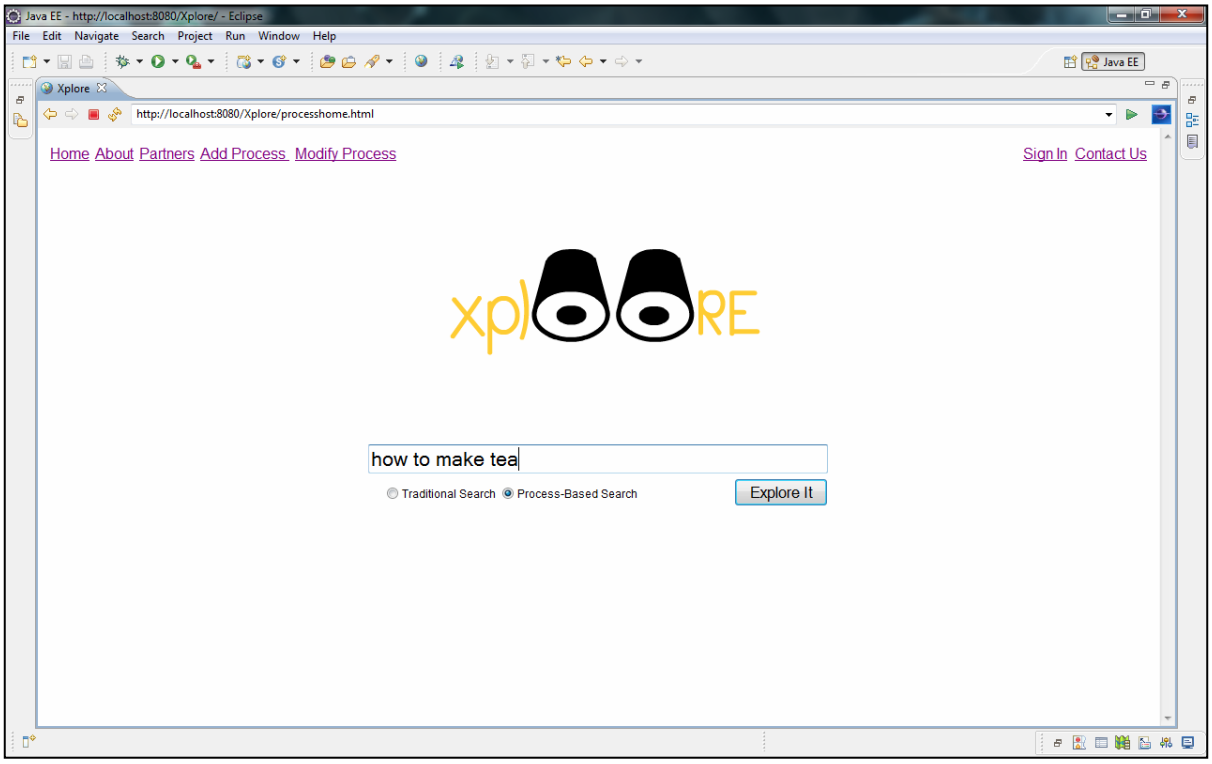

Figure 30. Process Based Search for the query 'how to make tea'.

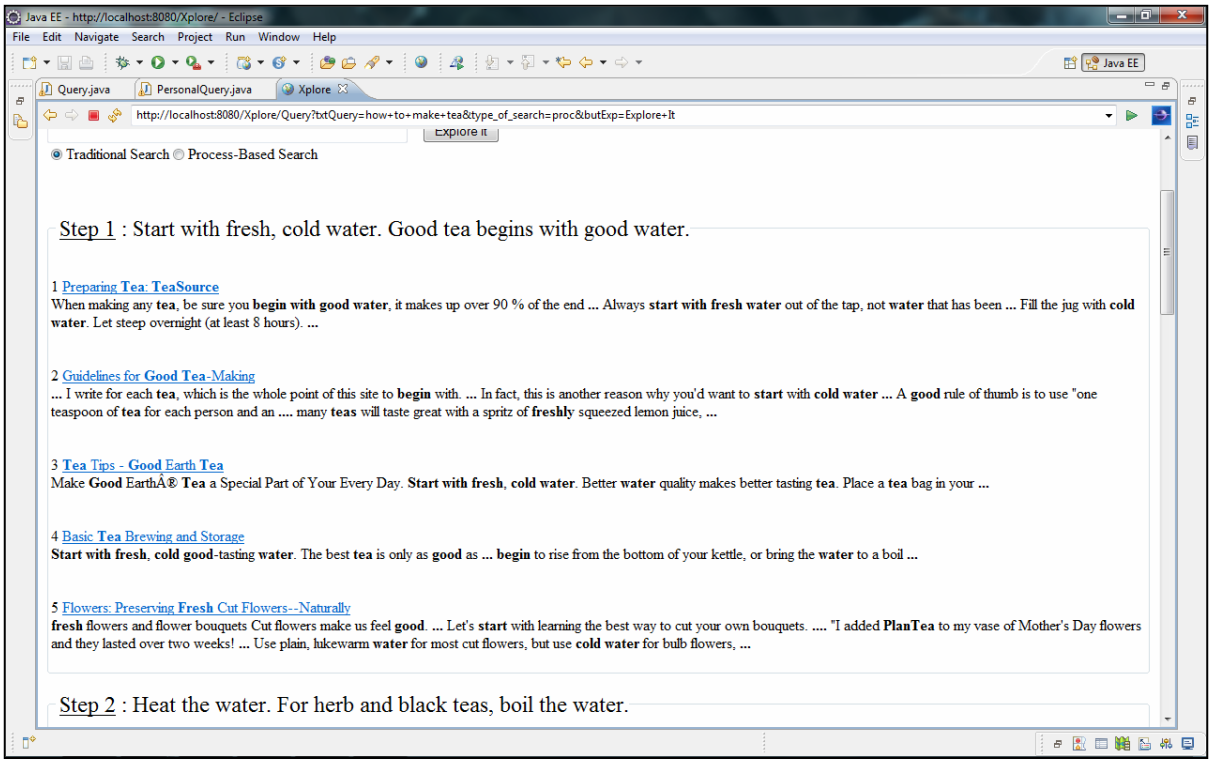

Figure 31. Process Based Results for the query 'how to make tea'.

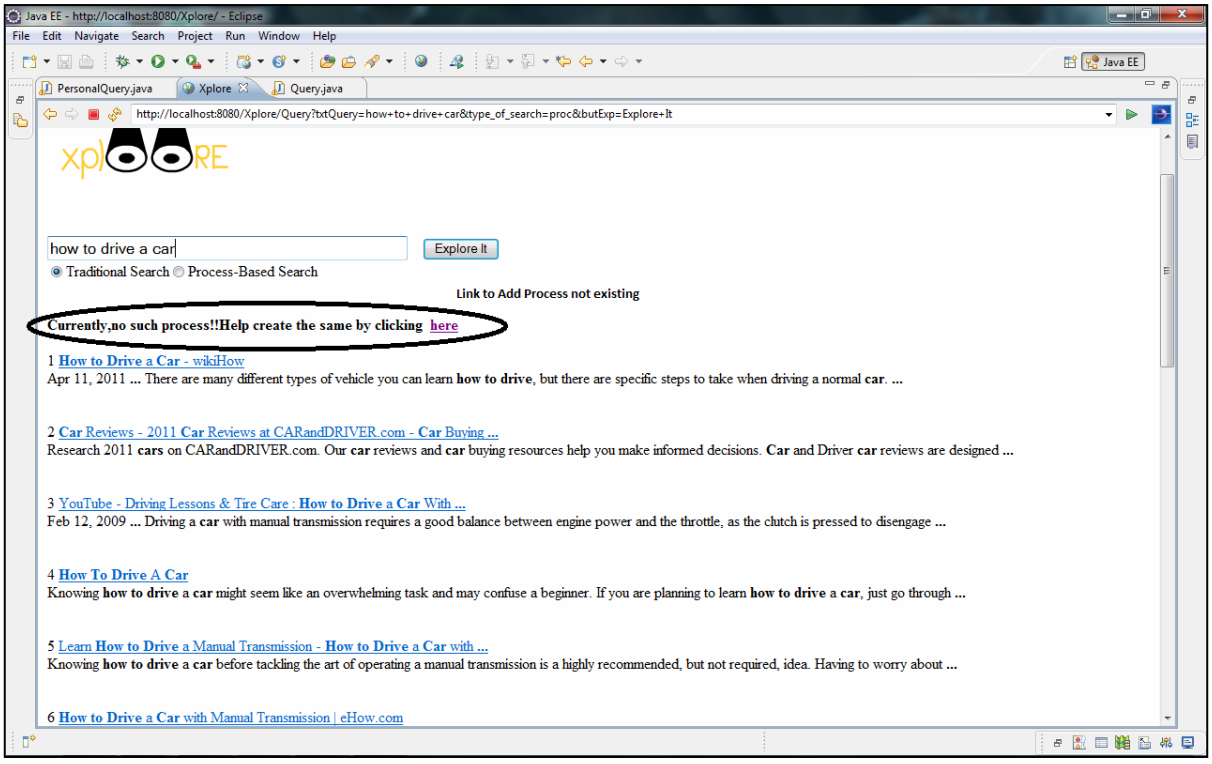

Figure 32. Process absence from the database.

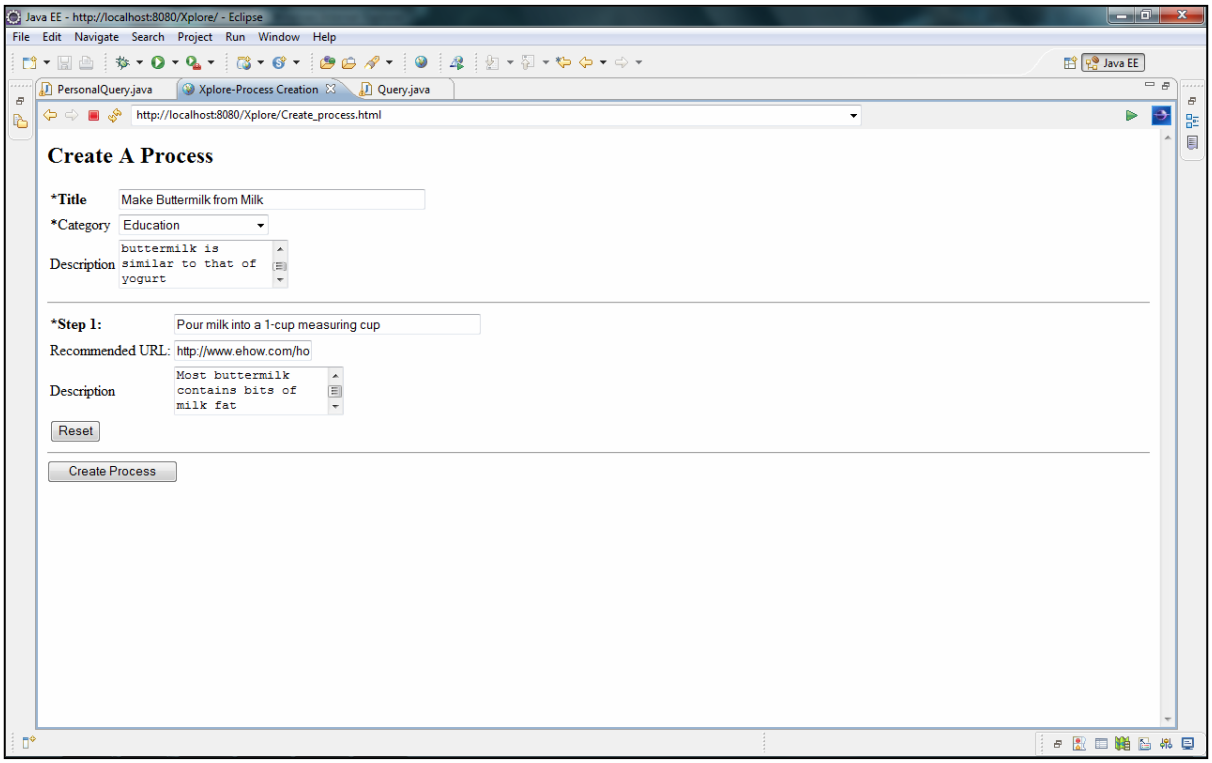

Figure 33. Creating and adding a process to the database.

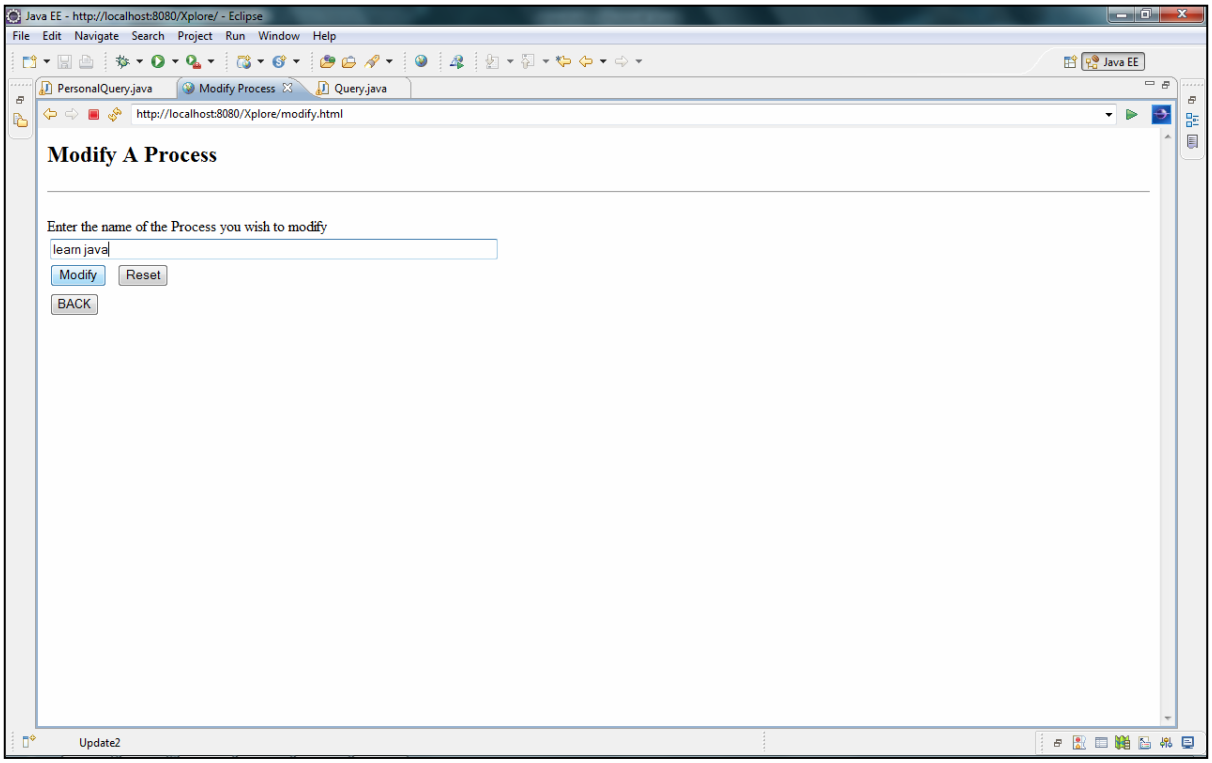

Figure 34. Modifying a process in the process base.

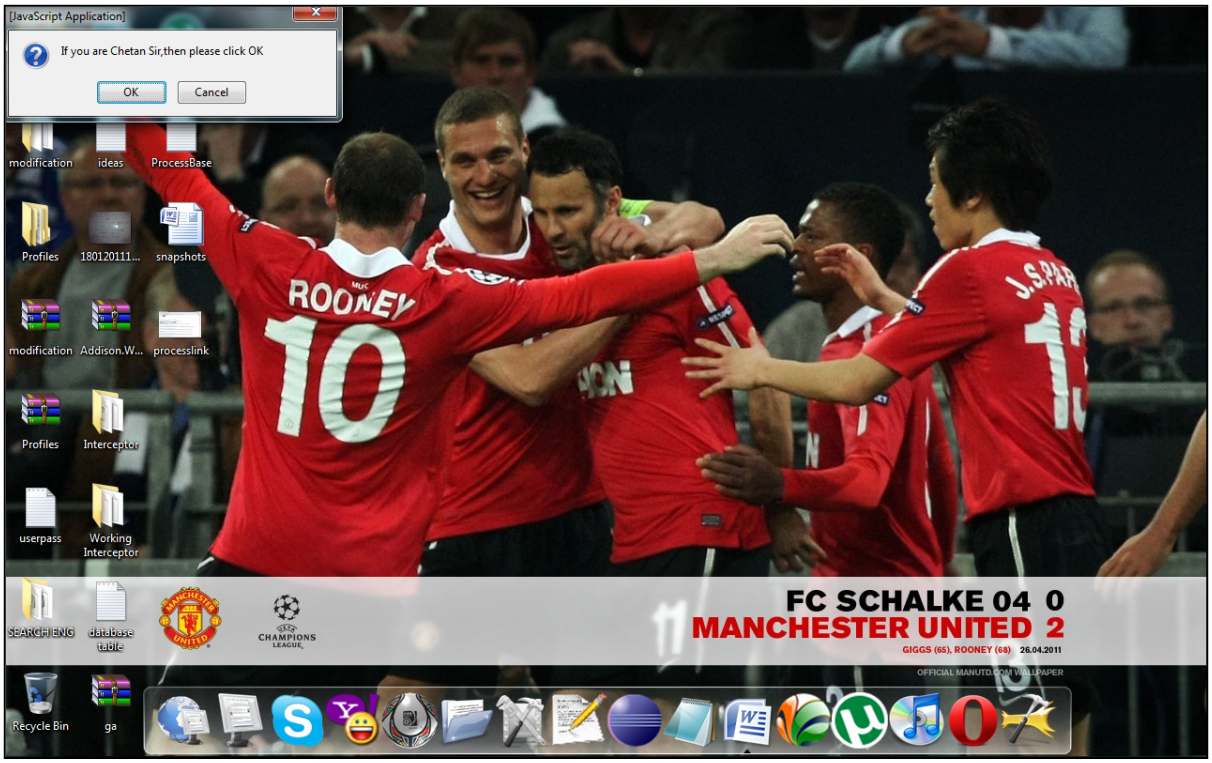

Figure 35. Validation for the add-on 'Interceptor'.

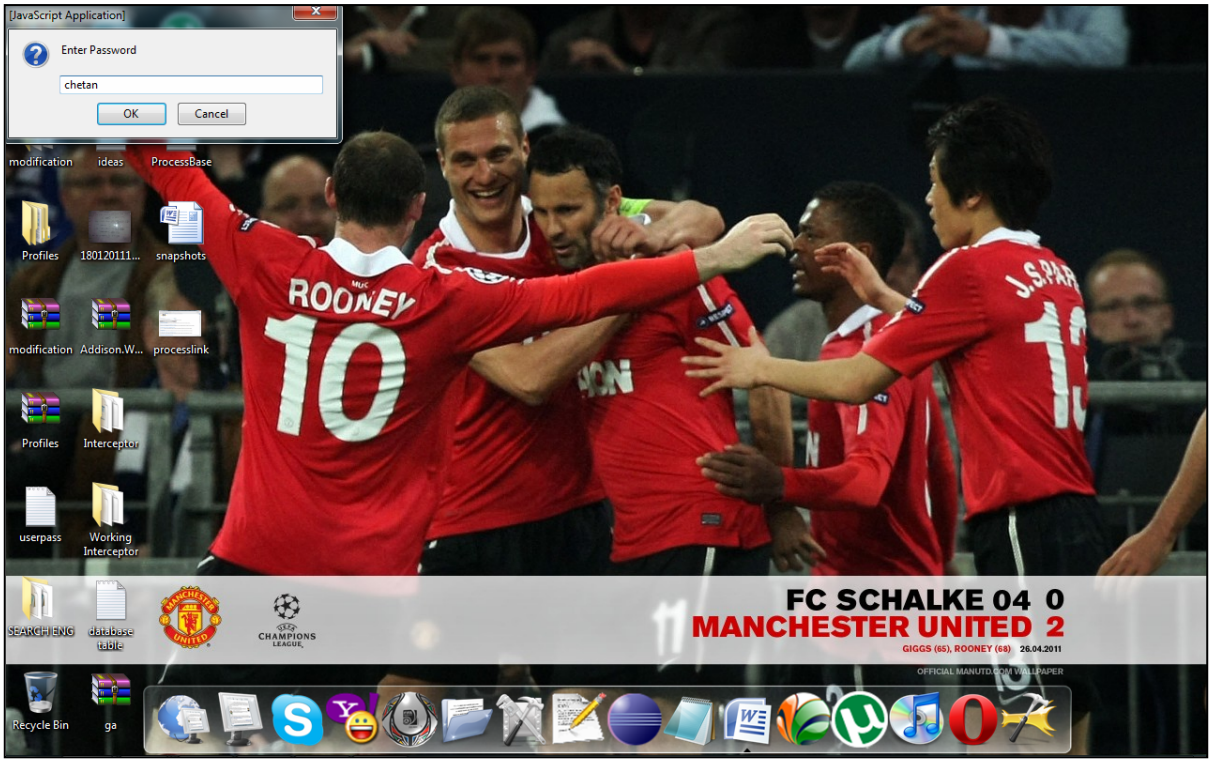

Figure 36. Password Verification for the add-on 'Interceptor'.

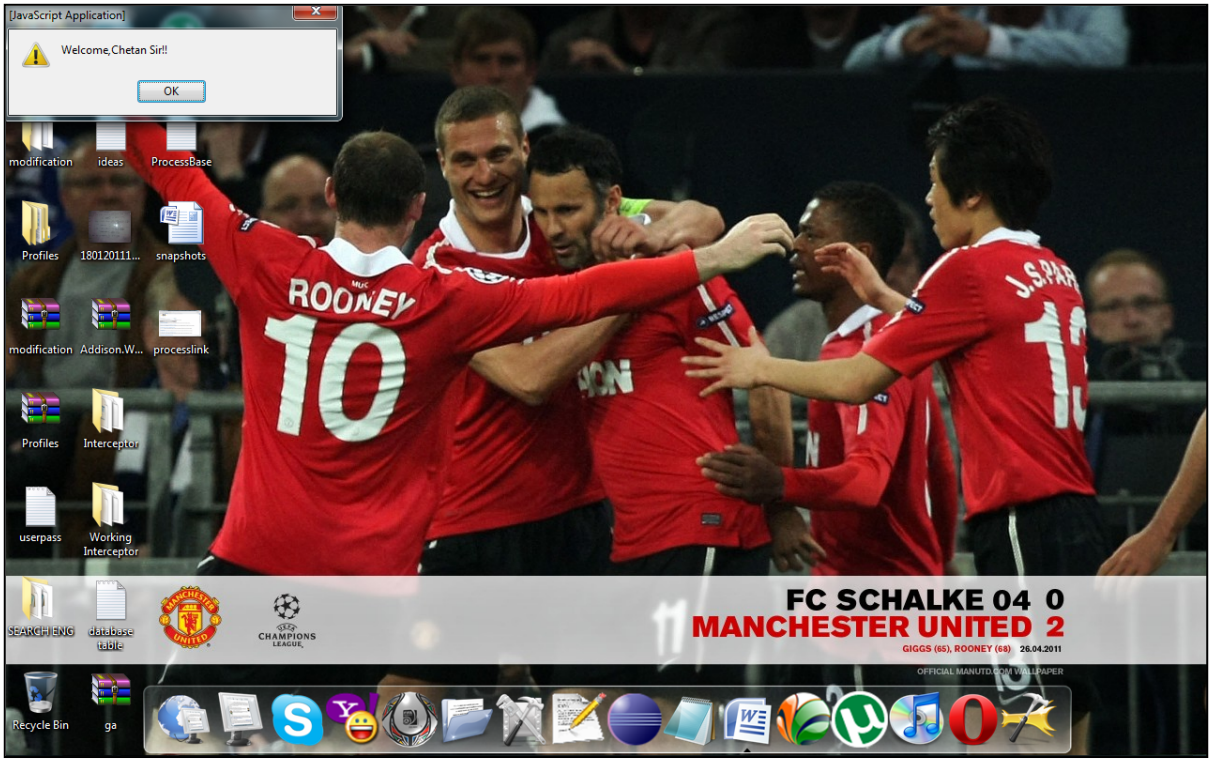

Figure 37. Password Verification completed.

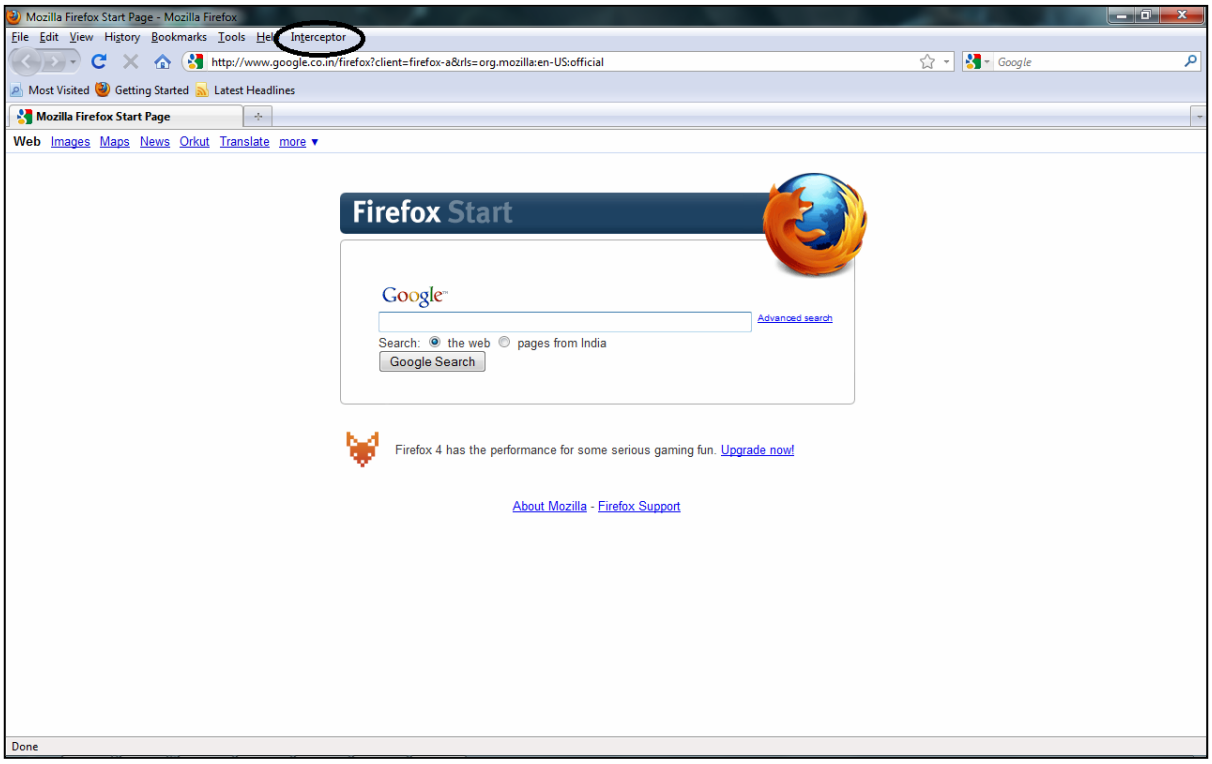

Figure 38. 'Interceptor' add-on at work with Mozilla Firefox.

5.2 Experimental Results for Personalization

In order to test and evaluate the system, the users' profile was collected using the Interceptor add-on and saved in a text file. The user preferences are assumed to be in the reverse order of the entries in the text file. Then the WordNet ontology was used to generate the enriched fuzzy concept networks. We then compare the results from the traditional search engine with the personalized results using EFCN.

An evaluating measure 'd' is used for comparing the ranking performed by the proposed system to that provided by the traditional search [1]. A lower value of 'd' indicates more relevant results for the user.

The average difference between the rankings is defined as follows:

$$d = \frac{1}{m} \sum_{i=1}^{m} | r_i - r_i' | \qquad \textbf{Eq. 5.1}$$

where m is the number of web pages, r_i is the ranking of the user, and r_i' is the ranking generated by the proposed system or by traditional search.

User 1	User 2	User 3	User 4	User 5
4	3	3	1	1
1	4	1	2	4
2	2	2	4	3
3	1	4	3	2

Table 5. Ranking of five users.

The results are presented as follows: Table 5 shows the ranks of the links of the users, captured using 'Interceptor'. Table 6 shows the personalized ranking of the users using the enriched fuzzy concept networks. Shaded box implies that personalized rank is equal to the user rank from Table 5. Table 7 lists, 'd' the evaluating measure for the enriched fuzzy concept networks as well as for the traditional search. Table 7 indicates an

improvement of 33 % against the traditional search results. Also evaluating measure ‘d’ for the two rankings is depicted in Figure 39.

User 1	User 2	User 3	User 4	User 5
2	2	3	1	1
1	1	1	2	4
4	3	2	3	2
3	4	4	4	3

Table 6. Ranking of Enriched FCN.

	User 1	User 2	User 3	User 4	User 5	Average
EFCN	**1**	**2**	**0**	**0.5**	**0.5**	**0.8**
Traditional	**1.5**	**2**	**1**	**0.5**	**1**	**1.2**

Table 7. Evaluating measure ‘d’.

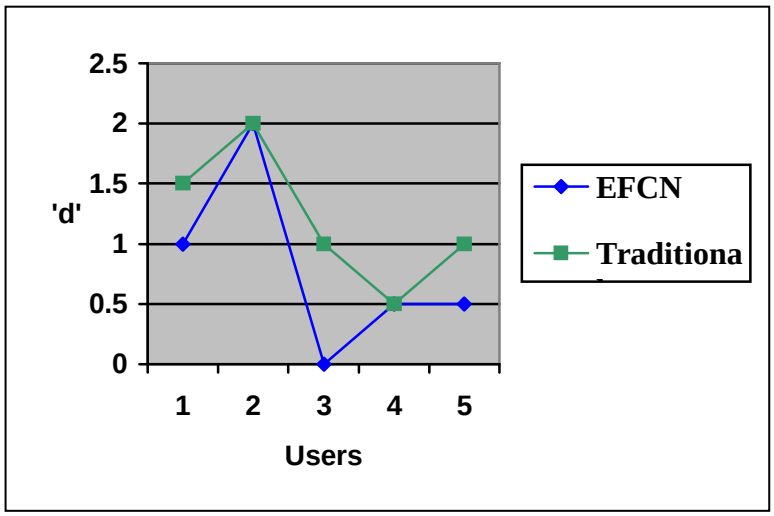

Figure 39. Evaluation measure ‘d’.

5.3 Experimental Results for Process Based Search Engine

Comparing the Process Based Search Engine (PBSE) with the traditional engine for the process based searches shows that queries can fall into one of these three categories[2]:

1. PBSE outperforms the traditional search engine, by providing comprehensive answers to the query without having the users to clarify the same.

2. PBSE does not outperform the traditional search engine because the answer to the query exists in one document, and the traditional search engine is able to extract the information without any extra effort from the user.

3. Neither PBSE nor traditional search is able to search for an acceptably complete answer to the query, due to lack of information on the Web, the ambiguity of the query or a malformed query.

6. CONCLUSION

In our project, we have described the Process-Based Search Engine, a system searching the Web based on the information of processes. The processes are automatically extracted from the Web, annotated, and stored in a relational database. PBSE is designed to extend traditional Web search engines.

Assisted by the information of processes, PBSE is able to understand users' complex queries better to perform more meaningful searches. If the user is searching for a solution or a method to pursue a goal, PBSE provides the user with a group of ordered information by searching a sequence of sub-queries.

A process extractor was developed to automatically gather processes from the Web, more specifically the website eHow. Results showed that this component could effectively learn to extract hundreds of processes.

In the personalization module, an add-on has been implemented that assists in capturing the user browsing patterns. This helps us construct the user profiles which are then stored in the database when the user registers with the system. After logging in thereafter he will get personalized results according to the browsing pattern of the period of data collection.

The concepts have been enriched using the WordNet ontology. These enriched concepts are used to extend the user profile for more accurate and relevant results that adapt better to real world scenarios.

7. FUTURE WORK

We have built the Web Based Search Engine – Xplore. The process based module needs a process extractor that can extract processes from other process centric websites.

Using the Google SOAP API restricts the number of results returned. Also the Google SOAP Search API has been deprecated and it may not be supported, and therefore a more long term solution needs to be developed.

The personalization module re-ranks the first four results due to the limitations of the Google SOAP Search API. It is imperative to increase the number of personalized results. A more customized solution will have to be developed for that.

The generalization and specialization of the processes in the database needs to be developed. This needs to be done to increase the reusability and extensibility of the process base.

The approach of “topical crawling” is simple but effective, which may be applied to general topical crawling after extensive experiments. Currently, only limited experiments on process crawling have been conducted.

Currently, there are only about 200 processes stored in the process base, therefore, the use of a general-purpose relational database (Oracle 10g) is not a problem. However, a specific storage system should be designed to facilitate efficient update and access to a large number of processes.

The current implementation of query analysis is invoking the OpenNLP APIs without any customization. However, as the query analyzer only involves a subset of grammars, a customized adoption of OpenNLP by eliminating uninteresting grammars can improve the efficiency of the query analyzer.

An evaluation model for process-based searches is needed for the purpose of theoretical analysis of the approach and more objective and larger scale experiments can further validate the proposed techniques.

In the personalization module, we have developed an add-on for the Mozilla Firefox browser called Interceptor that captures the user browsing pattern. But it does so offline and the patterns need to be manually collected and fed into the system the first time the user registers and logs into the system. This entire process needs to be taken live and automated, so that user patterns are captured on the fly and profiles are built automatically without having to manually collect the browsing pattern.

Personalization is extremely subjective and extensive tests need to be carried with a focused evaluation mechanism to prove the purpose of personalization using enriched fuzzy concept networks.

REFERENCES

[1] Fardin Akhlaghian, Batool Arzanian and Parham Moradi, ”A Personalized Search Engine Using Ontology-based Fuzzy Concept Networks”, 2010 International Conference on Data Storage and Data Engineering.

[2] Yaling Liu and Arvin Agah, “A Prototype Process-Based Search Engine”, 2009 IEEE International Conference on Semantic Computing.

[3] Alexander Pretschner, ”Ontology Based Personalized Search Engine”, Dipl.-Inform RWTH, Aachen, Germany, 1998.

[4] Qingzhao Tan, “Designing New Crawling and Indexing Techniques”, Ph.D. thesis, Pensylvania State University, U.S.A., December 2008.

[5] David Hawking, “Web Search Engines:Part I”, Computer; Jun 2006, 39(6).

[6] David Hawking, “Web Search Engines:Part II”, Computer, Aug. 2006, 39(8).

[7] Soumen Chakrabarti, “Mining the Web: Discovering Knowledge from Hypertext Data”, Morgan-Kaufmann, 2003.

[8] Jeff Heaton, “Programming Spiders, Bots, and Aggregators in Java”, Sybex Publishers, 2002.

[9] Google, Wikipedia, the free encyclopedia, [Online] Available: “http://en.wikipedia.org-/wiki/Google” (Accessed: 7 August 2010)

[10] Bing, Wikipedia, the free encyclopedia, [Online] Available:“http://en.wikipedia.org/wiki/Bing” (Accessed: 7 August 2010)

[11] Yahoo!, Wikipedia, the free encyclopedia, [Online] Available: “http://en.wikipedia.org-/wiki/Yahoo!” (Accessed: 7 August 2010)

[12] Index, Wikipedia, the free encyclopedia, [Online] Available: “http://en.wikipedia.org-/wiki/Index” (Accessed: 28 August 2010)

[13] Crawl, Wikipedia, the free encyclopedia, [Online] Available: “http://en.wikipedia.org-/wiki/Crawl” (Accessed: 28 August 2010)

[14] eHow, Home Page, [Online] Available at: "http://www.ehow.com/" (Accessed: 21 August 2010)

[15] Meta Search Engine, [Online] Available: "http://en.wikipedia.org/wiki-/Metasearch_engine" (Accessed: 5 January 2011)

[16] Google SOAP Search API, [Online] Available: "http://code.google.com/apis/soapsearch-/api_faq.html" (Accessed: 8 January 2011)

[17] JSON, [Online] Available: "http://www.json.org/" (Accessed: 15 January 2011)

[18] JSON to HTML, [Online] Available: "http://json.bloople.net/" (Accessed: 20 January 2011)

[19] NLP, [Online] Available: "http://www.cnlp.org/publications/03NLP.LIS.Encyclopedia-.pdf" (Accessed: 10 February 2011)

[20] Apache OpenNLP, [Online] Available: "http://incubator.apache.org/opennlp/" (Accessed: 10 February 2011)

[21] SourceForge.net: Tokenizer – OpenNLP, [Online] Available: "http://sourceforge.net-/apps/mediawiki/opennlp/index.php?title=Tokenizer" (Accessed: 12 February 2011)

[22] SourceForge.net: POS Tagger – OpenNLP, [Online] Available: "http://sourceforge.net-/apps-/mediawiki/opennlp/index.php?title=POS_Tagger" (Accessed: 12 February 2011)

[23] Related Projects – WordNet, [Online] Available: "http://wordnet.princeton.edu/wordnet-/related-projects/" (Accessed: 9 March 2011)

[24] Java API for WordNet Searching(JAWS), [Online] Available: "http://lyle.smu.edu-/~tspell/jaws/index.html" (Accessed: 9 March 2011)

[25] Building an extension – MDC Doc Center, [Online] Available: "https://developer-.mozilla.org/en/building_an_extension" (Accessed: 20 March 2011)

[26] About WordNet, [Online] Available: "http://wordnet.princeton.edu/" (Accessed: 8 March 2011)

[27] JavaScript Overview, [Online] Available: "https://developer.mozilla.org/en/JavaScript-/Guide/JavaScript_Overview" (Accessed: 21 March 2011)

[28] XUL - MDC Doc Center, [Online] Available: "https://developer-.mozilla.org/En/XUL" (Accessed: 22 March 2011)

ACKNOWLEDGEMENT

First and foremost, we would like to thank our Project Guide, Ms. Kumkum Saxena, who has supported and encouraged us from the very first year of our engineering at Mumbai University. We have always had all our doubts cleared by Ms. Saxena. Every time when we encountered difficulties, we regained confidence affected by Ms. Saxena's strong, positive and uncluttered thinking.

We have been taught the subject of Data Warehousing, Mining and Business Intelligence by Ms. Madhuri Rao. The techniques learnt during this course have been of great assist during the formative process of this project.

The project lab-in-charge Mr. Aaditya Dalvi has been very supportive and helpful during the entire duration of our project development. We would like to take this opportunity to thank him. Also we would like to thank all those who contributed their browsing patterns for the analysis to be carried during the personalization module of our project.

We would also like to extend our sincere gratitude to Mr. Arun B. Kulkarni, the Head, Department of Information Technology for his perpetual support and understanding of the needs and requirements of his students and his willingness to go out of his way to help and guide us.