WEB INTELLIGENCE

A PROJECT REPORT

Submitted by

ANANDAN.K 71201104004

BANUPRIYA.S 71201104008

RAMESH.N 71201104040

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINNEREING

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2005

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

ertified that this project report **"WEB INTELLIGENCE"** is the bonafide work of **NANDAN.K, BANUPRIYA.S, RAMESH.N"** who carried out the project work under y supervision.

[illegible]

IGNATURE

rof. Dr S.Thangaswamy

EAD OF THE DEPARTMENT

omputer Science and Engineering

umaraguru College of Technology

oimbatore - 641006.

[illegible]

SIGNATURE

Asst.Prof. Ranganathan Dinesh

SUPERVISOR

Computer Science and Engineering

Kumaraguru College of Technology

Coimbatore - 641006.

he candidates with University Register Nos. **71201104004, 71201104008, 71201104040** vere examined by us in the project viva-voce examination held on ...21/04/05......

[illegible] 2/4

INTERNAL EXAMINER

[illegible] 2/4

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We would like to express our gratitude to our beloved **principal Padmanabhan, Ph.D., Kumaraguru College of Technology, Coimbatore,** for his constant encouragement throughout the project.

We wish to thank **Dr.S.Thangaswamy, Ph.D., Head, Department of Computer science and Engineering, Kumaraguru College of Technology, Coimbatore,** for his invaluable guidance and suggestions that encouraged us to complete this project successfully.

We admit our heart full thanks to our project coordinator, **Mrs.P.Devaki, Assistant professor, Department of Computer science and Engineering, Kumaraguru College of Technology, Coimbatore,** for being supportive throughout the tenure of the project.

Our heartfelt thanks and gratitude to our project guide **Mr.Ranganathan Dinesh, Assistant professor, Department of Computer science and Engineering, Kumaraguru College of Technology, Coimbatore,** for extending valuable suggestions, Support and motivation throughout the project work.

We also take this opportunity to extend our sense of gratitude to all faculty members, non-teaching staffs of the computer science department, KCT, Coimbatore, for their guidance and cooperation rendered throughout our project.

ABSTRACT

The Web serves a broad spectrum of user communities. The Internet's rapidly expanding user community connects millions of workstations. Users from different backgrounds, interests, and usage purposes are using the web. Only a small portion of the web contains relevant or useful information. These information needs to be retrieved whenever it is needed by users. In this scenario, the project entitled **"Web intelligence"** is focused mainly on providing Internet users with a more efficient search tool.

To try and find relevant required information on the Internet is a big task even with the available search engines. The "Web Intelligence System" is to fully discover the immense information available on the web using various approaches. The main objective is to provide more RELEVANT high quality information.

As existing now, the Internet web based search engine searches, and provides voluminous results. That is currently assessed and found to be providing redundant information. In our project we have come up with a solution of improvement by using the method of user feedback to minimize this redundancy and hence provide relevant information as previously stated. Hence this web intelligence system would provide the Internet community with search results that will satisfy their time and effort constraint better.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS, ABBREVIATIONS, AND NOMENCLATURE

Web Crawler:

Web crawlers are software programs that automatically traverse the web and return the webpage content and URLs in which they retrieved the web content. And it is used

- To maintain mirror sites for popular Web sites.
- To test web pages and links for valid syntax and structure.
- To monitor sites to see when their structure or contents change.

Polysemy:

Highly relevant documents may not contain keywords that explicitly define the topic, a phenomenon known as the polysemy problem.

CGI:

The Common Gateway Interface (CGI) is not a programming language. W3C defines CGI as "the specification, which defines how a program interacts with the web server".

URL:

Uniform Resource Locater is used by the CGI program for locating a particular file or program.

JVM:

Byte code is a highly optimized set of instructions designed to be executed by the java run-time system, which is called the Java Virtual

API:

Application Programming Interface (API) that allows access to relational DB such as Access, Oracle and SQL server.

HTTP:

The Hypertext Transfer Protocol (HTTP) is an application level protocol is based on a request/response paradigm. HTTP communication is initiated by a user agent and consists of a request to be applied to a resource on some origin server. In simplest case, this may be accomplished via a single connection between the user agent and the origin server.

HTML:

HTML is a Hypertext Markup Language for telling a web browser how to format and display a web page.

SERVLETS:

Servlets are small programs that execute on the server side of a Web connection. Servlets dynamically extend the functionality of a Web server.

1. INTRODUCTION

CHAPTER# 1

1. Introduction

1.1Existing System and Its Limitations

The currently available search engines use keyword based index and provides voluminous results for a single search. Moreover it does not rank the pages based on the usage of web content. Hence lot of irrelevant results is generated. Certain banned sites information too should be displayed in search results based on the nationality. This can be achieved only if it is known from where the request is arising. Hence maintaining the user profile is necessary and login facility should be provided. The login facility and feedback based ranking is not provide in the existing keyword based search engines. Hence relevant results are not produced in these systems.

1.2 Proposed system and its advantages

Proposed system aims to provide more relevant information from the web to the internet community by maintaining the user profile and providing ranking to web resources based on user feedback. Login facility is provided to keep track of users and to generate results based on nationality of the user. This feature will facilitate the system to remove the resource pages that are banned by certain nations from available results. Proposed systems will posses the following

- Feedback based ranking
- Search results based on user profile data
- User login facility
- Keyword based search

2. LITERATURE SURVEY

CHAPTER# 2

2.0 Literature Survey:

2.1 Java Servlets

Java Servlets provides web developers with a simple, consistent mechanism for extending the functionality of a web server and for accessing existing business systems. A servlet can almost be thought of as an applet that runs on the server side without a face. Java servlets have made many web applications possible.

Servlets are the Java platform technology of choice for extending and enhancing web servers. Servlets provide a component-based, platform-independent method for building web-based applications, without the performance limitations of CGI programs.

A servlet is similar to a proprietary server extension, except that it runs inside a **Java Virtual Machine** (JVM) on the server, so it is safe and portable. Servlet can be defined as a mini server side program, similar to an applet. Servlets are loaded and executed by a web server in the same way as applets.

Today, servlets are a popular choice for building interactive web applications. Third party servlet containers are available for Apache Web Server, iPlanet Web Server, Microsoft IIS, and others. Servlet containers can also be integrated with web-enabled application servers, such as BEA Web Logic Application Server, IBM Web Sphere, iPlanet Application Server, and others.

A servlet accesses requests from the client, performs some task, and returns results. The following are the basic steps of using servlets.

- The Client makes a request.
- The web server forwards the request to the servlet after receiving it from the client.
- Some kind of process is done by the servlet after receiving the Http request.
- A response is returned back to the web server from the servlet.
- The web server will forward the response to the client.

Unlike CGI and fast CGI, which use multiple processes to handle separate programs and/or separate request, separate threads with in the web server process handle all servlets. This means that servlets are also efficient and scalable. Because servlets run within the web server, they can interact very closely with the server to do things run within the web server, they can interact very closely with the server to do things that are not possible with CGI scripts.

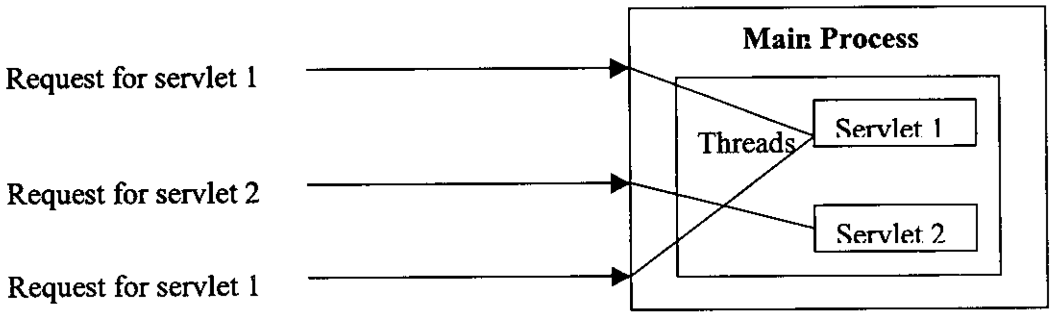


Fig 2.1Java servlet-based web server

2.1.1 The Power Of Servlets

So far, it has been portrayed servlets as an alternative to other dynamic web content technologies, but it hasn't really explained why we should use them.

What makes servlets available choice for web development? We believe that servlets offer a number of advantages they are,

- Portability
- Efficiency and endurance
- Safety
- Elegance
- Integration

Portability

Because servlets are written in Java and conform to a well defined and widely accepted API, they are highly portable across operating systems and across server implementations. A servlet developed on a Windows NT machine running the Java Web Server can be later deployed effortlessly on a high-end UNIX server running Apache.

With servlets, we can truly "write once, serve everywhere".

Efficiency and Endurance

Servlets, in general, are naturally enduring objects. Because a servlet stays in the server's memory as single object instance, it automatically maintains its state and can hold on to external resources, such as database connections, that may otherwise take several seconds to establish.

Safety

Servlets support safe programming practices on a number of levels. Because they are written in Java, servlets inherit the strong type safety of the Java language. In addition, the Servlet API is implemented to be type-safe Servlets can handle errors safely, due to Java's exception-handling mechanism. If a servlet divides by zero or performs some other illegal operation, it throws an exception that can be safely caught and handled by the server, which can politely log the error and apologize to the user.

Elegance

The elegance of servlet code is striking. Servlet code is clean, object oriented, modular, and amazingly simple. One reason for this simplicity is the Servlet API itself, which includes methods and classes to handle many of the routine chores of servlet development. Even advanced operations, like cookie handling and session tracking, are abstracted into convenient classes.

Integration

Servlets are tightly integrated with the server. This integration allows a servlet to cooperate with the server in ways that a CGI program cannot. For example, a servlet can use the server to translate file paths, perform logging, check authorization, perform MIME type mapping, and, in some cases, even add users to the server's user database. Server-specific extensions can do much of this, but the process is usually much more complex and error-prone.

2.1.2 The Servlet API

This is the one of the API provided by Sun Corporation, for developing Java Servlets. The package name is JSDK (Java Servlet Development Kit). It is available in 2.0, 2.1, 2.2 versions. This package is used to develop HTTP servlets, or any kind of servlets, for that matter. Servlets use classes and interfaces from two packages

- Javax.servlet

- Javax.servlet.http.

Javax.servlet package contains classes to support generic, protocol-independent servlets. The classes in the javax.servlet.http package to add HTTP specific functionality extend these classes. The top-level package name is javax instead of familiar java, to indicate that the Servlet API is a standard extension.

Every servlet must implement the javax.servlet.Servlet interface. Most servlets implement it by extending one of the two special classes: Javax.servlet.GnericServlet or javax.servlet.http.HttpServlet. A protocol-independent servlet should subclass Generic Servlet, while an HTTP servlet should subclass HttpServlet, which itself is a subclass of Generic-Servlet with added HTTP-specific functionality.

Unlike a regular Java program, and just like an applet, a servlet does not have a **main ()** method. Instead, the server in the process of handling requests invokes certain methods of a servlet. Each time the server dispatches a request to a servlet, it invokes the servlet's **service ()** method.

A generic servlet should override its service () method to handle requests as appropriate for the servlet. The service () method accepts two parameters: a **request object** and a **response object**. The request object tells that servlet about the request, while the response object is used to return a response.

Servlet Supporting Web Server

- Sun's Java Web Server ("Jeeves")
- Microsoft's Internet Information Server

- ❖ Apache Web Server
- ❖ JSDK (ServletRunner & Start Server)
- ❖ iPlanet Web Server
- ❖ BEA Web Logic Application Server

2.1.3 Servletrunner

Once you have written your servlet, you can run it in many web servers, or in the **servletrunner.** Where ever you decide to run your servlet, there are certain pieces of data that you might want or need to specify. When you are using servletrunner you do this with properties.

Using Servlet Runner

The servletrunner is a small utility, intended for testing. It is multithreaded, so it can run more than one servlet. It can be used, therefore, to run multiple servlets simultaneo0usly, or to test one servlet that calls other servlets in order to satisfy client requests. Unlike some web servers, it does not automatically reload servlets when they re updated. Because it is small, however, there is very little overhead associated with stopping and restarting it in order to use a new version of a servlet.

The servletrunner is in the <jdk>/bin directory. Invoking it with the help flag shows a usage message without running it:

```
% . /bin/servletrunner    -help
Usage: servletrunner [options]
```

Options:

-p port	the port number to listen on
-b backlog	the listen backlog
-m max	maximum number of connection handlers
-t timeout	connection timeout in milliseconds
-d dir	servlet directory
-r root	document root directory
-s filename	servlet property file name
-v verbose	output

In order to see the default values of these options, you can call servletrunner with the –v switch. This will, however, start the servlet runner. Just stop it after you have obtained the information, if you are not ready to run it yet, or want it to run with something other than the default values.

```
% ./bin/servletrunner –v
```

```
servletrunner starting with settings:
  port = 8080
  backlog = 50
  max handlers = 100
  timeout = 5000
```

```
servlet dir = .\examples
document dir = .\examples
servlet profile = .\examples\servlet.properties
```

Once the servletrunner is executing, you run servlets by calling them directly in your browser, or as the forms example shows, by using a form that calls a servlet to process its data. The URL for a servlet has the following general form:

Http: / /machine-name: port / servlet / servlet-name

where servlet-name corresponds to the name you have given your servlet. For example, to run the Phone Servlet, which has the property servlet.phone.code = PhoneServlet, you would use the following URL. (It assumes that servletrunner is running on your machine, localhost, at port 8080, and that the phone servlet is located in the servlet directory provided to servletrunner at startup:

http: / / localhost : 8080 /servlet /phone

2.2 JDBC (Java Data Base Connectivity)

JDBC is a SQL-level API –one that allows you to execute SQL statements and retrieve the results, if any. This API is a set of interfaces and classes designed to perform actions against any database.

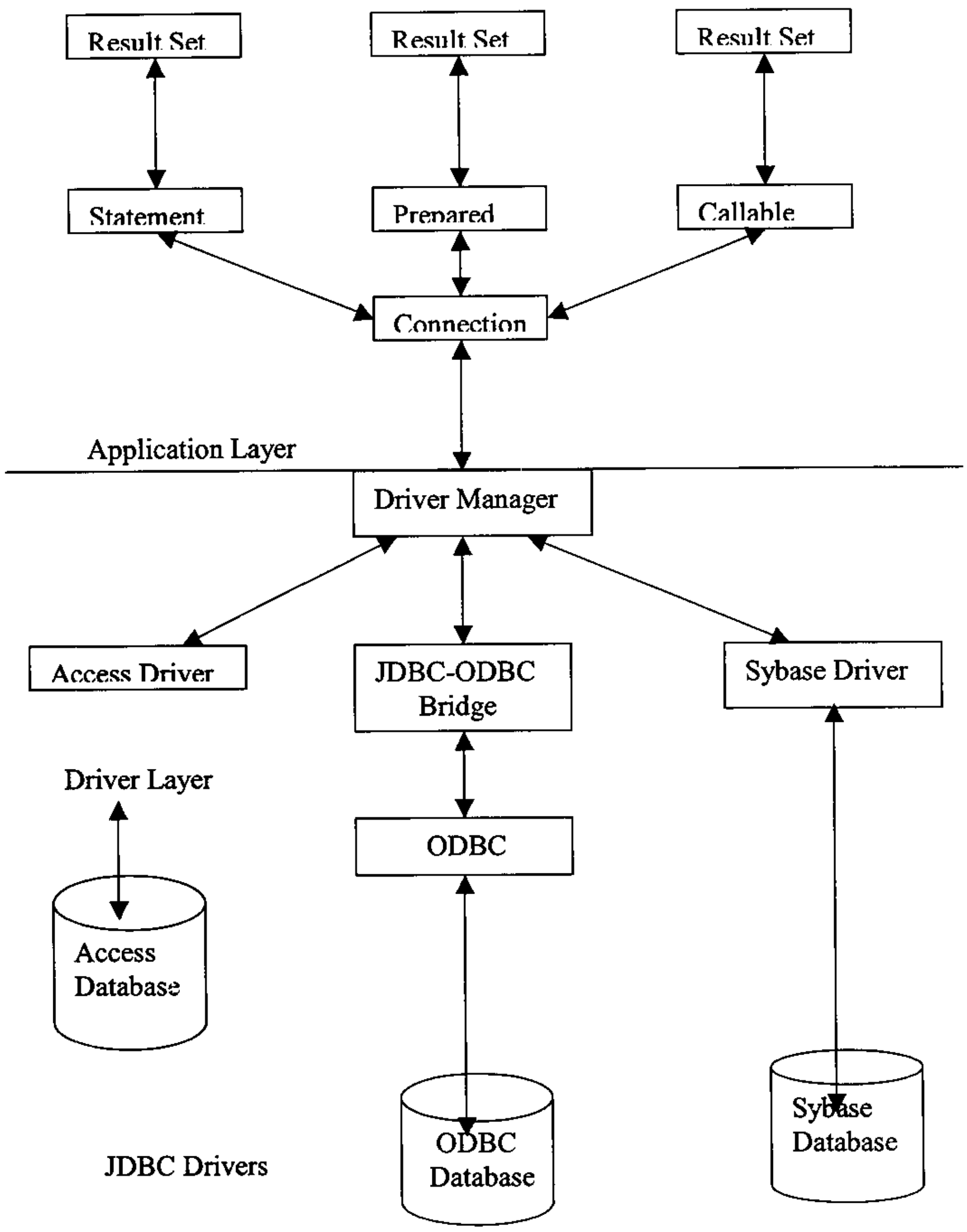


Fig 2.2 JDBC-ODBC connectivity

The JDBC API, found in the java.sql package, contains only a few concrete classes. Much of the API is distributed as database-neutral interface classes that specify behavior without providing any implementation. Third-party vendors provide the actual implementations.

An individual database system is accessed via a specific JDBC driver that implements the java.sql.Driver interface. Drivers exist for nearly all-popular RDBMS systems, though few are available for free are available free. Sun bundles a free JDBC-ODBC bridge driver with the JDK to allow access to standard ODBC data sources, such as a Microsoft Access database. However, Sun advises against using the bridge driver for anything other that development and very limited deployment. Servlet developers in particular should heed this warning any problem in the JDBC-ODBC bridge driver's native code section can crash the entire server, not just your servlets.

JDBC drivers are available for most database platforms, form a number of vendors and in number of different flavors. There are four driver categories:

Type 1 JDBC-ODBC Bridge Driver

Type 1 driver use a bridge technology to connect a Java client to an ODBC database service. Sun's JDBC-ODBC bridge is the most common type 1 driver. These drivers are implemented using native code.

Type 2 Native-API Partly-Java Drivers

Type 2 drivers wrap a thin layer of Java around database-specific native code libraries. For Oracle databases, the native code libraries might be based on the OCI (Oracle Call Interface) libraries, which were originally designed for

C/C++ programmers. Because Type 2 drivers are implemented using native code, in some cases they have better performance that their all-Java counter parts. They add an element of risk, however, because a defect in driver's native code section can crash the entire server.

Type 3 Net-Protocol All-Java Driver

Type 3 drivers communicate via a generic network protocol to a piece of custom middleware. The middleware component might use any type of driver to provide the actual database access. Web Login's Tengah product line as an example. These drivers are all Java, which makes them useful for applet deployment and safe for servlet deployment.

Type 4 Native-Protocol All-Java Driver

Type 4 drivers are the most direct of the lot. Written entirely in Java, Type 4 drivers understand database-specific networking protocols and can access the database directly without any additional software.

A list of currently available JDBC drivers can be found at http://java.sun.com/products/jdbc/jdbc.drivers.html.

2.3 About Java Script:

Java Script is an Interpreted language, rather than a compiled language. The Java script code runs inside a web page loaded in to a browser. All we need to create is a text editor like windows notepad, and a web browser such as Netscape Navigator or Internet Explorer, with which we can view our pages. These browsers come equipped with Java Script interpreters. Basically the job of a web browser is to hold lots of web pages on its hard drive. Then when a browser usually on a different computer requests a web page that is contained on that web server, the web server load it from its own hard drive and then passes the page back to the requesting computer via a special communication protocol called "Hyper Text Transfer Protocol (HTTP)". The computer running the web browser that makes the request is known as the Client and the computer satisfying the clients request is called the Server.

The main reason for using Java Script in this project is its wide spread use and availability. Java Script is very versatile and not just limited to use within a web browser.

2.3.1 Advantages of Java Script

Java Script is having more advantage over competing script languages like VB Script, this includes…

1. Capable of Running in any Browser

The Java Script is capable of running in any browser like Netscape Navigator, Internet Explorer and others, but the other scripting language VB Script can be run only on Internet Explorer and Perl.

2. Managing Administration Tasks

Java Script is used to automate computer administration tasks. It can also be used for Interacting with the users and getting information from them and validating their actions.

2.4 About Ms-Access:

Access is used as the back end. It allows creation of tables and queries in an easier way and it does not place much constraint before the user. Conversion of existing ODBC databases is also a good advantage of using Access. Forms can also be prepared in Access and when converted into MDE files give us a read only design of code and forms.

Access provides extensive new features to easily use the Internet and develop a World Wide Web (WWW) application. We need a web browser, such as Microsoft Internet Explorer, and a modem, Intranet Connection, or other network connection to access the Internet and take advantage of these features.

Access provides complete documentation for the tables created and also provides the option of viewing the relationships used in creating tables. Foreign Key can be created using the relationship window.

Access also comes with Wizards under all its options i.e., under Tables, Queries, Reports etc., All these Wizards help the user to help the user to work in an flexible and interactive environment.

We can use Access to create a World Wide Web Application. For example, we can create a corporate home page, an online magazine or newsletter, a registration system for a trade show, or online product catalog. To create this web application, we output objects to HTML format or use the Publish to the Web Wizard.

We can export reports to static HTML format and you can export datasheets and forms to static and dynamic HTML format. Access created one web page for each report page, datasheet, and form you export. Exporting objects to HTML format is useful for creating a simple web application, verifying the format and appearance of an object's output, or adding files to existing web applications.

3. ARCHITECTURE OF THE SYSTEM

CHAPTER# 3

3. Architecture of the System:

The architecture of the system consists of three parts. First it encompasses an efficient crawler which crawl the web and generates searchable content and place into the database. Crawler requests the web server for the particular URL and returns the searchable keywords and its associated URLs. Whenever it finds new URL it adds it to the URL queue, those URLs will be processed on later stage. The same is repeated by the crawler as long as there is URL in the queue. It performs ranking operation whenever it founds new word/repeated word in the web content. The result of the web crawler is placed in the main database file which can be searched later on when the query arises from the user.

The second part consists of searching and generating results against the user queries. Whenever the user enters a search query the validity of login session is verified and search operation is performed. The search results are generated by based on the user profile. The user profile consists of many details including area of interest and nationality. User profile details are placed in a separate database.

The third part consists of feedback processing, in which the user feedbacks are received and ranking of available web content is changed based on the user feedback. This operation enables the system to identify the most valuable web resource from the list of available web resources.

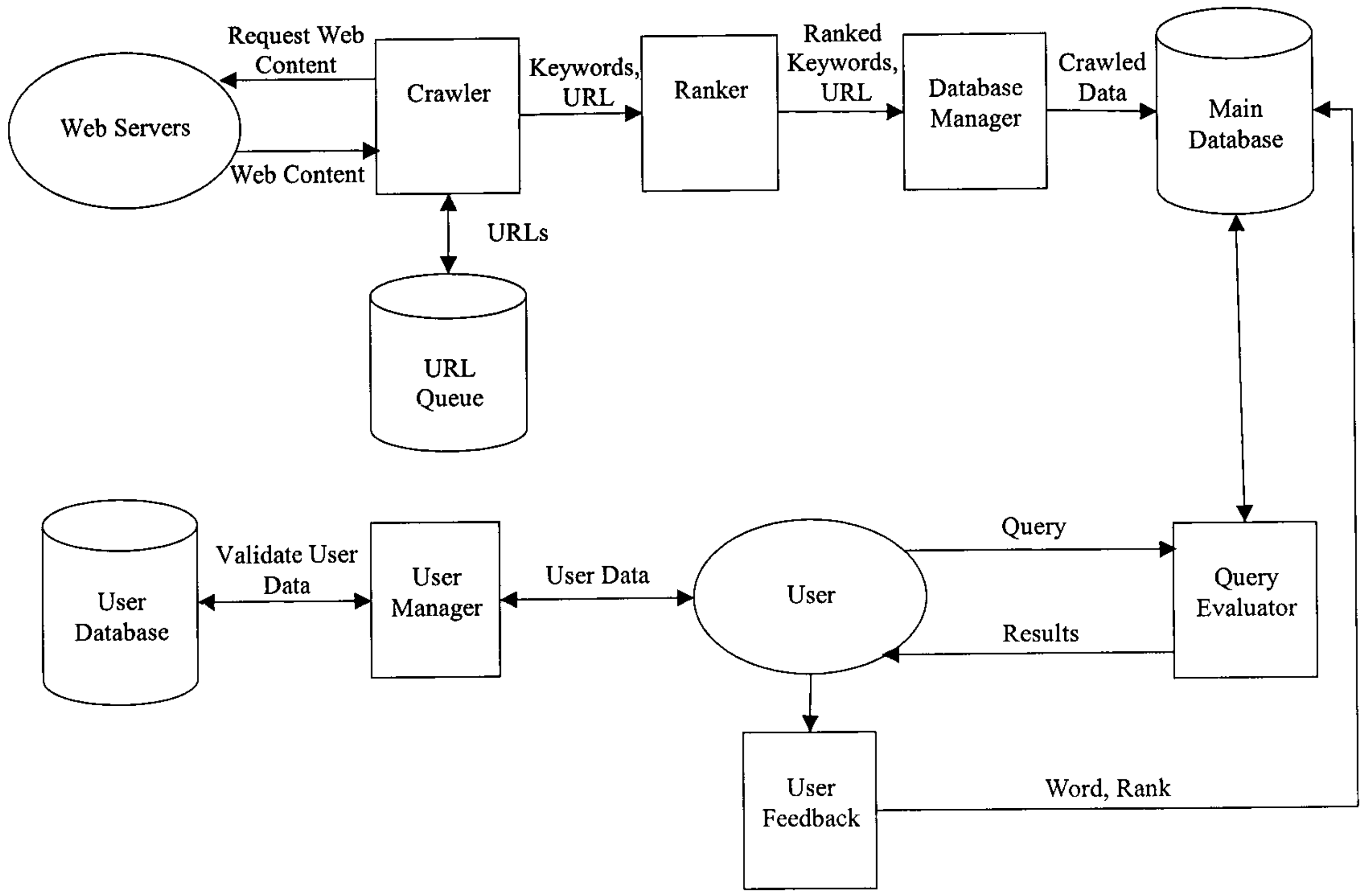


Fig 3.1 System Architecture

4. SYSTEM DESIGN

CHAPTER# 4

4.0 System design

4.1 System architecture description

The system provides the user interface for creating new user profile, updating user profile, interface for login facility and search query. In addition to this it provides user feedback facility in each search result page. Based on the feedback ranking is varied. System posses an efficient web crawler which crawls the web and retrieves the ranked keywords along with URLs .

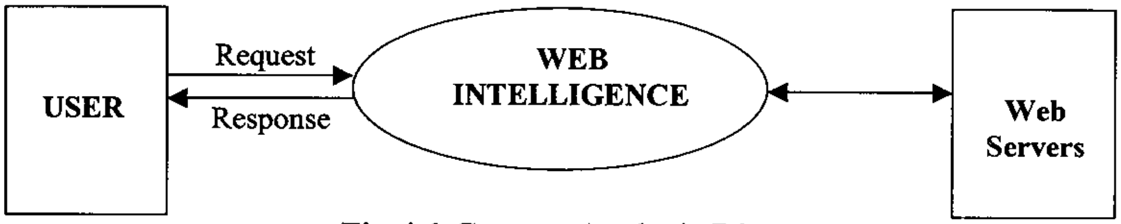


Fig 4.1 Context Analysis Diagram

4.1.1 Overview of Classes

The system is divided in terms of many classes. The overview of classes and their functionalities are listed below.

Classes	Functionality
Crawler	Crawls the web resources and returns the keywords and URLs
DbStore	Store the hash table contents into the database
Index	Displays index page for login
NewUserIndex	Displays HTML page for user profile creation
Newuser	Accepts new user profile after validation in the client side
Login	Creates and maintain session after validating user login and password
Queryevaluator	Evaluates the user query and generates the results.
Feedranker	Retrieves feedback from the user and change ranking of the web resource
Editprofile	Retrieves the already existing profile details and display to user.
Updateprofile	Updates the already existing profile details
Logout	Ends the user session

Table 4.1 Classes And Functionality

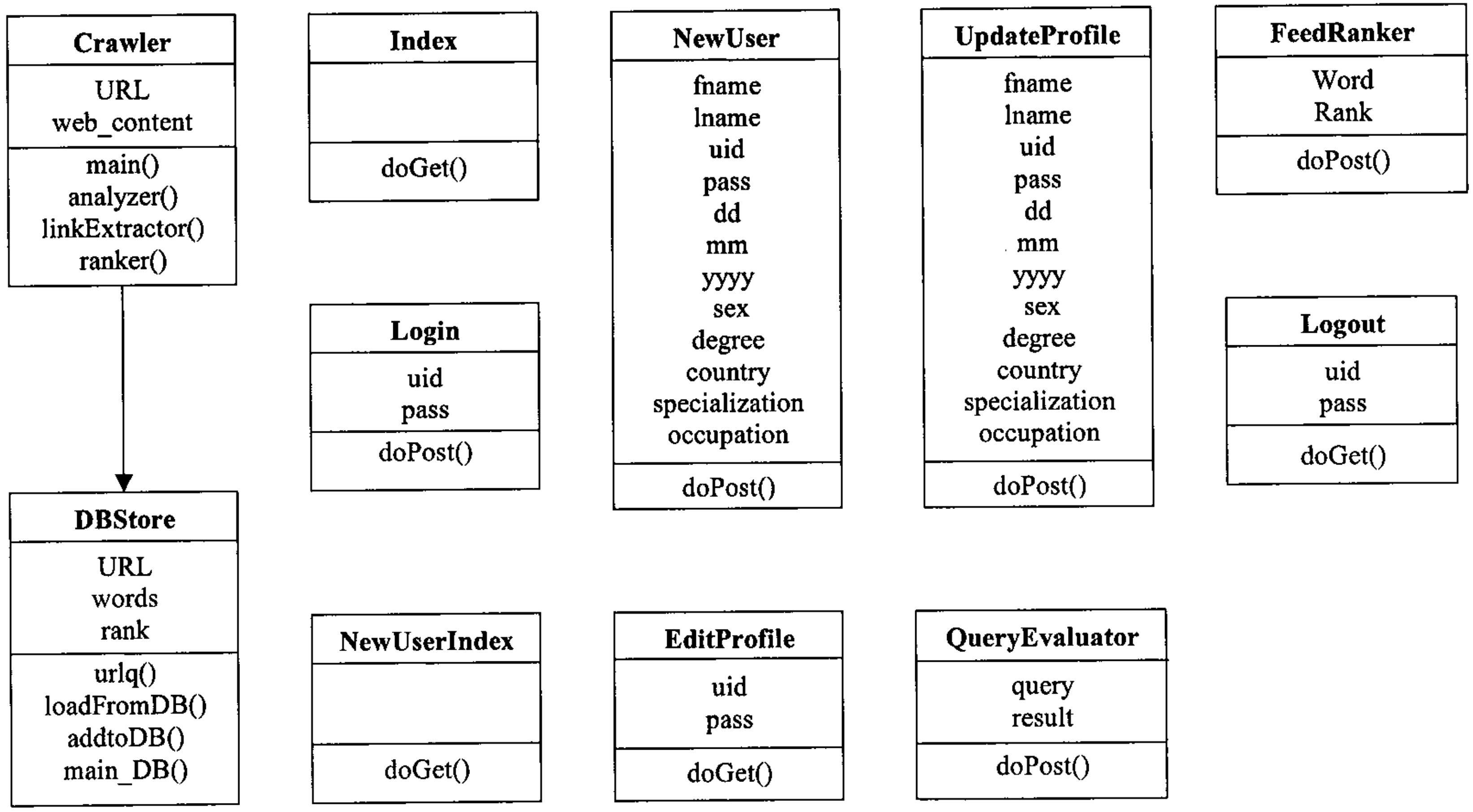


Fig 4.2 Class Diagram

4.1.2 Structure and relationships

The operational sequence of the entire system is depicted in the collaboration diagram which shows the relationship of the entire system. Whenever the user enters the system he/she is prompted for login name and password. If user already has login user can continue to search page after logging into the system. If not new user profile can be created. Already existing profile too can be updated according to the user wish as soon he enters into the system.

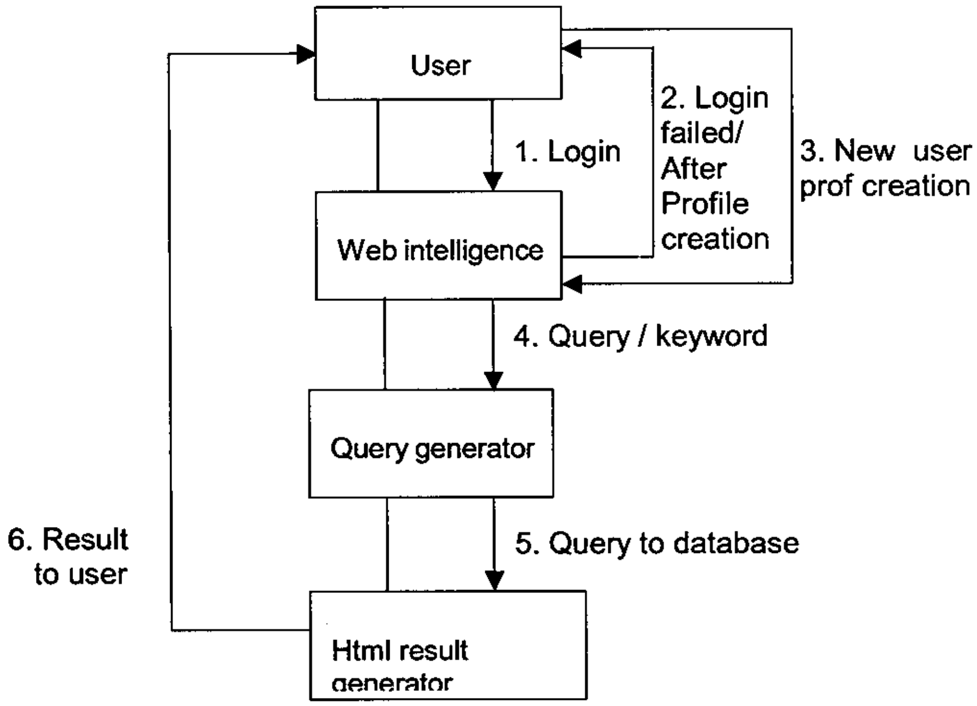


Fig 4.3 Collaboration Diagram

4.1.3 User interface issues

The user is provided with an efficient easy to use interface. The interface implemented in the system can be very well expressed in terms of activity diagrams and use case model. The activity diagram shows the sequence of operations the user performs as soon as he/she enters into the system. The user activities were designed using UML activity diagram.

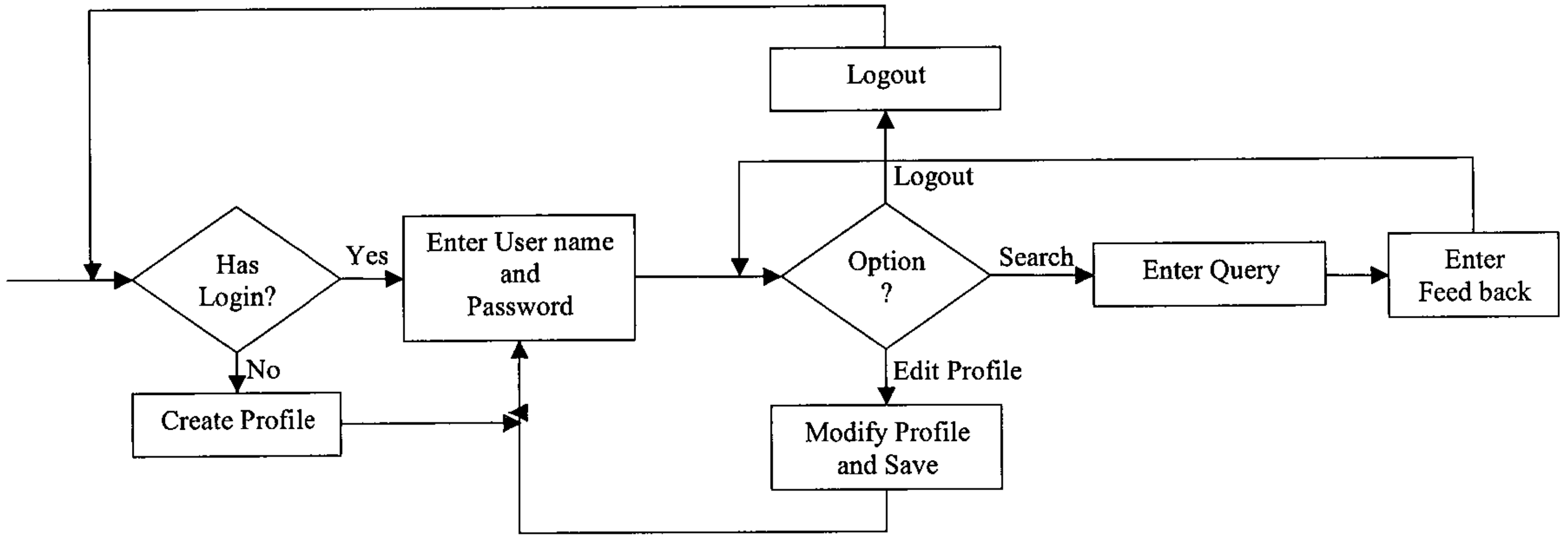


Fig 4.4 Activity Diagram

4.1.4 Usecase Model:

The usecase model describes the use of the system and shows the courses of events that can be performed. It shows the system in terms of how it is used from the user point of view.

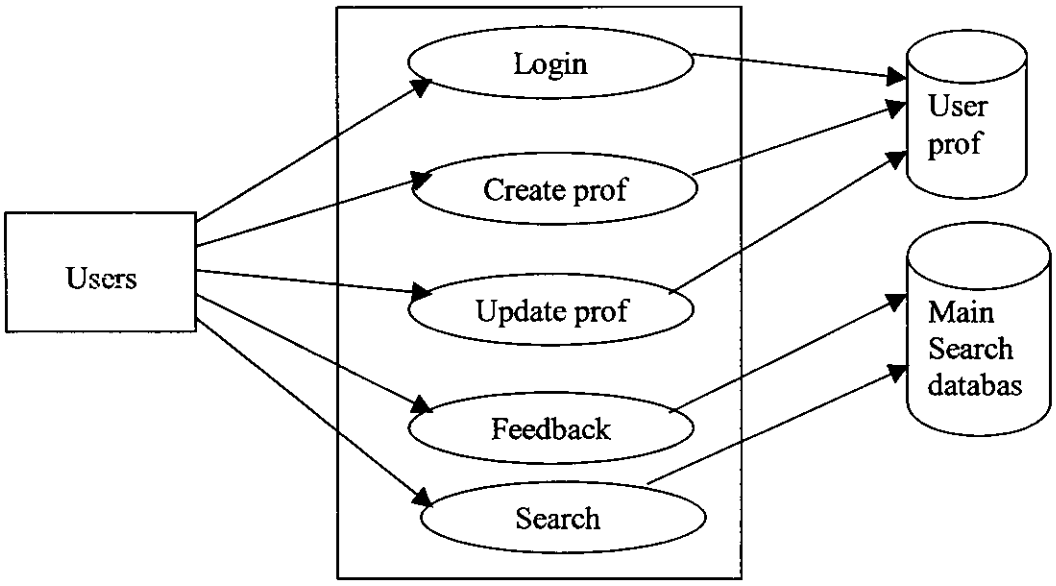


Fig 4.5 Use Case Diagram

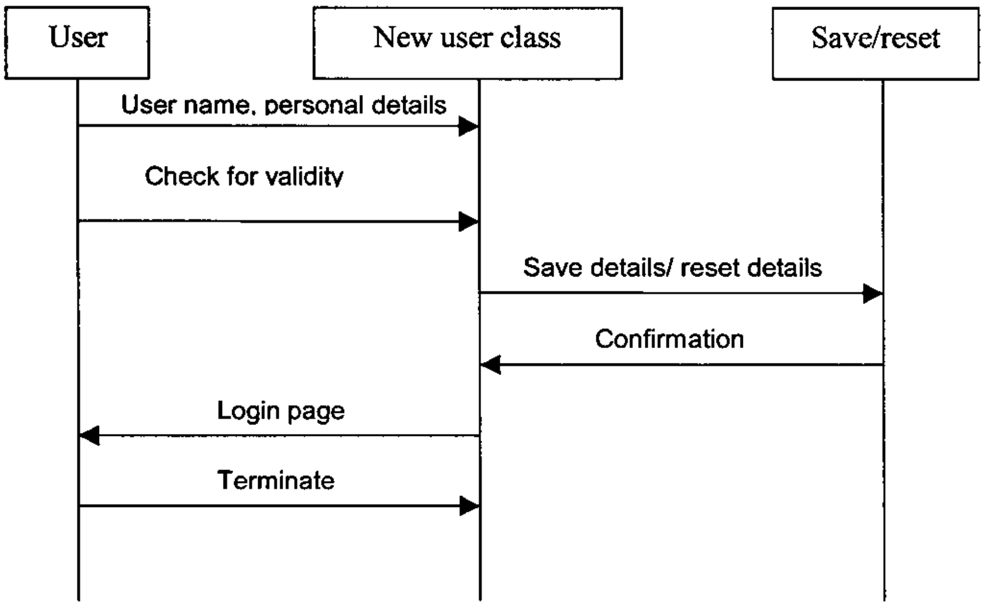


Fig 4.6 Sequence Diagram for Profile Creation

4.1.5 Table Design:

Field name	Type	size	Description
fame	Text	30	First name
lname	Text	30	Last name
uid	Text	15	User id
pass	Text	15	Password
DD	Number		Date of birth
MM	Number		Month of birth
YYYY	Number		Year of birth
sex	Text	10	Sex
country	Text	20	Nationality
degree	Text	10	Degree
specialization	Text	30	Field of interest
occupation	Text	20	Working area

Table 4.2 User profile table structure

5. PROGRAMMING ENVIRONMENT

CHAPTER# 5

5.0 Programming Environment

5.1 Hardware Requirements:

1. Processor:	Intel Pentium 4 Processor
2. Processor Speed:	1.8 GHz
3. Hard Disk:	40 GB IBM HDD
4. Memory:	256 MB SD RAM
5. Operating System:	Windows 98, Windows XP
6. Backup Devices:	3 ½ inch floppy drive, Maxell Floppy Disk, Samsung CD Writer Drive, Sony Optical Rewritable Disk (4x)

5.2 Software Requirements:

1. Java2 SDK 1.4
2. MS Access 2002

6. DETAILED DESIGN

CHAPTER# 6

6.0 Detailed Design:

The detailed design gives the list classes in the system along with the details such as identification, type, purpose and functions. The table given below shows the list of classes, files and their identification, purpose and the functions provided by them

Identification	Type	Purpose	Functions

Crawler	Class	To retrieve web content and rank the web resources.	analyser()
			linkextractor()
			ranker()

Dbstore	Class	Stores the crawler generated output into the main.db file.	urlq()
			main_db()
			addToDB()
			loadFromDB()

Index	Class	Displays index page for login.	doGet()
NewUserIndex	Class	Displays HTML page for user profile creation.	doGet()
Newuser	Class	To create new user profile	doPost()
Editprofile	Class	To display user profile details when updating is required by user.	doGet()
Updateprofile	Class	To make changes in the existing profile.	doPost()
Login	Class	To create and maintain login session.	doPost()
Queryevaluator	Class	Accepts user query and search the main.db file and generates search results based on user profile.	

Identification	Type	Purpose	Functions
Feedranker	Class	To change the rank of the available web page based on user feedback.	doPost()
Logout	Class	To terminate the user session.	doGet()
main.db	DBF file	To store the crawler generated output	Not applicable
init_url.db	DBF File	To place the initial URL list by the administrator	Not applicable
urlq.db	DBF File	To maintain the queue of URLs that is to be traversed by crawler	Not applicable
userdb.mdb	DBF File	Access database file which holds the user profile	Not applicable
<country>.db	DBF file	Holds the list of URLS banned by the <country>	

Table 6.1 Classes and Files List

7.FUTURE ENHANCEMENTS

CHAPTER# 7

7.0 Future Enhancements

The **Web Intelligence System** has been developed to optimize the search results and provide more relevant results to the internet community. System can be enhanced further by utilizing the user profile for search results generation. The simple easy to use interface is provided in the client side.

7.1 Scope for Future Enhancements

- Graphics based interface can be provided in the server side.
- Multithreaded crawler can be developed so that efficiency of crawler can be improved.
- User profile can be fully utilized for search result generation.
- Image search can be provided.
- Phrase based search facility can be provided.
- Polysemy problem can be solved in future versions.
- Relevant keywords can be found out for the search query and results can be generated for the relevant keyword too.
- User events tracking can be done to rank the pages
- Care can be taken to generate result pages to without dead links.
- User can be provided with an option to the relevant area while entering the search keyword.

8. CONCLUSION

CHAPTER# 8

8.0 Conclusion:

The complete design and development of the system is presented in this dissertation. The system has user-friendly features. It is possible for any user to use this system.

The programming techniques used in the design of the system provide a scope for future expansion and implementation of any changes, which may occur in the future. The system has tested by with many client side browsers and they provide satisfactory performance.

Since the requirements of internet community and their standards are changing day by day the system has been designed in such a way that its scope and boundaries could be expanded in future with little modifications. Since object oriented approach is followed throughout the development process changes can be made very easily. As a further enhancement this system can be integrated with any other system.

The main aim behind the development of this system is to facilitate the internet community to explore the resources in the web and to retrieve more relevant results.

9. REFERENCES

CHAPTER# 9

9.0 References

9.1 Journal papers

1) **"Data Mining for Web Intelligence"**, IEEE Computer Society, November-2002, Jiawei Han, Kevin Chen-Chuan Chang, University of Illinois
2) **"Mining the web's link structure"**, IEEE Computer Society, Byron E. Dom , S. Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, Andrew Tomkins, David Gibson
3) **"Development of a Self – Adaptive Web Search engine",** IEEE Computer Society, W.Zhang, B.Xu, H.Yang

9.2 Books

1) **"The Complete Reference Java2"**, Herbert Schildt, Edition 5
2) **"Java Script Bible",** fifth edition by Danny Goodman, Michael Moorison
3) **"Database Programming with JDBC and Java",** George Reese
4) **"How to Do everything with HTML"**, James H. Pence
5) **"Java Servlet programming",** second edition by Jason Hunter

9.3 Web Sites

1. *http://www.java.sun.com/*
2. *http://www.servlets.com/*
3. *http://www.javascript-coder.com/*
4. *http://www.w3schools.com/*
5. *http://www.webreference.com/*

10. APPENDIX

CHAPTER# 10

10.0 Appendix:

10.1 Sample Code:

10.1.1 Crawler.java

```
// crawls the web and retrieves keywords and links from the visited URLs
import java.util.*;
import java.io.*;
import java.net.*;
import java.lang.*;
public class Crawler implements Serializable{
protected static LinkedList q;
protected static Hashtable rank_ht;
protected static Hashtable htab;
protected static Hashtable htab1;
protected static URL u;
protected static String tit;

public Crawler(){
        rank_ht=new Hashtable();
}//End of constructor

public static void main(String args[]) throws Exception
{
 FileInputStream fi=new FileInputStream("init_url.db");
 LineNumberInputStream li=new  LineNumberInputStream(fi);
 DataInputStream ds=new DataInputStream(li);
 rank_ht=new Hashtable();
```

```
htab1=new Hashtable();
DBStore. loadFromDB();//Loads data to hashtable from the main.db
q=new LinkedList();
String inpu;
while((inpu=ds.readLine()) != null)
  q.addLast(inpu);
 while(q.size()>0)
 {
 u=new URL((String)q.removeFirst());
 System.out.print("Processing: "+u+"   ");
 boolean b=DBStore.urlq(u.toString());
  if(b)
  {
   System.out.println("Already Processed: "+u);
   continue;
  }
 if(analyser(u.toString()))
   continue;
 String ranked_words =new String();
 ranked_words=rank_ht.toString();
 rank_ht.clear();
 StringTokenizer ranked_st=new StringTokenizer(ranked_words,", {}");
 int tot_words=ranked_st.countTokens();
 while(ranked_st.hasMoreElements())
 {
  String s_word = ranked_st.nextToken();
  StringTokenizer ranked_st1=new StringTokenizer(s_word,"=");
  int tc=ranked_st1.countTokens()/2;
 {
  DBStore.main_db(u,ranked_st1.nextToken(),(1000000)-((999999 /tot_words )
* (Integer.parseInt(ranked_st1.nextToken()))));
 }//End of for(int i=0;i<=tc;i++)
 if(tit.length()>0)
 {
 StringTokenizer titl=new StringTokenizer(tit);
 while(titl.hasMoreTokens())
 {
  DBStore.main_db(u,titl.nextToken(), (2*tot_words)/100 );
 }//End of while(st.hasMoreToken())
 }
 }//End of while(ranked_st.hasMoreElements())
 System.out.println("Processed: "+u);
 DBStore.addToDB();
}//End of while(q.size()>0)
}//End of Main

public static boolean analyser(String u) throws Exception
{
 /*URL u=new URL("file://c:/wis/copy.htm");
 URLConnection urlcon=u.openConnection();*/
 /*To convert URL into File*/
 u=u.substring(7);
try{
 FileInputStream fin=new FileInputStream(u);
 int n;
 char c;
```

```
while( (n=fin.read())!=-1 )
{
 c=(char)n;
 input=input+c;
}
input=input.toLowerCase();
 /*Title Extracted Here*/
tit="";
int f=input.indexOf("<title>");
int g=input.indexOf("</title>");
f=f+7;
if(f>1 && g>1 && g>f)
   tit=input.substring(f,g);
StringTokenizer st=new StringTokenizer(input,">");
while(st.hasMoreTokens())
 {
  String s=st.nextToken();
  linkExtractor(s);
  if(s.charAt(0)!='<')//Tokens with or without />
  {
   StringTokenizer st1=new StringTokenizer(s,"<");
   String s1=st1.nextToken();
   StringTokenizer st2=new StringTokenizer(s1);
   while(st2.hasMoreTokens())//Very Important. Extracted Words are Taken here
      ranker(st2.nextToken(),1);
   }//End of if(s.charAt(0)!='<')
  }//End of while(st.hasMoreTokens())
 fin.close();
```

```
{
 System.out.println(" URL/File "+u+" not reachable");
 return true;
}
return false;
}//End of analyser

public static void linkExtractor(String s)
{
 StringTokenizer st=new StringTokenizer(s," =");
 while(st.hasMoreElements())
 {
  String s1=st.nextToken();
  s1=s1.toLowerCase();
  if(( s1.compareTo("href") )==0)
  {
   s1=st.nextToken();
   StringTokenizer st1=new StringTokenizer(s1,"\"");
   String s2=st1.nextToken();
   if( (s2.indexOf(".htm")!=-1) || (s2.indexOf(".html")!=-1) )
   {
   if(s2.startsWith("www") || s2.startsWith("http") )
   {
    q.addLast(s2);
   }
   else
    {
     String s3=u.toString();
    int n1=s3.lastIndexOf('/');
    int n2=s3.lastIndexOf('\\');
    if(n1>n2)
     m=n1+1;
    else
     m=n2+1;
    s3=s3.substring(0,m);
   q.addLast(s3+s2);
    }
   }//End of if( (s2.indexOf(".htm")!=-1) || (s2.indexOf(".html")!=-1) )
  }//End of if(( s1.compareTo("href") )==0)
 }//End of while(st.hasMoreElements())
}//End of Link Extractor

public static void ranker(String word,int score)
{
 String s;
 int s1;
 String s2;
if(word.charAt((word.length())-1)=='.')
   word=word.substring(0,( (word.length())-1 ));
 word=word.toLowerCase();
if((s=(String)rank_ht.get(word))!=null)
 {
  s1=Integer.parseInt(s);
  s1=s1+score;
  s2=""+s1;
  rank_ht.put(word,s2);
```

```
else
 {
  rank_ht.put(word,"1");
 }//End else
}//End of ranker
}//End of class
```

10.1.2 DBStore.java

```
//stores the hash table contents into the .db file in objectstream
import java.util.*;
import java.io.*;
import java.net.*;
public class DBStore extends Crawler
{
public static boolean urlq(String ur) throws Exception
{
 String ss;
 if( (ss=(String)Crawler.htab1.get(ur))==null )
 {
  ss=new String("1");
  Crawler.htab1.put(ur,ss);
  try
  {
   FileOutputStream fout1=new FileOutputStream("urlq.db");
   ObjectOutputStream oos1=new ObjectOutputStream(fout1);
   oos1.writeObject(Crawler.htab1);
   oos1.flush();
   oos1.close();
  catch(Exception e)
  {
   System.out.println(e);
  }
  return false;
 }
 else
  return true;
}//End of urlq

public static void main_db(URL ut,String word,int score) throws IOException
{
 word=word.toLowerCase();
/* if(word.charAt((word.length())-1)=='.')
   word=word.substring(0,( (word.length())-1 ));*/
 String ob;
 String ob1=score+">"+ut.toString();
 int i=ob1.indexOf('>');
 for(int k=0;k<(6-i);k++)
  ob1="0"+ob1;
 SortedSet ss=new TreeSet();
 if( (ss =(SortedSet)Crawler.htab.get(word))==null )
 {
  ss=new TreeSet();
  ss.add(ob1);
 }
 else
 {
}
Crawler.htab.put(word,ss);
}//End of main_db

public static void addToDB()
{
try
{
 FileOutputStream fout=new FileOutputStream("main.db");
 ObjectOutputStream oos=new ObjectOutputStream(fout);
 oos.writeObject(Crawler.htab);
 oos.flush();
 oos.close();
}
catch(Exception e)
{System.out.println(e);
}
}//End of addToDB

public static void loadFromDB()
{
try
 {
 FileInputStream fin=new FileInputStream("main.db");
 ObjectInputStream ois=new ObjectInputStream(fin);
 Crawler.htab=(Hashtable)ois.readObject();
 ois.close();
 FileInputStream fin1=new FileInputStream("urlq.db");
```

```
Crawler.htab1=(Hashtable)ois1.readObject();
ois1.close();
}
catch(Exception e)
{
 System.out.println(e);
}
}//End of loadFromBD
}//End of Class
```

10.1.3 Index.java

```
//sends the homepage to the user when requested
import javax.servlet.*;
import javax.servlet.http.*;
public class Index extends HttpServlet{
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
                //Sends the Home page to the client
}//End doGet
}//End class Index
```

10.1.4 NewUserIndex.java

```
//Sends the html page to the client for user profile creation
import javax.servlet.*;
import javax.servlet.http.*;
public class NewUserIndex extends HttpServlet{
```

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
            //Sends the html page to the client for user profile creation
}//End doGet
}//End class NewUserIndex
```

10.1.5 NewUser.java

```
//Validates and creates New user
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;
import java.util.*;
public class NewUser extends HttpServlet{
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
            //Validates and creates New user
}//End doPost
}//End class NewUser
```

10.1.6 Login.java

```
//Creates and maintains the session
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.util.*;
public class Login extends HttpServlet{
```

```
public void doPost(HttpServletRequest requcst, HttpServletResponse response)
throws ServletException, IOException
{
        //Creates and maintains the session
}//End doPost
}//End class Login
```

10.1.7 EditProfile.java

```
//Sends the html page to the client for user profile creation
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;
import java.util.*;
public class EditProfile extends HttpServlet{
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
        //Sends the html page to the client for user profile creation
}//End doGet
}//End class EditProfile
```

10.1.8 UpdateProfile.java

```
//Validates the new information and Updates user's existing profile
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;
```

```
public class UpdateProfile extends HttpServlet{
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
        //Validates the new information and Updates user's existing profile
}//End doPost
}//End class UpdateProfile
```

10.1.9 QueryEvaluator.java

```
//Evaluates user query and generates result page
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class QueryEvaluator extends HttpServlet{
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
        //Evaluates user query and generates result page
}//End doPost
}//End class QueryEvaluator
```

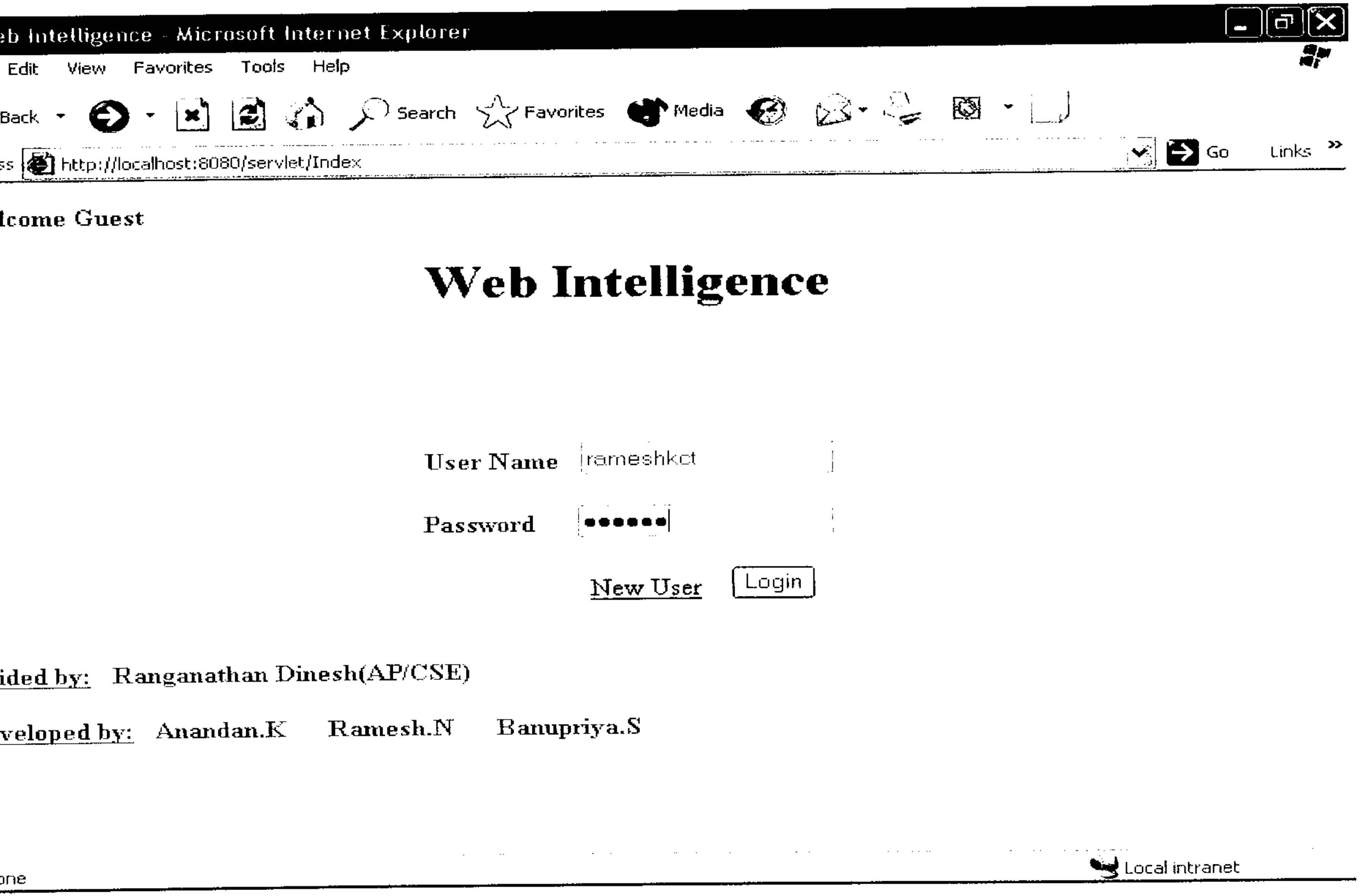
10.1.10 FeedRank.java

```
//Analyses user feedback and ranks the resources
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class FeedRank extends HttpServlet{
public void doPost(HttpServletRequest request, HttpServletResponse response)
{
            //Analyses user feedback and ranks the resources
}//End doPost
}//End class FeedRank
```

10.1.11 Logout.java

```
//Ends the session
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.util.*;
public class Logout extends HttpServlet{
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
            //Ends the session
}//End doGet
}//End class Logout
```

Sample Output:


eb Intelligence - Microsoft Internet Explorer
Edit View Favorites Tools Help
Back Search Favorites Media
ss http://localhost:8080/servlet/Index
Go Links
one
Local intranet

lcome Guest

Web Intelligence

User Name rameshkct

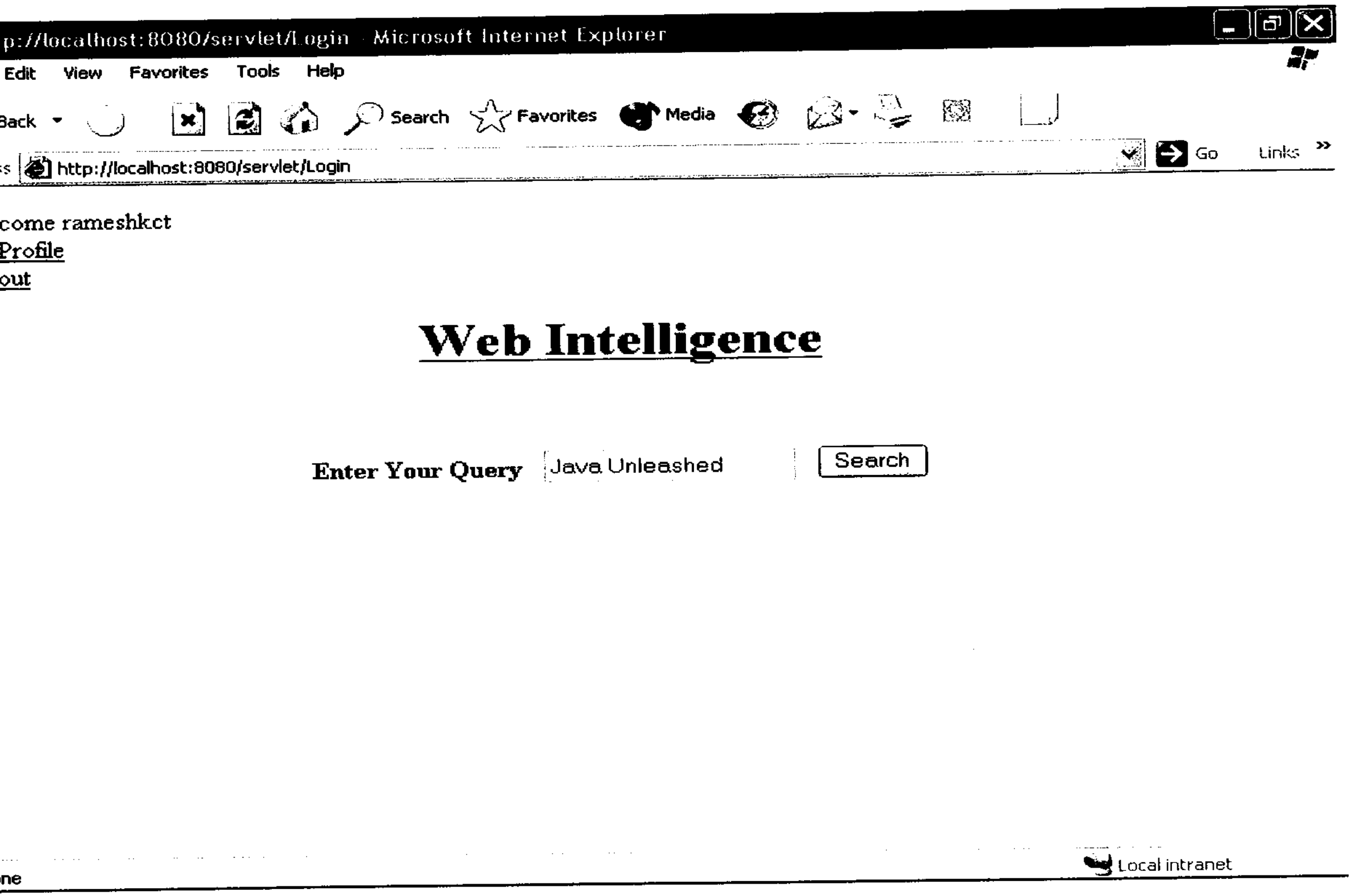
Password ●●●●●●●

New User Login

ided by: Ranganathan Dinesh(AP/CSE)

veloped by: Anandan.K Ramesh.N Banupriya.S

Fig 10.1 Home Page

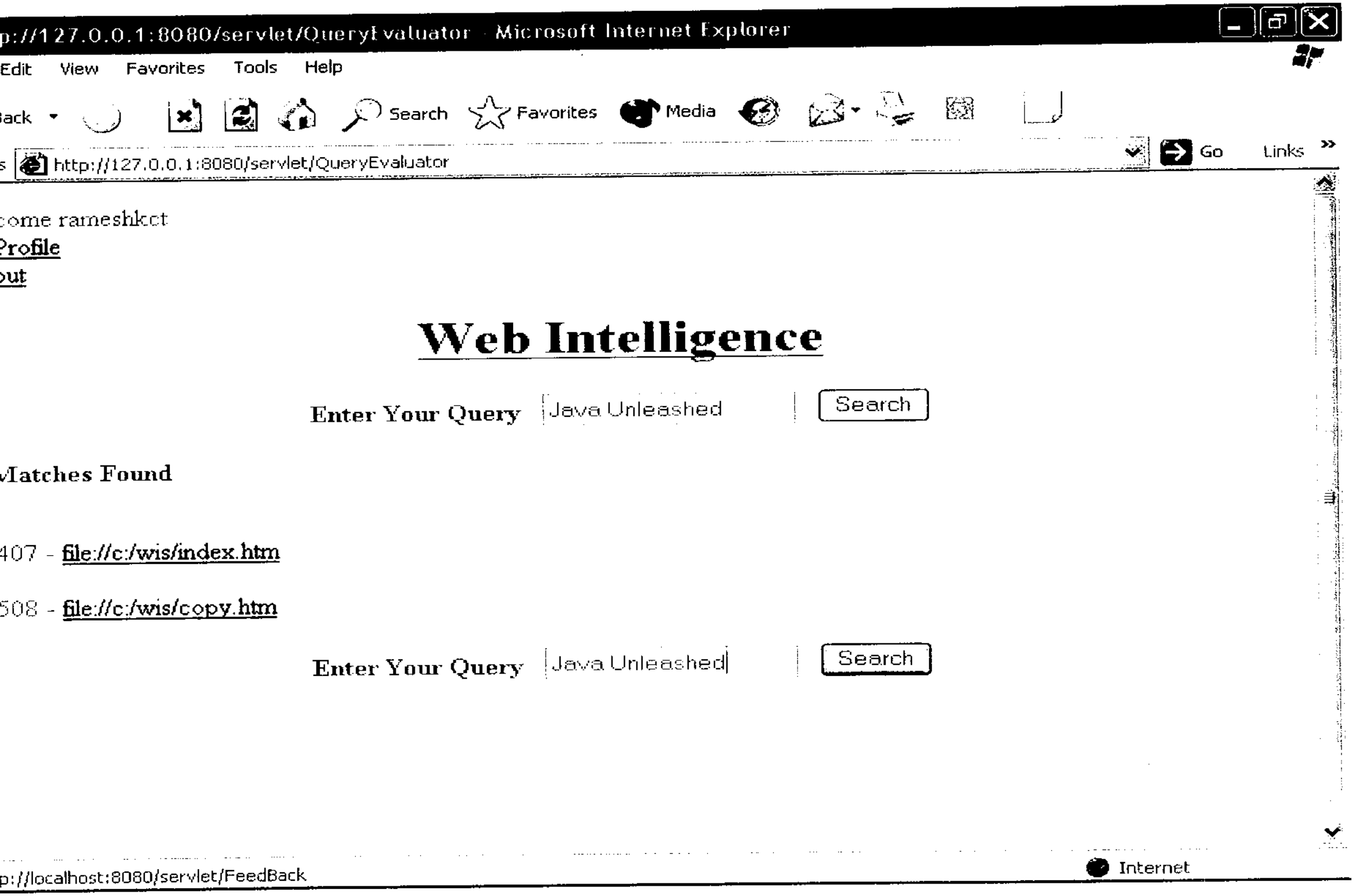

p://localhost:8080/servlet/Login - Microsoft Internet Explorer
Edit View Favorites Tools Help
Back
Search
Favorites
Media
s http://localhost:8080/servlet/Login
Go
Links

come rameshkct
Profile
out

Web Intelligence

Enter Your Query Java Unleashed

Fig 10.2 Search Page


p://127.0.0.1:8080/servlet/QueryEvaluator - Microsoft Internet Explorer
Edit
View
Favorites
Tools
Help
ack
Search
Favorites
Media
s
http://127.0.0.1:8080/servlet/QueryEvaluator
Go
Links
Search
Search
Internet

ome rameshkct
Profile
out

Web Intelligence

Enter Your Query Java Unleashed

Matches Found

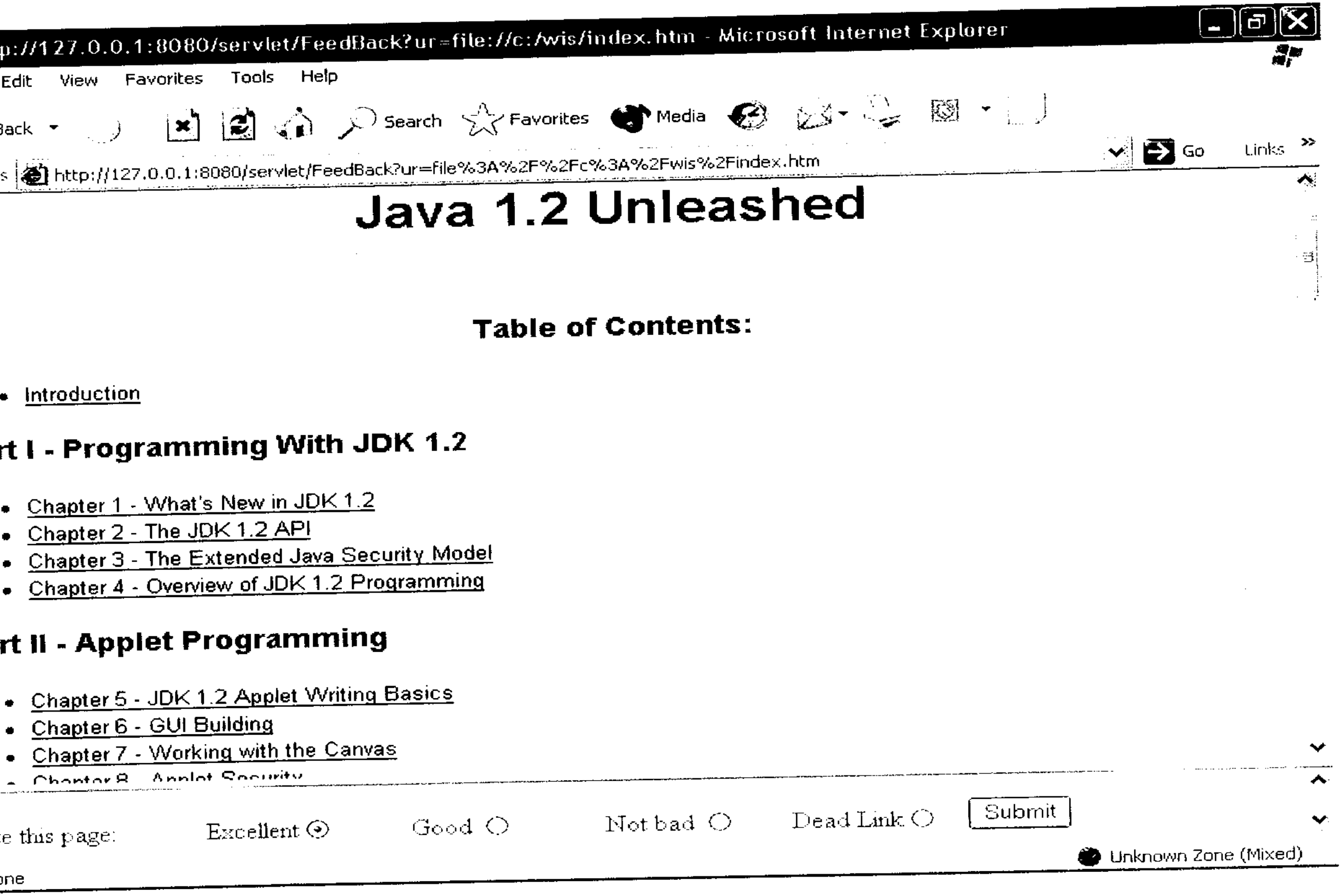
407 - file://c:/wis/index.htm

508 - file://c:/wis/copy.htm

Enter Your Query Java Unleashed

p://localhost:8080/servlet/FeedBack

Fig 10.3 Result Page


p://127.0.0.1:8080/servlet/FeedBack?ur=file://c:/wis/index.htm - Microsoft Internet Explorer
Edit View Favorites Tools Help
Back Search Favorites Media
s http://127.0.0.1:8080/servlet/FeedBack?ur=file%3A%2F%2Fc%3A%2Fwis%2Findex.htm
Go Links
Submit
one
Unknown Zone (Mixed)

Java 1.2 Unleashed

Table of Contents:

- Introduction

rt I - Programming With JDK 1.2

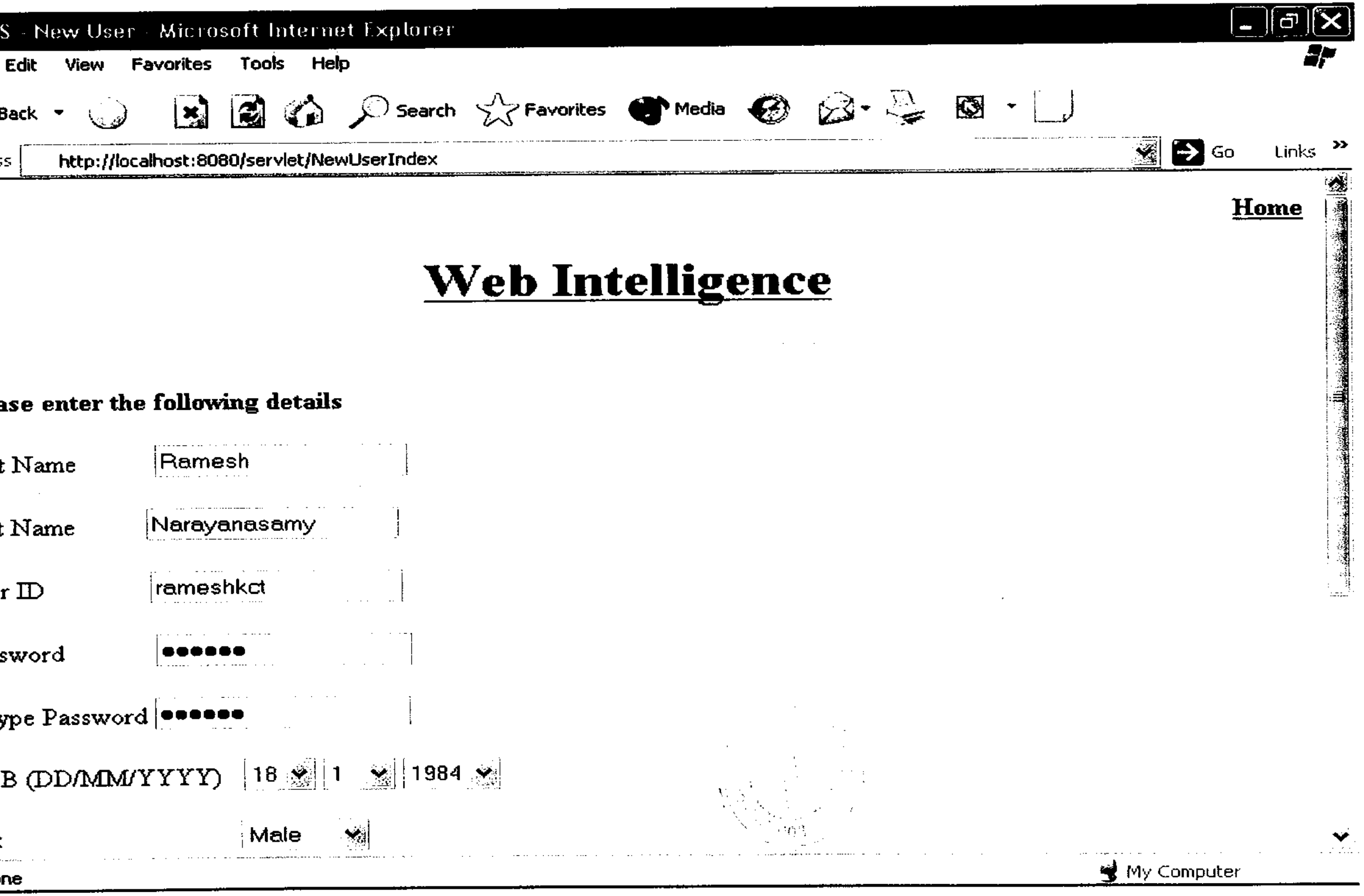
- Chapter 1 - What's New in JDK 1.2
- Chapter 2 - The JDK 1.2 API
- Chapter 3 - The Extended Java Security Model
- Chapter 4 - Overview of JDK 1.2 Programming

rt II - Applet Programming

- Chapter 5 - JDK 1.2 Applet Writing Basics
- Chapter 6 - GUI Building
- Chapter 7 - Working with the Canvas
- Chapter 8 - Applet Security

e this page: [✓ Excellent] Good ○ Not bad ○ Dead Link ○

Fig 10.4 Feedback Page


S - New User - Microsoft Internet Explorer

http://localhost:8080/servlet/NewUserIndex

Home

Web Intelligence

ase enter the following details

t Name	Ramesh
t Name	Narayanasamy
r ID	rameshkct
sword	●●●●●●
ype Password	●●●●●●
B (DD/MM/YYYY)	18 1 1984
	Male

Fig 10.5 New user Page