imoogle

An Image Based Search Engine

Software Project Report
By

Gaurav Gupta & Shishir Mittal

B.Tech. CSE II Yr. 06000029 B.Tech. CSE II Yr. 06000002

Under the guidance of

Mr. Rajeev Srivastava & Dr. S.K.Singh

Department of Computer Science and Engineering
Institute of Technology
Banaras Hindu University, Varanasi - 221005

DEPARTMENT OF COMPUTER ENGINEERING
INSTITUTE OF TECHNOLOGY
BANARAS HINDU UNIVERSITY
Varanasi – 221005, INDIA

Dr. S.K.Singh
Reader
Computer Science and Engineering

Tel # +91- 9336512759
email: sks.cse@itbhu.ac.in

Ref. No. IT/CSE/2007-08/ *Dated:* _____

CERTIFICATE

This is to certify that **Mr. Gaurav Gupta, Roll Number : 06000029 & Mr. Shishir Mittal, Roll Number : 06000002 ,** student of Department of Computer Science and Engineering, Institute of Technology, Banaras Hindu University, Varanasi have been working for their Software Project entitled " **Imoogle-An Image Based Search Engine** " under my supervision from beginning of IV Semester of Computer Science and Engineering B.Tech. Program 2006-2010. The report submitted by them embodies the literature from various reputed resources and from the material provided by me during the period.

Dr. S.K.Singh
Dated:
Supervisor
Department of Computer Science and Engineering
IT-BHU

DEPARTMENT OF COMPUTER ENGINEERING
INSTITUTE OF TECHNOLOGY
BANARAS HINDU UNIVERSITY
Varanasi – 221005, INDIA

Mr. Rajeev Srivastava
Reader
Computer Science and Engineering

Tel # +91-9305483829
email: rajeev.cse@itbhu.ac.in

Ref. No. IT/CSE/2007-08/

Dated: _____

CERTIFICATE

This is to certify that **Mr. Gaurav Gupta, Roll Number : 06000029 & Mr. Shishir Mittal, Roll Number : 06000002 ,** student of Department of Computer Science and Engineering, Institute of Technology, Banaras Hindu University, Varanasi have been working for their Software Project entitled " **Imoogle-An Image Based Search Engine** " under my supervision from beginning of IV Semester of Computer Science and Engineering B.Tech. Program 2006-2010. The report submitted by them embodies the literature from various reputed resources and from the material provided by me during the period.

Mr. Rajeev Srivastava
Dated:
Supervisor
Department of Computer Science and Engineering
IT-BHU

ACKNOWLEDGEMENT

It has indeed been a great privilege for me to have **Dr. S.K.Singh & Mr. Rajeev Srivastava ,** faculty of Department of Computer Science and Engineering, Institute of Technology, Banaras Hindu University, as our mentor for this project. Their guidance and constant encouragement are the motive force behind this project work. We take this opportunity to express our utmost gratitude to them. We are indebted to them for their timely and valuable advice.

We are highly grateful to Prof. **A.K.Tripathi**, Head, Department of Computer Science and Engineering, Institute of Technology, Banaras Hindu University for providing necessary facilities and encouraging us during the course of work.

We are thankful to all technical and non-teaching staff of the Department of Computer Science and Engineering for their constant assistance and co-operation.

Gaurav Gupta
06000029
B.Tech. II Year

Shishir Mittal
06000002
B.Tech. II Year

CONTENTS

ABSTRACT

A large number of search engine exists today on the web which takes input keyword as a query and display all related links i.e. They performs text based content retrieval , but our aim is bit different viz. Developing a search engine which shall take images as input and reveal all links of web pages where exact image can be found. Project comprises of concepts of Web-Mining , Database-Management and Image-Processing. Web-Mining concept was applied to retrieve images from web. The major concept used was comparison of query image with a large database. This comparison algorithm was developed by us which will distinguish an image within a huge(10^502) database of image with very high efficiency. Besides,clustering of images has been done according to their types and then tables are indexed according to their PiVS in order to reduce time taken in search an image. So this project presents a new way of information retrieval through images and a new way of comparing an image in a large database.

Keywords : PiVS(pixel vector string), Color histogram, Rank, Intensity, Quantization, Crawler, Joint Histograms.

1.INTRODUCTION

This project completely based upon **image based content retrieval**.'Image based' means search will analyze the query image and stored images. This analysis may be on the basis of various image-processing techniques like colors,shape,texture etc. And after 'Image based' analysis resultant informations are displayed.

Complete details related with project is given in following chapters.

Chapter 2 discusses about the background of the image search engines. The next discusses different components of a search engine. Chapter 4 reviews the working of a Crawler in general and image crawler in particular. Next we discuss about the different image processing techniques used.

Chapter 6 summarizes the research paper we implemented. The major part of the project is discussed in th analysis and design project. This is the core of the project.

Implementation details are discussed next and the experiments and results obtained are shown their after. Chapter shows some snap shots of various parts of the project.

Chapter 11 discusses conclusion and scope of future work is discussed next.

References and Source code are given their after.

2. Background

Actually it is very tough to say about previous work that has been done on 'Image Based Search Engine' because very less work has been done in this field. Some 2-3 such search engine exists on web with very low efficiency and very low applicability range. The works are going in the various part of this project like on comparison,crawling etc. Many image comparing algorithms have been proposed that use certain image properties such as height, width, histogram intensity, etc. to detect the similarity between images.
Here, we have proposed a self-developed algorithm challenging the previous ones and proving it to be a more efficient and effective one. Looking at the other end,many crawling algorithms exists to retrieve information from web,and our project involves an efficient image crawler(extension to crawler).

3. Componets of Search Engine

3.1 Preprocessing

It is the phase which should be completed before the query image has been given. Main aim of this step is to construct a data structure with good search behavior. Then generated data structure is used to process the series of query without further reordering to reduce the time taken in search during query time.

In search engines preprocessing phase contains retrieval of information from the web and arrange in suitable data structure. In our search engine we are going through following preprocessing phases :

1. Retrieve images from web.
2. Process images to generate PiVS.
3. Save these PiVS and corresponding links in our database.

We will explain all these steps in details as we will move further.

3.2 Query time Processing

It is the phase in which query has been done by user. In search engines it has generally three phase in first phase user formulates the query and enters it into search engine, in second stage,search engine translate it into tokens and finally these tokens are searched in document collection or database. In our search engine when a query has been issued then following steps will occur :

1.Browse a Query image.

2.Generate PiVS of uploaded image.

3.Search PiVS in database.

4.Show the Web links for the matched PiVS.

4. Crawler

A **web crawler** is a program or automated script which browses the World Wide Web in a methodical, automated manner .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.

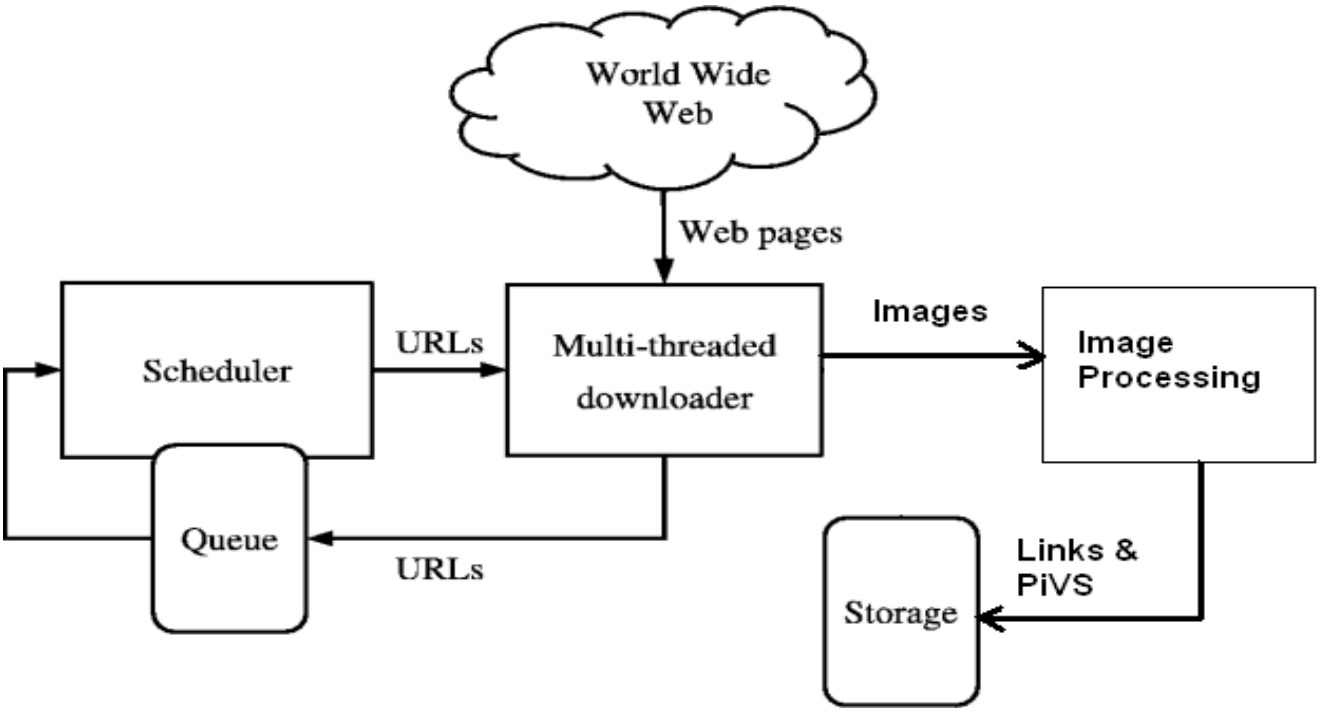


Fig-1 General architecture of a Crawler

In general, it starts with a list of URLs to visit, called the seeds. As the crawler visits these URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs to visit, URLs from the list of links to visit are recursively visited according to a set of policies like selection policy,revisit policy etc. In our project we have to retrieve images and corresponding links from web so we will use an image crawler. Which will download images from web recursively and after passing some image processing operations it will insert information in database.

5.Basic Image Processing Techniques

5.1. Quantization of colors

Color quantization or **color image quantization** is a process that reduces the number of distinct colors used in an image, usually with the intention that the new image should be as visually similar as possible to the original image. Color quantization is critical for displaying images with many colors on devices that can only display a limited number of colors. This quantization may be used as various image comparison algorithms using various color histograms.

There are many quantization algorithms exists to evaluate whole image using color histogram. The most commonly used quantization methods are :

1. Three color color-space(RGB – Red,Green,Blue)

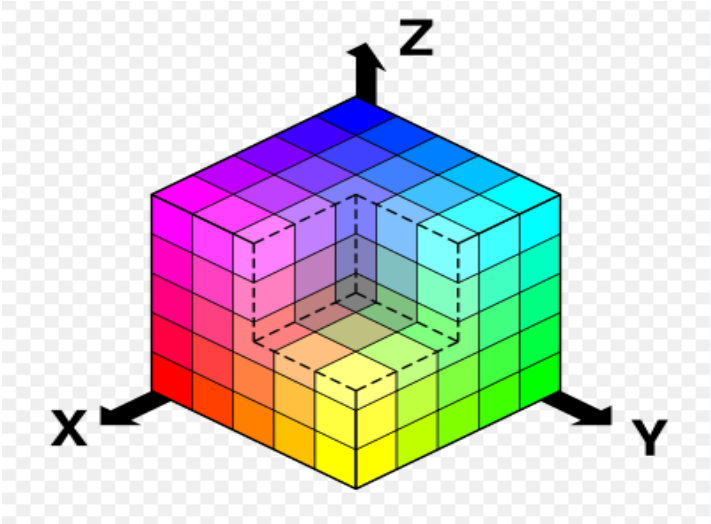


Fig-1 RGB color space

2. Four color color-space(YMCK – Yellow,Magenta,Cyan,Key)

   where key is black.

5.2. Intensity

In RGB color composition three colored light beams (one red, one green, and one blue) must be superimposed . Each of the three beams is called a *component* of that color, and each of them can have an arbitrary intensity, from fully off to fully on, in the mixture. More the content of a particular beam less will be the intensity. Zero intensity for each component gives the darkest color and largest intensity of each color gives lightest color,white as a result. When the intensities for all the components are the same, the result is a shade of gray, darker or lighter depending on the intensity. Intensity is widely used in various image processing techniques like edge-detection , shape-extraction etc. Because intensity changes unexpectedly in case of any edge .

5.3. Histogram concept

5.3.1 Color Histograms

A color histogram is a vector where each entry stores the number of pixels of a given color in the image. The colors of the image are mapped into a discrete color-space containing n colors. Since color histograms do not relate spatial information with the pixels of a given color,they are largely invariant to the rotation and translation of objects in the image. They are global properties of images because histogram is calculated as number of pixels having same color in whole image.

Color histograms have proved less successful on databases with tens of thousands of images. Because a color histogram records only color information, images with similar histograms can have dramatically different appearances. So for better accuracy Joint Histograms are another alternatives.

5.3.2 Joint histograms

This approach incorporates additional information into the summary while preserving the robustness of color histograms. We create a joint histogram by selecting a set of local pixel features and constructing a multidimensional histogram. Each entry in a joint histogram contains the number of pixels in the image that are described by a particular combination of feature values. A joint histogram is a k-dimensional vector, such that each entry in the joint histogram contains the number of pixels in an image that are described by a k-tuple of feature values.

6.Paper IMPLEMENTED

Comparing image using joint histogram by Gregpass & Ramin Zabih from Cornell University.

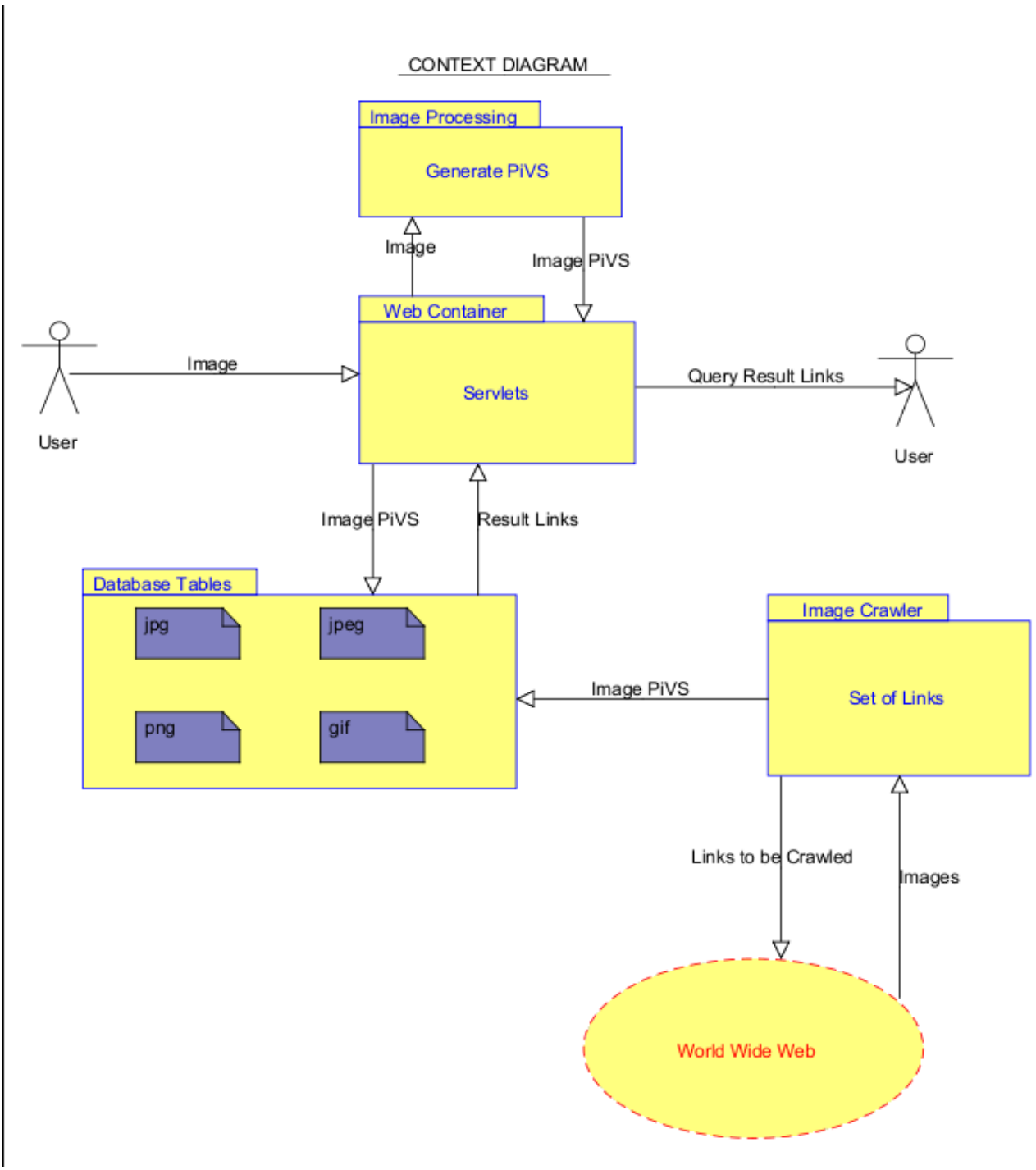
Abstract

Color histograms are widely used for content-based image retrieval due to their efficiency and robustness. However, a color histogram only records an image's overall color composition, so images with very different appearances can have similar color histograms. This problem is especially critical in large image databases, where many images have the same color histogram. In this paper we propose an alternative to color histograms called a joint histogram, which incorporates additional information without sacrificing the robustness of color histograms. We create a joint histogram by selecting a set of local pixel features and constructing a multidimensional histogram. Each entry in a joint histogram contains the number of pixels in the image that are described by a particular combination of feature values. We describe a number of different joint histograms, and evaluate their performance for image retrieval on a database with over 210,000 images. On our benchmarks, joint histograms outperform color histograms by an order of magnitude.

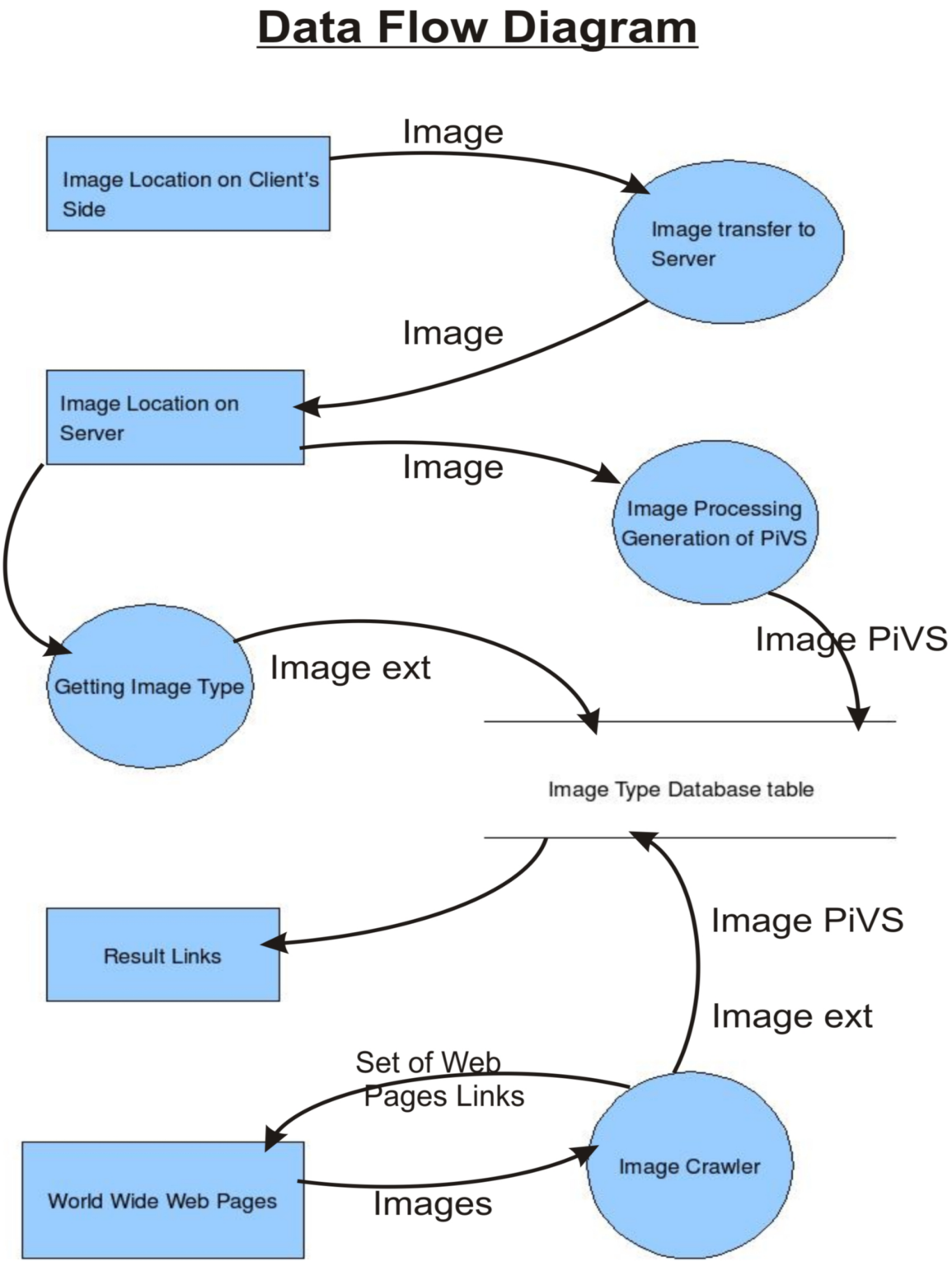
7.ANALYSIS and DESIGN

7.1.Analysis

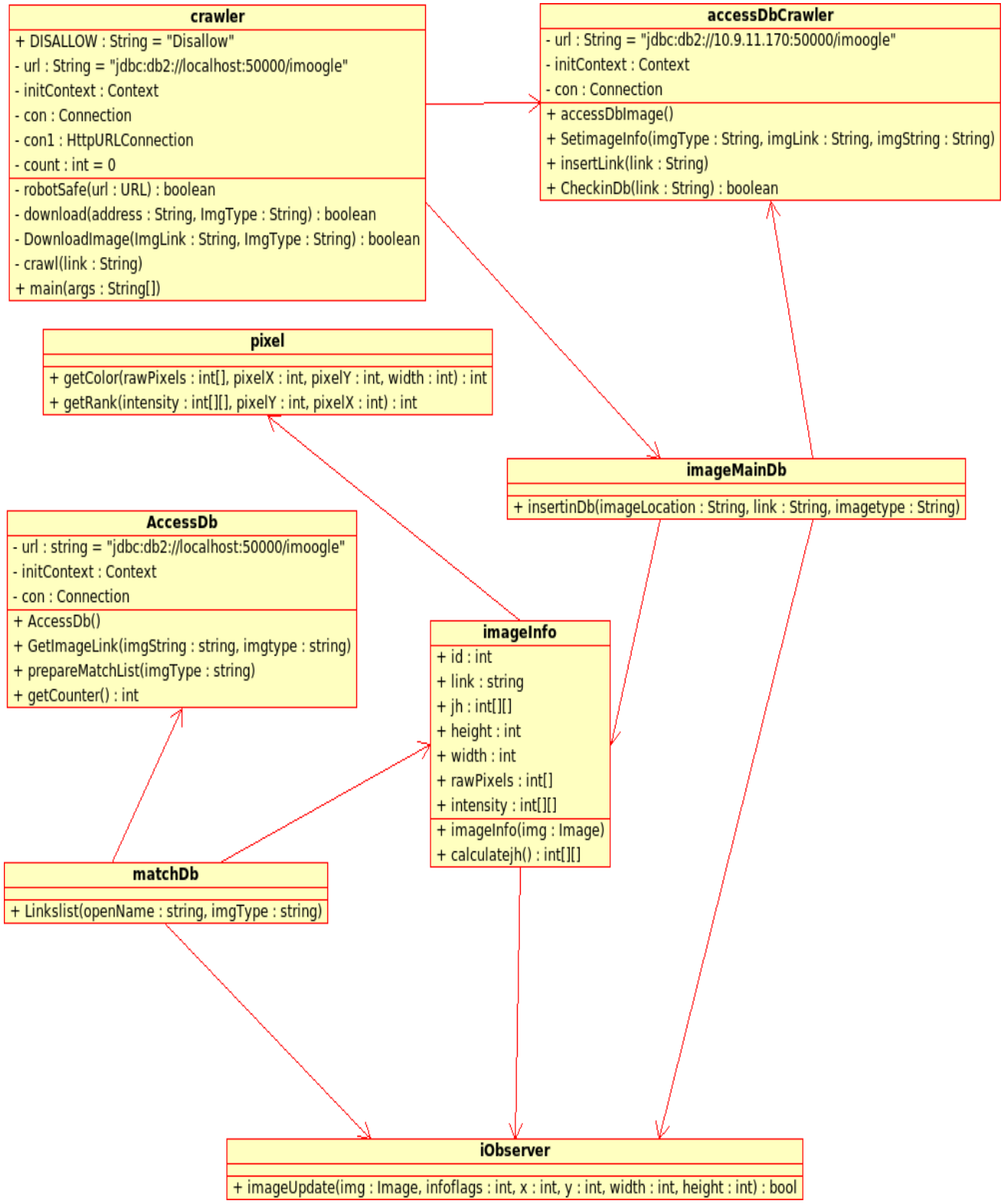
7.1.1.Context Diagram



7.1.2Data Flow Diagram



7.1.3. Class Diagram



7.2.Design

7.2.1. Functions / Methods

Classes

- **accessDb** : Encapsulates all the attributes and methods for interaction with the database. It has following methods:
  - accessDb() : its the constructor which makes a database connection.
  - ArrayList<String> GetImageLink(String imgString, String imgType) : Takes the PiVS and image extension as arguments and returns the links of the PiVSs where it is matched with the passed argument.
  - int getCounter() : returns the number of users who have used the site.

- **imageInfo** : Encapsulates all the attributes and methods regarding the image. It has following methods:
  - imageInfo(Image img) : its the constructor which reads the image and stores pixel values in rawPixels[] matrix.
  - int[][] calculatejh() : calculates the PiVS of the image.

- **iObserver** : keeps track of the image in processing.
  - boolean imageUpdate (Image img, int infoflags,int x, int y, int width, int height) : Updates the imge class when a new image is considered.

- **matchDb** :
  - ArrayList<String> Linkslist(String openName, String imgType): returns all the result links for the image located at openName and image extension imgType.

- **pixel** : Contains method for pixels manipulations.
  - int getColor(int[] rawPixels, int pixelY, int pixelX, int width) : returns the color of the pixel.
  - int getRank(int[][] intensity, int pixelY, int pixelX) : retuns the Rank of a pixel.
- **AccessDbCrawler** : Contains methods which makes the crawler access the database.
  - AccessDbCrawler() : constructor which creates the database connection for the crawler to access the database.
  - **void** SetimageInfo(String imgType, String imgLink, String imgString) : enters the database entry in the imgType database table,
  - **void** insertLink(String link) : makes database entry of links in the visited database table.
  - **boolean** CheckinDb(String link) : checks in database table visited whether the link is visited or not

- **imageMainDb** :
  - **void** insertinDb(String imageLocation, String link, String imagetype) : this is the function called by image crawler which further calls other methods.

- **Crawler** :
  - **boolean** robotSafe(URL url) : to check whether given link is allowed to visit by automated robots.

- **boolean** download(String address, String ImgType) : to download image at given address.

Servlets

- **start** : This is the servlet contained in the web container that takes the request of the image from the index.jsp . Copies the image from the client to the server and then calls mdb.Linkslist(fn, ext) method to get the result links based on the query. It then prints the page to display the result links.

JSPs

- **index.jsp :** consists of a form asking for the image location on th client side. Forwards the request to start servlet on submission of the form.
- **showimage.jsp :** displays the image located on the server based on the request from the start servlet.

7.2.2. Algorithm (Pseudo Code)

I) Creation of Database

```
ImageCrawler( link )
{

if(visited(link))

        return;

InputStream = Downoad(link);

Images = getLinkOfImages(InputStream);

links = getAllLinks(InputStream);


for(all Images)

{

                        Download(image);

                        PiVS=Calculate_PiVS(image);

                        InsertInDatabase(PiVS,link);

}

for(all links)

                        ImageCrawler(links);

}
```

II) Query Image & Calculation of PiVS

```
Input Image

    Intensity[height][width] = CalculateIntensityMatrix(Image).
```

```
getRank(PixelX, PixelY)
        {
        for(i=1 ; i<height ;i++)
                for(j=1 ; j<width; j++)
                {
                        comparison of intensity of each pixel with neighbouring pixels.
                }
        return rank;
    }

     for(i=0 ; i<height ; i++)
        for(j=0 ; j<width ; j++)
        {
                rank = getRank(i,j);

                color = getColor(i,j);

                PiVS[color][rank]++;
        }


   Find the match of PiVS in Database

   if(PiVS exists in Database)

        return (List of links where the image was found);
```

8.IMPLEMENTATION

1. Used standard Database of 60,000+ images along with 40,000 crawled images for implementation of the Image Comparison algorithm.
2. Used JAVA API for reading image and getting the pixel values for different types of images viz. jpeg, jpg, png, gif. Toolkit.getDefaultToolkit() is used for reading the image. PixelGrabber class defined in java.awt.image.* package is used for storing the pixel values of the image in a 1D array of rawpixel[], which is then later converted in a 2D array for easy calculation of Rank.

3. The complete algorithm of image processing is implemented using JAVA API.
   For calculation of intensity and color of a pixel bitwise operators are used.
4. The same API is used for file transfer and image crawler.
   In this class Main function starts with a set of links and checks for its
   - validity
   - Allowness for robots
   - Visited or not visited

   If all above checks are successfully done then crawler opens the URL in an input stream and saves
   it in a string. This String is now scanned for various substrings which are :
   - <img src=
     For searching of addresses of images.
   - <a href=
     For searching of all available links on web page.

   After getting all addresses of image,all images are downloaded on local address and call this crawler function with all links got in previous stages.
5. Used IBM Websphere 2.0 as Web container for servlets and jsps.
6. Used IBM DB2 as database manager.
   - Database has 4 tables viz. gif, jpg, jpeg, png with 2 columns each jh(String type) and link(String type).
   - Database tables has been index on jh. So that we can get quicker results of the queries based on the PiVS.

9.EXPERIMENTS and RESULTS

Implemented algorithm on a Standard Image Database used for similar image based content retrieval by others of **51,000** images.

Server Specifications Used to conduct Experiments

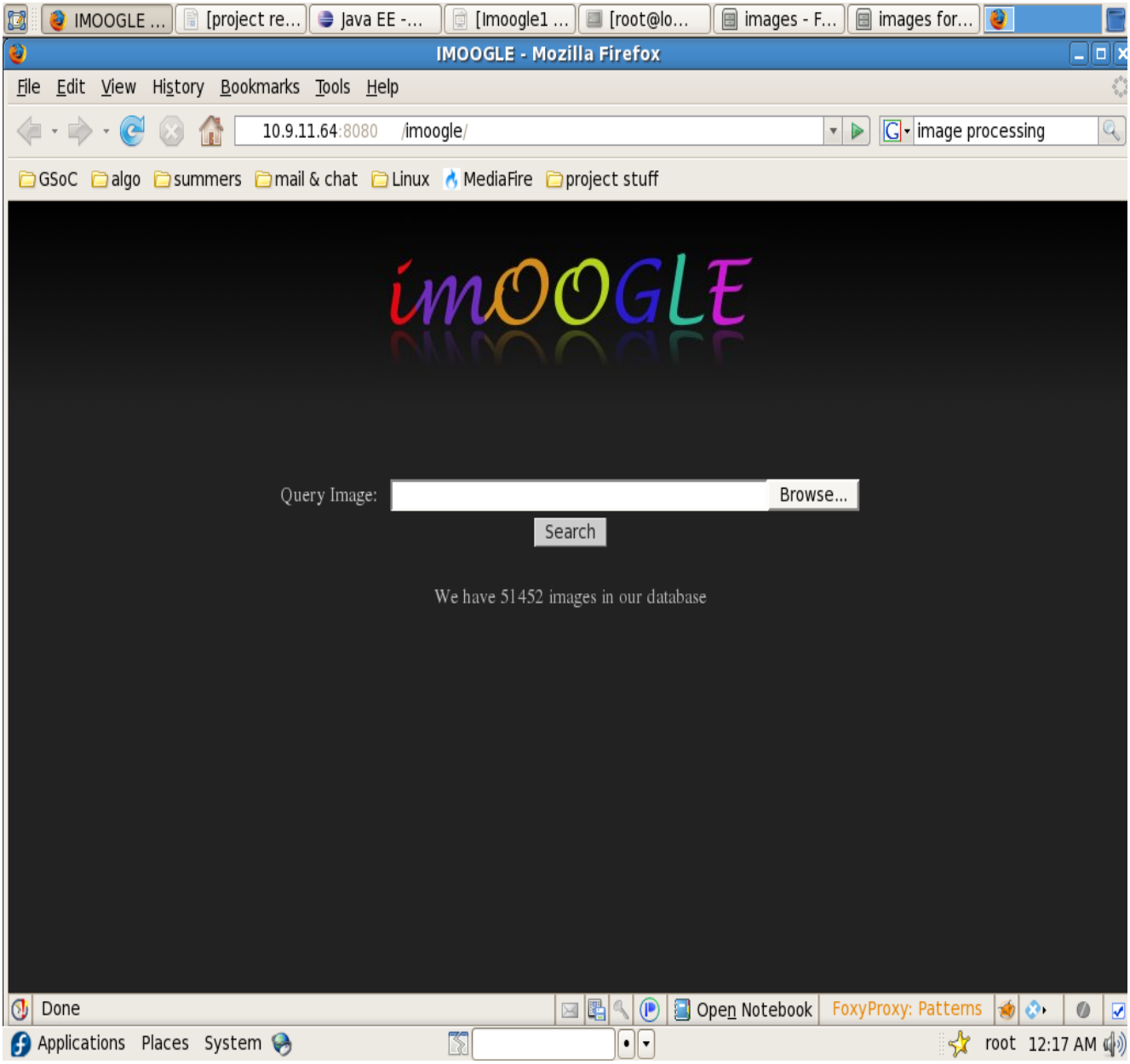
- 1.8 GHz Core 2 Duo Processor
- 1 GB RAM

Size of Image (in kb)	Time for generation of PiVS (sec)	Time for querying in the database (sec)	Total time to display result Links
1	0.006	0.025	0.03
10	0.015	0.025	0.04
100	0.08	0.023	0.10
1000	0.85	0.025	0.88

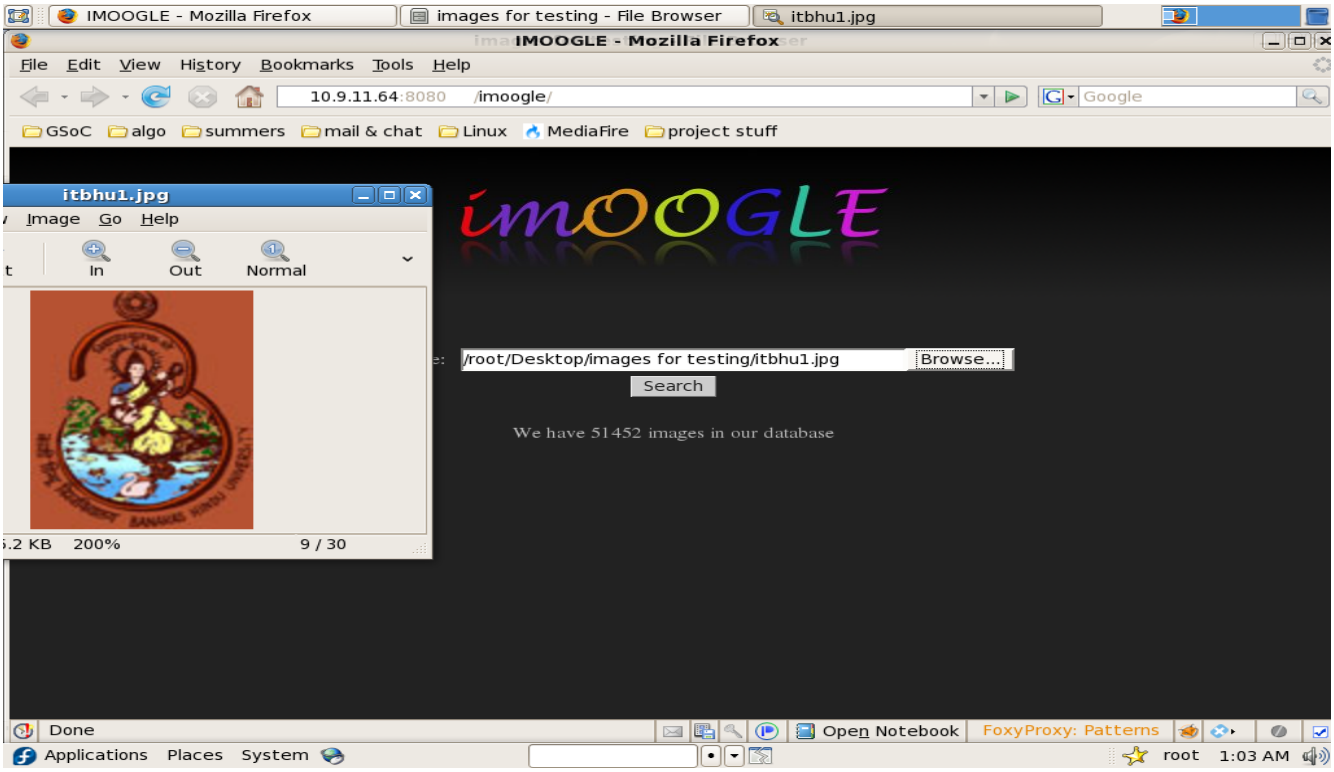
The algorithm uniquely determined the images with **99.4% efficiency**.
Some Cases where the algorithm failed were :

- When mirror images occurred
- When in perfect Black & white image only orientation varies like following image have same PiVS :

10.Screen Shots Of Input & Output

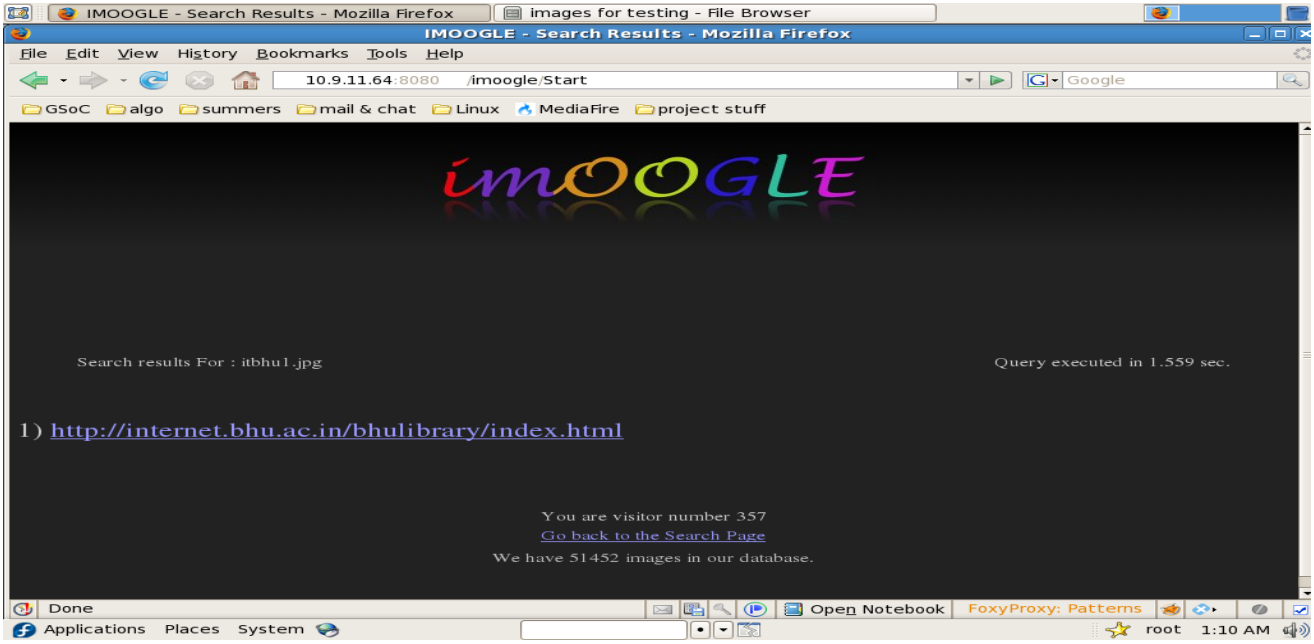


Snap-1 Home page of search engine

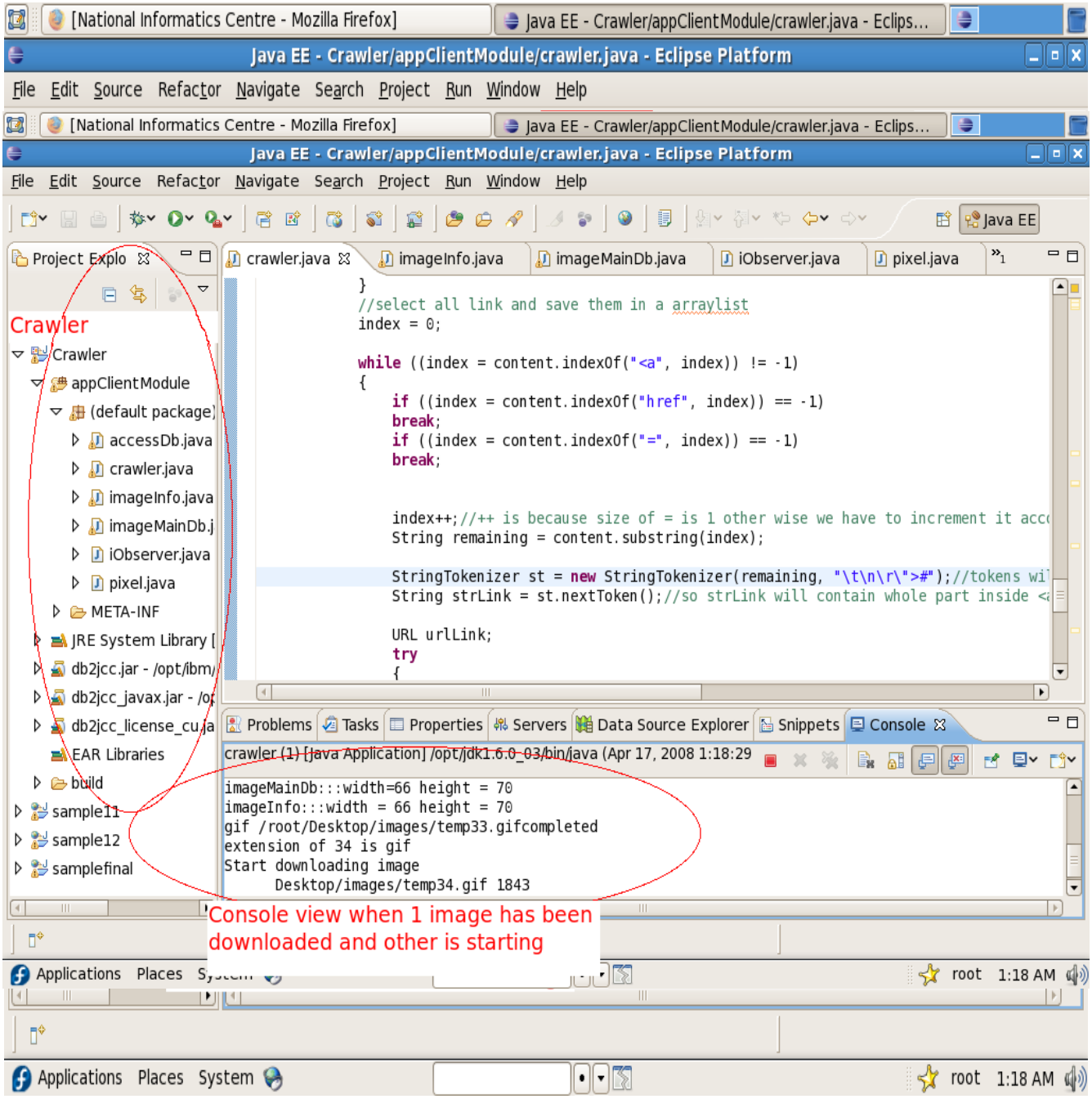


Snap-2 Image Query as shown side wise

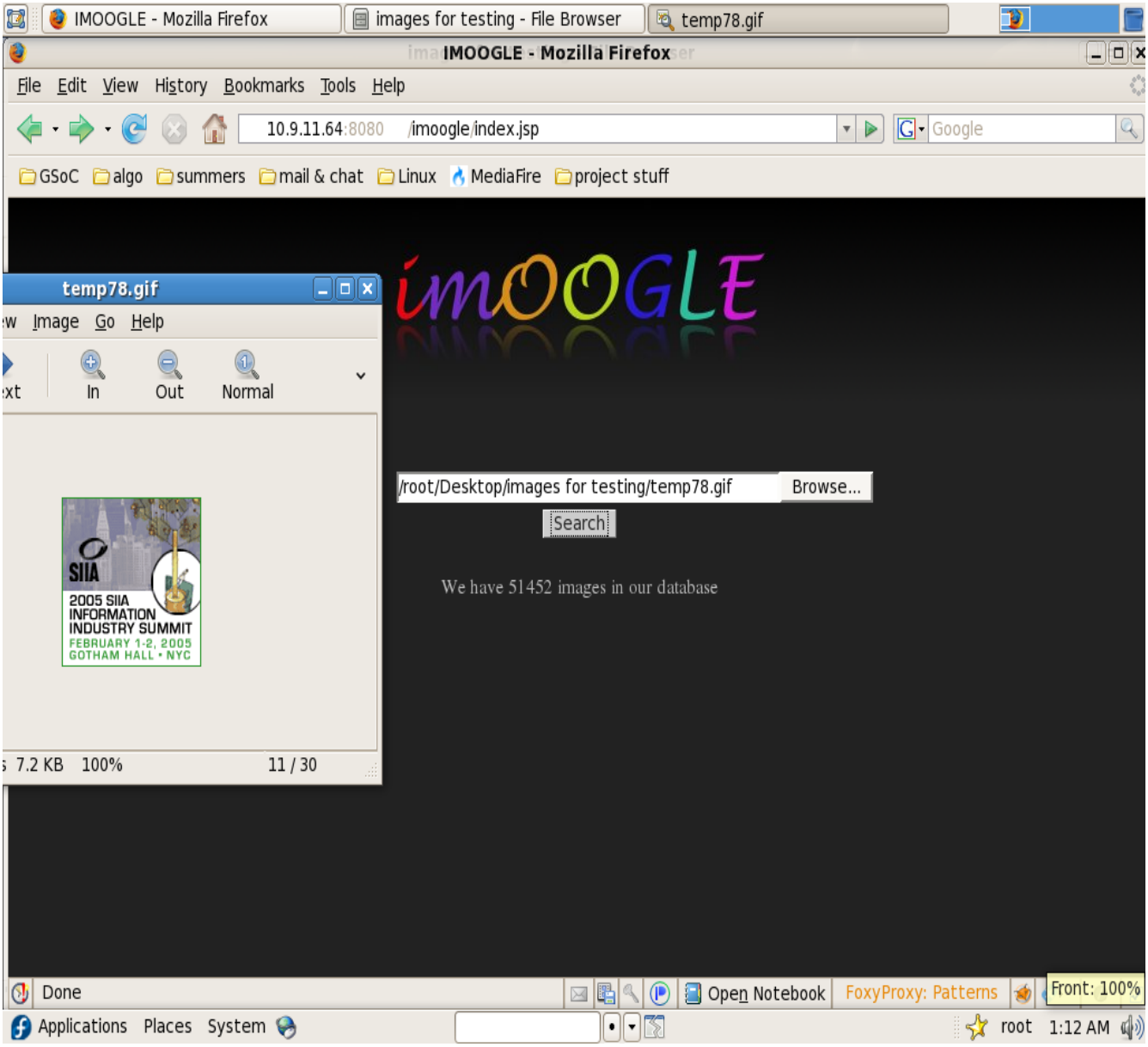
Snap-3 Output links of above query



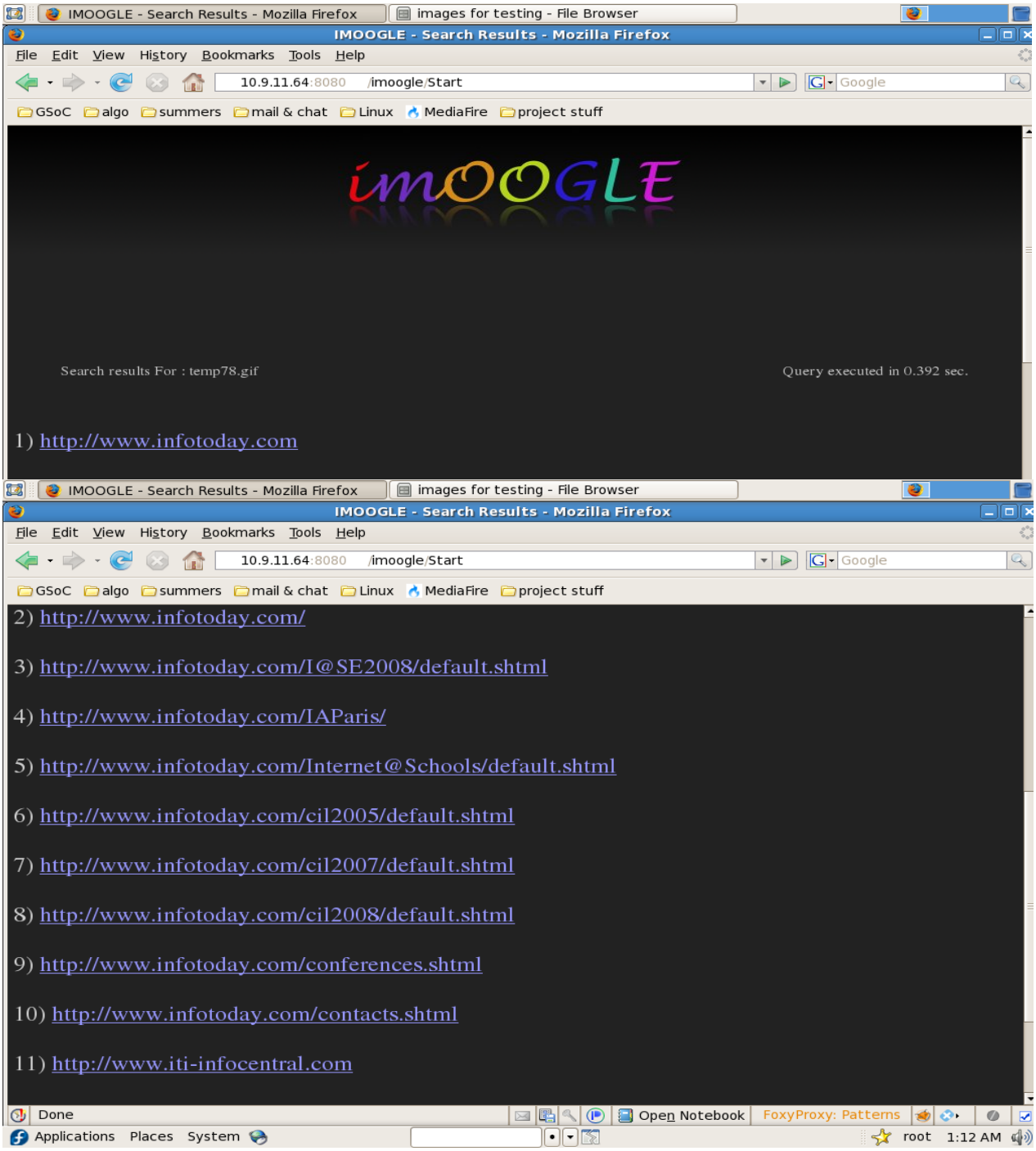
Snap-4 Classes Of Crawler



Snap-5 Crawler While Downloading images



Snap-6 Query image



Snap-7 Output links for above Query

11.CONCLUSION

We had developed a search engine successfully which performs " Image based content retrieval " and results all links where exact same image may be found. It was such a nice experience to do project
"Imoogle" because we learned a lot of new things in various fields like **Web-Crawling** , **Image-Processing** , **J2EE** , **J2SE** , **HTML** , **CSS** , **Java-Script** , **Database-Handling** , **SQL** etc. We had developed an algorithm to compare an image with large database of images which has been working successfully and very efficiently. It may be used in Investigation Purposes & PiVS matching algorithm can be used for any large image database where exact comparison of image is required. So this algorithm and this search engine can be implemented on this huge Web to extract and arrange Web's information.

12.FUTURE WORK

Since this is a useful project for research work and educational purpose, so it has a large scope of modification . Some of the future work that we are going to do are :

1.Creating a **Firefox Plugin** for Image Search which will redirect searches on Imoogle server.
2. Clustering of Images to reduce time complexity .
3. Implementation of **Parallel Processing** for Image Crawler & Generation of PiVS.
4. Maintaining of cache of recent queries.

13.References

Wikipedia
www.wikipedia.org

Color Space Fundamentals
http://dx.sheridan.com/advisor/cmyk_color.html

Color Histograms
http://www.kenrockwell.com/tech/yrgb.htm

Joint Histograms

Paper : Comparing image using joint histogram by Gregpass & Ramin Zabih from Cornell University.

Image Processing

www.ipsi.fraunhofer.de/kueppersfarbe/en/theorie.html

Visual Object Classes

http://www.pascal-network.org/challenges/VOC/

Web Crawler

http://www.beansoftware.com/NET-Tutorials/Web-Crawler.aspx

Image processing Using Java
http://www.javaworld.com/javaworld/jw-09-1998/jw-09-media.html
http://www.developer.com/java/other/article.php/3403921

Java Server Pages
http://java.sun.com/products/jsp/overview.htm

Java Servlet Technology
http://java.sun.com/products/servlet/

SQL
http://www.w3schools.com/sql/default.asp

Standard Image Databse

Pierre-Alain Moëllic, Patrick Hède, Gregory Grefenstette, Christophe Millet “Evaluating Content Based Image Retrieval Techniques with the One Million Images CLIC TestBed” WEC (2) 2005: 171-174

Gregory Grefenstette, Pierre-Alain Moëllic, Patrick Hède, Christophe Millet and Christian Fluhr, "New Testbed of One Million Images", Ercim News, no. 62, July 2005.

Large Image database

http://www.ercim.org/publication/Ercim_News/enw62/grefenstette.html

Eclipse

http://www.eclipse.org/

14.Appendix : Source Code

CRAWLER.java

```
import java.util.*;
import java.io.*;
import java.text.*;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Properties;
import java.util.StringTokenizer;
import java.lang.Object;
import javax.sql.*;
import javax.naming.*;
import java.net.HttpURLConnection;
import java.io.PrintWriter;
import java.sql.*;


public class crawler
{
        public static final String DISALLOW = "Disallow:";
                private String url = "jdbc:db2://localhost:50000/imoogle";
           private Context initContext;
           private Connection con;
           private HttpURLConnection con1;
           private int count =0 ;
         //for check whether robots are allowed or not
        boolean robotSafe(URL url)
          {
System.out.println("Robot Safe Started");
                        String strHost = url.getHost();
System.out.println("link of host "+strHost);
                        // form URL of the robots.txt file
                        String strRobot = "http://" + strHost + "/robots.txt";
System.out.println("link of robots.txt "+strRobot);
                        URL urlRobot;
                        try
                        {
                                urlRobot = new URL(strRobot);
                        } catch (MalformedURLException e)
                        {
System.out.println(" something weird is happening, so don't trust it");
                                                                e.printStackTrace();
```

```
return false;

                        }

                        StringBuffer strCommands= new StringBuffer();
                        try
                        {
                                //open the file robot.text present at URL urlrobot
System.out.println("opening urlRobotStream");
                        HttpURLConnection con = (HttpURLConnection) urlRobot.openConnection();
                                //Page requires authentication
                                //First set the Proxy settings on the System
                                Properties systemSettings = System.getProperties();
                                systemSettings.put("http.proxyHost","10.1.1.19") ;
                                systemSettings.put("http.proxyPort", "80");

                                //prepare the proxy authorization String
                                sun.misc.BASE64Encoder encoder = new sun.misc.BASE64Encoder();
                                String encodedUserPwd =
encoder.encode("282278:iostream".getBytes());
                                //get authorization from the proxy
                                con.setRequestProperty("Proxy-Authorization", "Basic " +
encodedUserPwd);
                                BufferedReader reader = new BufferedReader(
                                        new InputStreamReader(con.getInputStream()));


System.out.println("opened urlRobotStream");
                          // read in entire file
                         /*  byte b[] = new byte[1000];
                          int numRead = StreamReader.read(b);

                          strCommands = new String(b, 0, numRead);
                          while (numRead != -1)
                          {
                                //if (Thread.currentThread() != searchThread)
                                //        break;
                                numRead = StreamReader.read(b);
                                if (numRead != -1)
                                {
                                        String newCommands = new String(b, 0, numRead);
                                        strCommands += newCommands;
                                }
                          }*/
                                String line;

                                while((line=reader.readLine())!=null)
                                {
                                        strCommands.append('\n');
```

```
                                        strCommands.append(line);
                                }

System.out.println("closed urlRobotStream");
                        }catch (IOException e)
                        {
                           // if there is no robots.txt file, it is OK to search
System.out.println("Problem in opening urlRobotStream or no robot.txt exixt on this link");
                                e.printStackTrace();
                                        return true;
                        }

                        // assume that this robots.txt refers to us and
                        // search for "Disallow:" commands.
                        String strURL = url.getFile();
                        int index = 0;
                        //complete file
                        while ((index = strCommands.indexOf(DISALLOW, index)) != -1)//while no
disallow is there it will fail
                        {
                           index += DISALLOW.length();//next time when loop will execute our index
has been updated
                           String strPath = strCommands.substring(index);//this will be the substring of
file after which we will check what are the things which are disallowed
                           StringTokenizer st = new StringTokenizer(strPath);//deviding file in tokens so
we can get after this index what is the first disallow thing

                           if (!st.hasMoreTokens())
                                break;

                           String strBadPath = st.nextToken();//this will be the first thing after index
which is disallowed
System.out.println("Disallow found with "+strBadPath);
                           // if the URL contains a disallowed path in its file name(path+filename=link),
it is not safe
                           if (strURL.indexOf(strBadPath) == 0)
                                return false;
                         }

                                return true;
           }
        boolean download(String address, String ImgType)
        {
                OutputStream out = null;
                URLConnection conn = null;
                InputStream  in = null;
                try {
                        URL url = new URL(address);
                        String localFileName="/home/shishir/Desktop/images/temp"+count+"."+
```

```
ImgType;
                count++;
                        out = new BufferedOutputStream(
                                new FileOutputStream(localFileName));
                        conn = url.openConnection();
                        Properties systemSettings = System.getProperties();
                        systemSettings.put("http.proxyHost","10.1.1.19") ;
                        systemSettings.put("http.proxyPort", "80");

                        //prepare the proxy authorization String
                        sun.misc.BASE64Encoder encoder = new sun.misc.BASE64Encoder();
                        String encodedUserPwd = encoder.encode("282278:iostream".getBytes());
                        //get authorization from the proxy
                        conn.setRequestProperty("Proxy-Authorization", "Basic " + encodedUserPwd);
                        in = conn.getInputStream();
                        byte[] buffer = new byte[1024];
                        int numRead;
                        long numWritten = 0;
                        while ((numRead = in.read(buffer)) != -1) {
                                out.write(buffer, 0, numRead);
                                numWritten += numRead;
                        }
                        System.out.println(localFileName + "\t" + numWritten);


                } catch (Exception exception) {
                        exception.printStackTrace();
                        return false;
                } finally {
                        try {
                                if (in != null) {
                                        in.close();
                                }
                                if (out != null) {
                                        out.close();
                                }
                        } catch (IOException ioe) {
                                return false;
                        }
                }
                System.out.println("image downloaded");
                return true;
        }

        void crawl(String link)
        {

                accessDb access = new accessDb();
                if(access.CheckinDb(link))//when link already exists in our database
```

```
                {
                        System.out.println("Starting Checking in database");
                        return;
                }

System.out.println("url declared");
                URL url;
                try
                {
                         url = new URL(link);
                }catch (MalformedURLException e)
                {
System.out.println(" something weird is happening, so don't trust it");
                                        e.printStackTrace();
                                return ;
                }
System.out.println(url);
                ArrayList list = new ArrayList();//array list which will store links occured in page
                ArrayList ImgList = new ArrayList();//array in which we will store links of all images
avilable on this page
                //checking that protocol should be http
                 if (url.getProtocol().compareTo("http") != 0)
                 {
System.out.println("it is not an http protocol");
                        return;
                 }
                 if(!robotSafe(url))
                 {
System.out.println("Not safe to crawl");
                        return ;
                }
System.out.println("Http protocol and safe to crawl");

                 try
                 {
                         //insert link in database
                        // insert(link);
                         HttpURLConnection con = (HttpURLConnection) url.openConnection();
                          // Skip empty links.
                            //Page requires authentication
                                //First set the Proxy settings on the System

                                Properties systemSettings = System.getProperties();
                                systemSettings.put("http.proxyHost","10.1.1.19") ;
                                systemSettings.put("http.proxyPort", "80");

                                //prepare the proxy authorization String
                                sun.misc.BASE64Encoder encoder = new sun.misc.BASE64Encoder();
                                String encodedUserPwd =
```

```
encoder.encode("282278:iostream".getBytes());
                        //get authorization from the proxy
                        con.setRequestProperty("Proxy-Authorization", "Basic " +
encodedUserPwd);
                        BufferedReader reader = new BufferedReader(
                                new InputStreamReader(con.getInputStream()));

System.out.println("open URL stream");

                    StringBuffer content=new StringBuffer();
                    String newContent;
                    while((newContent=reader.readLine())!=null)
                    {
                        content.append('\n');
                        content.append(newContent);
                    }

System.out.println("close URL stream");

                    //String lowerCaseContent = content.toLowerCase();//complete html page for get
links from page

                    //first select all image and download them and enter all entries in database
                    int index = 0;

                    while ((index = content.indexOf("<img", index)) != -1)
                    {
                       if ((index = content.indexOf("src", index)) == -1)
                            break;
                       if ((index = content.indexOf("=", index)) == -1)
                            break;

                       index++;//++ is because size of = is 1 other wise we have to increment it
according to size
                       String remaining = content.substring(index);

                       StringTokenizer st = new StringTokenizer(remaining, "\t\n\r\">#");//tokens
will delimited by these chars
                       String strLink = st.nextToken();//so strLink will contain whole part inside <a
href= > which will be following link

                       URL urlLink;
                       try
                       {
                            if(strLink.indexOf("'http://")!=-1)
                                    continue;
                            else if(strLink.indexOf("http://")==-1)
                            {
                                    urlLink = new URL(url,strLink);
```

```
                                        strLink = urlLink.toString();
                                    }
                                else
                                {
                                urlLink = new URL(strLink);
                                strLink = urlLink.toString();
                                }
System.out.println("Image Found : " +strLink );
                                ImgList.add(strLink);

                            } catch (MalformedURLException e)
                            {
System.out.println("can't convert url and strlink to urlLink for image");
                                        e.printStackTrace();
                                continue;
                            }
                        }

                        int i;
                        boolean download;



                        for(i=0;i<ImgList.size();i++)
                        {//download all images present on this page
                                //download
                                String ImgLink;
                                ImgLink=ImgList.get(i).toString();//image link will be imagedir / image
source

                                //to extract extension from image link
                                char[] ImgArr = new char[ImgLink.length()];
                                ImgArr=ImgLink.toCharArray();

                                int a=0;
                                for(a=ImgLink.length()-1;a>=0;a--)
                                {
                                        if(ImgArr[a]=='.')
                                                break;
                                }
                                a++;
                                String ext="";
                                for(int t=a;t<ImgLink.length();t++)
                                {
                                        ext+=ImgArr[t];
                                }
System.out.println("extension of "+i+" is "+ext);
                                String localAdd="/home/shishir/Desktop/images/temp"+count+"."+ ext;
                                if(!ext.equals("jpg") && !ext.equals("gif") && !ext.equals("tif") && !
```

```
ext.equals("jpeg") && !ext.equals("png") )
                                        continue;
                                else
                                {
System.out.println("Start downloading image");
                                        //ImgLink="http://"+url.getHost()+ImgLink;
                                        download = download( ImgLink , ext);
                                        //enter in database
                                        if(!download)//if image has not been downloaded
                                        {
System.out.println("image not downloaded");
                                                continue;

                                        }
                                else
                                        {//do enter in database


                                                imageMainDb imagemain = new imageMainDb();

                                                imagemain.insertinDb(localAdd, link, ext);
                                                //File f = new File(localAdd);
                                        //      f.delete();
                                        }
                                //delete image
                                }
                        }
                        //select all link and save them in a arraylist
                        index = 0;

                        while ((index = content.indexOf("<a", index)) != -1)
                        {
                           if ((index = content.indexOf("href", index)) == -1)
                                break;
                           if ((index = content.indexOf("=", index)) == -1)
                                break;


                           index++;//++ is because size of = is 1 other wise we have to increment it
according to size
                           String remaining = content.substring(index);

                           StringTokenizer st = new StringTokenizer(remaining, "\t\n\r\">#");//tokens
will delimited by these chars
                           String strLink = st.nextToken();//so strLink will contain whole part inside <a
href= > which will be following link

                           URL urlLink;
                           try
```

```
                        {
                                // Skip empty links.
                                if (strLink.length() < 1)
                                {
                                 continue;
                                }
                                // Skip links that are just page anchors.
                                if (strLink.charAt(0) == '#')
                                {
                                 continue;
                                }
                                // Skip mailto links.
                                if (strLink.indexOf("mailto:") != -1)
                                {
                                 continue;
                                }
                                // Skip JavaScript links.
                                if (strLink.toLowerCase().indexOf("javascript") != -1)
                                {
                                 continue;
                                }
                              if(strLink.indexOf("http://")==-1)
                                        urlLink = new URL(url,strLink);
                              else if(strLink.indexOf("http://")!=-1 && strLink.charAt(0)=='/')
                                        urlLink = new URL(strLink);
                              else
                                        continue;
                          //    strLink="http://"+url.getHost()+strLink;
                                        strLink=urlLink.toString();
System.out.println("Link found : "+ strLink);
                                        list.add(strLink);
                                } catch (MalformedURLException e)
                                {
System.out.println("can't convert url and strlink to urlLink for link");
                                                e.printStackTrace();
                                        continue;
                                }
                        }
                        //for all link call this function again
                }catch (IOException e)
                {
                        // if there is no  file, it is OK to search
                        //or error in reading urlString
System.out.println("Can't open UrlStream");
                        e.printStackTrace();
                                return;
                }
                accessDb access2 = new accessDb();
                        access2.insertLink(link);
```

```
                int i;
                if(list.size()==0)
                        return;
               for(i=0;i<list.size();i++)
                {
                        System.out.println("crawling "+i+" th link "+list.get(i));
                        crawl(list.get(i).toString());
                }
        }
        public static void main(String args[])
        {
                Scanner in = new Scanner(System.in);

                String link = in.nextLine();
                //object of crawler
                //String link = args[0];
                System.out.println("Starting crawler");
                crawler c = new crawler();
                System.out.println("Declared crawler");
                c.crawl(link);
                System.out.println("End of crawler");

        }
}
```

AccessImageDb.java

```
import java.io.File;
import java.io.IOException;
import java.net.URL;
import java.util.Iterator;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.sql.*;
import javax.naming.*;
import java.io.PrintWriter;
import java.sql.*;
import java.util.*;


public class accessImageDb {
    private String url = "jdbc:db2://10.9.11.170:50000/imoogle";
    private Context initContext;
    private Connection con;
   public accessDb() {
   try {
   initContext = new InitialContext();
    Class.forName("com.ibm.db2.jcc.DB2Driver");
    con = DriverManager.getConnection(url,"db2inst1","gaurav");
     } catch (NamingException ex) {
```

```
            ex.printStackTrace();
          System.out.println("Naming");
    } catch (SQLException ex) {
            ex.printStackTrace();
         System.out.println("SQL");
    } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
         System.out.println("CLASS");
     }
     }
     public void SetimageInfo(String imgType, String imgLink, String imgString)
     {
          try
          {
                    //Connection connection = dataSource.getConnection();
          PreparedStatement addNewImage = con.prepareStatement("insert into "+imgType+"1
values(?, ?)");
          addNewImage.setString(1, imgLink);
          addNewImage.setString(2, imgString);
          addNewImage.executeUpdate();
          con.close();
          }
          catch(SQLException ex)
          {
          System.out.println("Query with image link "+imgLink+" not exec.");
          ex.printStackTrace();
       }
     }
     public void insertLink(String link)
     {
          try
          {
                    PreparedStatement addNewLink = con.prepareStatement("insert into visited values(?)");
                    addNewLink.setString(1, link);
                    addNewLink.executeUpdate();
                    con.close();
          }
          catch(SQLException ex)
          {
                    ex.printStackTrace();
       }
     }
     public boolean CheckinDb(String link)
     {
          try
          {
                    PreparedStatement checkLink = con.prepareStatement("select * from visited where link
= ?");
                    checkLink.setString(1,link);
```

```
                ResultSet resultset = checkLink.executeQuery();
                if(resultset.next())
                {
                        con.close();
                        return true;
                }
                else
                {
                        con.close();
                        return false;
                }
        }
        catch(SQLException ex)
        {
        ex.printStackTrace();
        return true;
    }
  }
}
```

ImageMainDb

```
import java.util.*;
import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.image.*;
import java.awt.MediaTracker.*;

public class imageMainDb
{
        public static void main(String[] args)
  {
    System.out.println("Welcome to the Image Program.");
    System.out.println("");

    // This program is supposed to be run without any arguments.
    // If the user supplies arguments, issue an explanatory message.
    if (args.length!=0)
    {
      System.out.println("Usage: java imageProgram");
      System.exit(1);  // exit.  The exit code 1 indicates failure
    }
    crawlimageDB();
  } // end of main()

        public static void crawlimageDB()
        {
                Image img = (Image)null;
                int i,j,k,m;
                String openName, imageString;
```

```
		String SourceLocation = "/ImageDatabase";
		//String[] imgType = {"gif", "jpg", "jpeg", "png", "tif"};
		String[] imgType = { "riskyjpg" };
		String imgType2 = "jpg";
		int[][] jh = new int[8][9] ;
		for(i=0 ; i<1 ; i++)
		{
			File f = new File(SourceLocation + "/imagedatabase/"+imgType[i]);
			File files[] = f.listFiles();
	System.out.println(files.length);
			try
	{
				//PrintStream output = new PrintStream(new File("/home/shishir/Projects/
imoogle/output"+imgType[i]+".txt"));
				for(j=0 ; j<files.length ; j++)
				{
					if(files[j].isFile())
				   {

						openName = SourceLocation +
"/imagedatabase/"+imgType[i]+"/"+files[j].getName();
						img = Toolkit.getDefaultToolkit().getImage(openName);

						//System.out.println(openName);
						if (img != null)
						{
		// Make sure the entire image is loaded before continuing
							Button b = new Button();  // Create a button to use
as a parameter
				// to the constructor for MediaTracker.
							MediaTracker tracker = new MediaTracker(b);
							tracker.addImage(img,0);
							tracker.waitForID(0);

		// Create "observer", an object that allows us to
		  // use getWidth and getHeight.
							iObserver observer = new iObserver();
							int width = img.getWidth(observer);
							int height = img.getHeight(observer);
							System.out.println("imageMainDb:::width="+width
+" height = "+height);


							if(width==-1 || height==-1)
							{
		// the image has not loaded.
								img = (Image)null;
							}
```

```
                                                    else
                                                    {
                                                            imageInfo image = new imageInfo(img);
                                                            jh = image.calculatejh();
                                                            image.link =
"gsso://"+"imagetype"+imgType2+"imageName"+files[j].getName();

                                                            for(m=0,imageString = "";m<8;m++)
                                                                    for(k=0;k<9;k++)
                                                                            imageString += jh[m][k];
                                                            accessDb Dbobj = new accessDb();
                                                            Dbobj.SetimageInfo(imgType2, image.link,
imageString);

                                                            System.out.println(imgType2 +"
"+openName+"completed");
                                                    }
                    // If the image did not load, print an explanatory
                    // message to the user and ask him/her to try again.
                                                    }
                                                    if (img == null)
                                                    {
                                                            System.out.println("Could not read an image from
file "+openName);
                            //System.out.println("Make sure that you supply the name of an image file, \nand that
you include the gif, jpg or jpeg extension.");
                                                    }  // if (img==null)
                                            }//end of file

                    }//end of for
                    //output.close();
                }//end of try
                                    /*catch(IOException e)
                {
                        System.out.println(e);
                        System.exit(1);

                }*/
                catch(InterruptedException e)
                {
                        e.printStackTrace();
                        System.out.println(e);
                  System.exit(1);
                }
                        }//for
                        //Dbobj.closeConnection();
                }//crawlDb

                public void insertinDb(String imageLocation, String link, String imagetype)
```

```
    {
            Image img = (Image)null;
            int i,j,k,m;
            int[][] jh = new int[8][9] ;
            String imageString;
            //File f = new File(imageLocation);


                    //openName = f.getName();
                    img = Toolkit.getDefaultToolkit().getImage(imageLocation);

                    //System.out.println(openName);
                    if (img != null)
                    {
 // Make sure the entire image is loaded before continuing
                            Button b = new Button();  // Create a button to use as a parameter
                  // to the constructor for MediaTracker.
                            MediaTracker tracker = new MediaTracker(b);
                            tracker.addImage(img,0);
                            try
                            {
                            tracker.waitForID(0);
                            }
                            catch(InterruptedException e)
            {
                    e.printStackTrace();
                    System.out.println(e);
              System.exit(1);
            }


// Create "observer", an object that allows us to
  // use getWidth and getHeight.
                            iObserver observer = new iObserver();
                            int width = img.getWidth(observer);
                            int height = img.getHeight(observer);
                            System.out.println("imageMainDb:::width="+width+" height = "+height);

                            if(width==-1 || height==-1)
                            {
    // the image has not loaded.
                                    img = (Image)null;
                            }
                            else
                            {
                                    imageInfo image = new imageInfo(img);
                                    jh = image.calculatejh();
```

```
                                        for(m=0,imageString = "";m<8;m++)
                                                for(k=0;k<9;k++)
                                                        imageString += jh[m][k];
                                        accessDb Dbobj = new accessDb();
                                        Dbobj.SetimageInfo(imagetype, link, imageString);

                                        System.out.println(imagetype +" "+imageLocation+"completed");
                                }
  // If the image did not load, print an explanatory
  // message to the user and ask him/her to try again.
                        }
                        if (img == null)
                        {
                                System.out.println("Could not read an image from file "+imageLocation);
        //System.out.println("Make sure that you supply the name of an image file, \nand that you
include the gif, jpg or jpeg extension.");
                        }  // if (img==null)

        }
                                                }// imagemain
```

Start.java

```
import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import javax.servlet.RequestDispatcher;
import java.util.Enumeration;

import javax.naming.InitialContext;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.sql.DataSource;
import java.util.ArrayList;


import com.oreilly.servlet.* ;
/**
 * Servlet implementation class for Servlet: Start
 *
 */
 public class Start extends javax.servlet.http.HttpServlet implements javax.servlet.Servlet {
  static final long serialVersionUID = 1L;
```

```
   /* (non-Java-doc)
	 * @see javax.servlet.http.HttpServlet#HttpServlet()
	 */
	public Start() {

		super();
		System.out.println("constr");
	}

	/* (non-Java-doc)
	 * @see javax.servlet.http.HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		// TODO Auto-generated method stub



	}

	/* (non-Java-doc)
	 * @see javax.servlet.http.HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		// TODO Auto-generated method stub

		response.setContentType("text/html");
		PrintWriter pw = response.getWriter();
		ServletContext ctx=this.getServletContext();

		//String file=request.getParameter("FileName");

		String dir=ctx.getRealPath(this.toString());
		char[] dirarr = new char[dir.length()];
		dirarr=dir.toCharArray();
		int k;
		for(k=dir.length()-1;k>=0;k--)
		{
			if(dirarr[k]=='/')
				break;
			/*if(dirarr[k]=='\\')
				break;*/
		}

		String dirName = "";
		for(int o=0;o<=k;o++)
```

```
        dirName+=dirarr[o];

dirName+="Buffer";
//pw.println(dirName);
int maxPostSizeBytes = 10 * 1024 * 1024;
System.out.println(" "+maxPostSizeBytes);
System.out.println(request);
MultipartRequest mr = new MultipartRequest( request, dirName, maxPostSizeBytes );
//
//      log request files/parameters
log( "Dump for request " + mr );

log( "Files:" );
Enumeration   fileNames = mr.getFileNames();
String fn="";


for( int i = 0; fileNames.hasMoreElements(); i++ ) {

        String fileName = ( String )fileNames.nextElement();
        fn=mr.getOriginalFileName(fileName);
        File file = mr.getFile( fileName );
        log( "" + ( i + 1 ) + ". fileName=" + fileName + "\r\n" +
                "fileName=" + fileName + "\r\n" +
                "originalName=" + mr.getOriginalFileName( fileName ) + "\r\n" +
                "contentType=" + mr.getContentType( fileName ) + "\r\n" +
                "filesystemName=" + mr.getFilesystemName( fileName ) + "\r\n" +
                "localPath=" + file.getAbsolutePath() + "\r\n" +
                "length=" + file.length() + "\r\n" +
                "" );
}
log( "Parameters:" );
Enumeration parameterNames = mr.getParameterNames();
for( int i = 0; parameterNames.hasMoreElements(); i++ ) {
        String parameterName = ( String )parameterNames.nextElement();
        String parameterValue = mr.getParameter( parameterName );
        log( "" + ( i + 1 ) + ". " + parameterName + "=" + parameterValue );
}

char[] fnarr = new char[fn.length()];
fnarr=fn.toCharArray();
int a=0;
for(a=fn.length()-1;a>=0;a--)
{
        if(fnarr[a]=='.')
                break;
}
a++;
String ext="";
```

```
		for(int b=a;b<fn.length();b++)
		{
			ext+=fnarr[b];
		}
		for(a=fn.length()-1;a>=0;a--)
		{
			if(fnarr[a]=='/')
				break;
		}
		a++;
		String name="";
		for(int b=a;b<fn.length();b++)
		{
			name+=fnarr[b];
		}

		//RENAME
		//Obtain the reference of the existing file
				File oldFile1 = new File(dirName+"/"+fn);
		accessDb db = new accessDb();
		int i=db.getCounter();

		fn=dirName+"/"+i+"."+ext;
		//Now invoke the renameTo() method on the reference, oldFile in this case
			oldFile1.renameTo(new File(fn));
			long initime = System.currentTimeMillis();


		ArrayList<String> resultLinks = new ArrayList<String>();

		matchDb mdb = new matchDb();
		resultLinks = mdb.Linkslist(fn, ext );
		//consider non matching and non exicting type of errors
		long endtime = System.currentTimeMillis();
		double exetime = (endtime - initime)/1000.0;
		pw.println("<html> <head> <meta http-equiv='Content-Type' content='text/html;
charset=UTF-8' /><title>IMOOGLE - Search Results</title></head><body style='margin:
0;background-color:#222222;background-image: url(images/bg.gif);background-repeat: repeat-
x;'><div id='header' style='height:200px;width:100%;	text-align:center;margin:0;padding:
0;'><center><div id='splash' style='height: 350px;width: 350px;background-image:
url(images/imoogle.jpg);background-repeat: no-repeat;'></div></center> </div><br/>");
		pw.println("<table style='width:100%; padding:50px; color:#aaa'><tr> <td>");
		pw.println("Search results For : "+name);
		pw.println("</td><td style='text-align:right'> Query executed ");
		pw.println( " in "+ exetime+" sec. </td> </tr> ");
		if( resultLinks.size()==0 )
		{
			pw.println("<tr><td colspan='2' style='text-align:center'>No matching images
found. Please try a different Search Image.</td></tr>");
```

```
                }
                for(int j=0 ; j<resultLinks.size() ; j++)
                {
                        //RequestDispatcher dispatcher =
request.getRequestDispatcher("ShowImage.jsp?link="+resultLinks.get(j));
                        //dispatcher.forward(request, response);
                        pw.println("<tr><td colspan='2' style='text-align:center'> "+(j+1)+" <a
style='color:#fff' href=\""+resultLinks.get(j)+"\">"+resultLinks.get(j)+"</a></td></tr><br/>");
                }
                pw.println("<tr><td colspan='2' style='text-align:center'>");
                pw.println("You are visitor number "+i+"<br/><a href='index.jsp' style='color:#ccc'>Go
back to the Search Page</a> </tr><tr><td colspan='2' style='text-align:center; color:#bbb'>We have
51452 images in our database.</td></tr></table></body></html>");
                File f = new File(fn);
                f.delete();
        }
        }
```

MatchDb.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author shishir
 */
import java.util.*;
import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.image.*;

public class matchDb
{

  public ArrayList<String> Linkslist(String openName, String imgType )//openName = "Downloaded
Image Location"
  //imgType = "Downloaded Image Type";
  {
     String imageString = null;

     int i=0,j=0;
     int[][] jh = new int[8][9];

     try
     {
           Image img = (Image)null;
           //Check that the file name has a legal extension
```

```
            img = Toolkit.getDefaultToolkit().getImage(openName);
            Button b = new Button();  // Create a button to use as a paraemter
                  // to the constructor for MediaTracker.
            MediaTracker tracker = new MediaTracker(b);
            tracker.addImage(img,0);
            tracker.waitForID(0);
            iObserver observer = new iObserver();
            int width = img.getWidth(observer);
            int height = img.getHeight(observer);
            if(width==-1 || height==-1)
            {
                  // the image has not loaded.
                     img = (Image)null;
                     System.out.println("Could not read an image from file "+openName);
                     System.out.println("upload a valid image!!!");
                     return null;
            }
            else
            {
               imageInfo image = new imageInfo(img);
               jh = image.calculatejh();

               for(i=0 , imageString = ""; i<8; i++)
                  for(j=0;j<9;j++)
                           imageString += jh[i][j];
               System.out.println("ImageString of downloaded image :"+imageString);

               accessDb Dbobj = new accessDb();
               ArrayList<String> resultList = Dbobj.GetImageLink(imageString, imgType);
               return resultList;

            }
         }

         catch(InterruptedException e)
         {
            e.printStackTrace ();
            System.out.println(e);
            System.exit(1);
            return null;
         }

      }

}
```

Imageinfo.java

```
import java.util.*;
import java.io.*;
```

```
import java.util.*;
import java.awt.*;
import java.awt.image.*;
import java.awt.MediaTracker.*;

public class imageInfo
{
        public int id;
        public String link;
        public int[][] jh = new int[8][9];
        public int height;
        public int width;
        public int[] rawPixels;
        public int[][] intensity;

        public imageInfo(Image img)
        {
                int i,j;
                iObserver observer = new iObserver();
      width = img.getWidth(observer);
      height = img.getHeight(observer);
      //System.out.println("width = "+width+" height = "+height);
      for(i=0;i<8;i++)
         for(j=0;j<9;j++)
                jh[i][j] = 0;

      rawPixels = new int [width * height];
      //System.out.println("Inside Constructor imageInfo");
      PixelGrabber pg = new PixelGrabber (img, 0, 0, width, height,rawPixels, 0, width);
      try
      {
          pg.grabPixels ();
      }
      catch (InterruptedException e)
      {
          e.printStackTrace ();
      }

      //Calculation of Intensity matrix;
      intensity = new int [height][width];
      for(i=0 ; i<height ; i++)
         for(j=0 ; j<width ; j++)
         {
                intensity[i][j] = (rawPixels[i*width + j] & 127) + (rawPixels[i*width + j]>>8 & 127) +
                                                (rawPixels[i*width + j]>>16 & 127);
                //System.out.println("intensity["+i+"]["+j+"] = "+ intensity[i][j]);
         }
         }
```

```
	public int[][] calculatejh()
	{
		//System.out.println("Calculating jh \n height = "+height+" width = "+width);
		int i,j,rank,color;

		for(i=1 ; i<height-1 ; i++)
			for(j=1 ; j<width-1 ; j++)
			{
				//System.out.println("i ="+i+" j= "+j);
				color = pixel.getColor(rawPixels,i,j,width);
				//System.out.print(" color ="+color);
				rank = pixel.getRank(intensity,i,j);
				//System.out.print(" rank ="+rank);
				jh[color][rank]++;

			}
		return jh;
	}
}
```

pixel.java.

```
public class pixel
{

	public static int getColor(int[] rawPixels, int pixelY, int pixelX, int width)
	{
		//System.out.println("calculating color");
		return (((rawPixels[pixelY*width + pixelX]>>7)&1) + 2*((rawPixels[pixelY*width +
pixelX]>>15)&1) +
				4*((rawPixels[pixelY*width + pixelX]>>23)&1));
	}

	public static int getRank(int[][] intensity, int pixelY, int pixelX)
	{
		//System.out.println("Cal rank");
		int rankctr, i;
		for(i=0, rankctr = 0; i<9 ; i++)
		{
			if(intensity[pixelY][pixelX] < intensity[pixelY + i/3 -1][pixelX + i%3 -1])
				rankctr++;
		}
		return rankctr;
	}
}
```

iObserver.java

```
import java.awt.image.ImageObserver;
import java.awt.*;

public class iObserver implements ImageObserver {
```

```
    public boolean imageUpdate (Image img, int infoflags,
                    int x, int y, int width, int height) {
        return true;
    }

}
```

AccessDb.java

```
import java.io.File;


import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
//import com.google.gdata.util.common.xml.XmlWriter;

import javax.sql.*;
import javax.naming.*;



import java.io.PrintWriter;
import java.sql.*;
import java.util.*;
import com.ibm.db2.jcc.DB2Driver;

public class accessDb {
    private String url = "jdbc:db2://localhost:50000/imoogle";
    private Context initContext;
    private Connection con;

    public accessDb() {

    try {

    initContext = new InitialContext();
    Class.forName("com.ibm.db2.jcc.DB2Driver");
    con = DriverManager.getConnection(url,"db2inst1","gaurav");

      } catch (NamingException ex) {
          System.out.println("Naming_ERROR");

    } catch (SQLException ex) {
        System.out.println("SQL_ERROR");
```

```
    }
    catch (ClassNotFoundException ex) {

        System.out.println("CLASS_NOT_FOUND");
    }
    }


    public ArrayList<String> GetImageLink(String imgString, String imgType) // update from us:
changed return type from ArrayList to ArrayList<LocationInfo>
    {

        try {


            PreparedStatement query;

            query = con.prepareStatement("select distinct link from "+imgType+"1 where JH = ?");

            query.setString(1, imgString);
            ResultSet resultSet = query.executeQuery();
            ArrayList<String> links = new ArrayList<String>();

            while(resultSet.next())
            {
                    //System.out.println("resultsetgetLInk ");
                    String newLink = null;
                    newLink = resultSet.getString("link");
               links.add(newLink);

            }
            con.close();
            return links;

        }catch(SQLException ex){

            ex.printStackTrace();
            return null;
        }
    }

    public ArrayList<String> prepareMatchList(String imgType)
    {
            try
            {
                    ArrayList<String> resultString = new ArrayList<String>();
                    ArrayList<String> resultLinks = new ArrayList<String>();

                    int inputId, i;
```

```
                PreparedStatement query2 = con.prepareStatement("select * from "+imgType);

                ResultSet resultSet2 = query2.executeQuery();
                while(resultSet2.next())
                {

                        String newString = null;
                String inputString = null;
                        inputString = resultSet2.getString("JH");
                        inputId = resultSet2.getInt("ID");
                        newString = inputId + " ";
                        //System.out.println("Searching for image with imgtype "+ imgType +" image id
"+inputId);
                        accessDb Dbobj = new accessDb();
                        resultLinks = Dbobj.GetImageLink(inputString, imgType);
                        //System.out.println("result links fetched");
                        for(i=0 ; i<resultLinks.size() ; i++)
                                        newString += resultLinks.get(i)+" ";
                        resultString.add(newString);
                }
                con.close();
                return resultString;
        }
        catch(SQLException ex){

        ex.printStackTrace();
        return null;
    }

  }

  public int getCounter()
  {
        try
        {
                PreparedStatement query = con.prepareStatement("select ctr from counter");
                ResultSet resultset = query.executeQuery();
                int ctr = 0;
                while(resultset.next())
                {
                         ctr = resultset.getInt("ctr");

                }
                query = con.prepareStatement("update counter set ctr = ctr +1");
                query.executeUpdate();
                con.close();
                return ctr;

        }
```

```
        catch(SQLException ex){

        ex.printStackTrace();
        return -1;
      }
   }


}
```

Showimage.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
  <html>

  <%
      //String string_reservation_cost = request.getParameter("reservation_cost");
            String linkf = request.getParameter("link");//got via link as setting session variable
            //image.link = "gsso://"+"imagetype"+imgType2+"imageName"+files[j].getName();
            String type="";
            String name="";
            System.out.println("link to be opened :"+linkf);
            String imagelink = "";
            //char[] linkarr = new char[linkf.length()];
            int i;
            for( i=16;i<19;i++)
            {
                  type+=linkf.charAt(i);
            }
            System.out.println(type+"linkflegth ="+linkf.length());
            i+=9;
            while(i<linkf.length())
            {
                  name+=linkf.charAt(i);
                  i++;
            }

            imagelink="ImageDatabase/"+type+"/"+name;
            System.out.println("link to be uploaded " +imagelink);
      %>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Selected Image</title>
</head>
<body>
```

```
<div id="show">

	<form action=Start method=post>
	<%System.out.println("Printing Image"); %>

	<img src = "<%= imagelink %>">
	<% System.out.println(" Image Printed");%>
	</form>
</div>


</body>
</html>
```

Index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
	<head>
	<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
	<title>IMOOGLE</title>

	</head>
<body style = "margin:0;
	background-color: #222222;
	background-image: url(images/bg.gif);
	background-repeat: repeat-x;">
	<div id="header" style = "height:200px;
	width:100%;
	text-align:center;
	margin:0;
	padding:0;">
		<center><div id="splash" style = "height: 350px;
	width: 350px;
	background-image: url(images/imoogle.jpg);
	background-repeat: no-repeat;
		"></div></center>
	</div>
	<form action="Start" enctype="multipart/form-data" method="post" style = "text-align:center;">
		<label style="color:#ccc">Query Image:</label>   <input type="file"
name="keyimage" size="40"/><br/>
		<input type="submit" value="Search" id="submit" style ="margin-top:5px; border: 1px
outset #ccc;background-color: #ccc; color: #222;"/>
	</form><br/>
	<div style='width:100%; color:#bbb; text-align:center'>We have <%=counter%> images in our
database</div>
```

```
</body>
</html>
```