1. **INTRODUCTION**

Content-based image retrieval (**CBIR**), also known as query by image content (QBIC) and content-based visual information retrieval (CBVIR) is the application of computer vision to the image retrieval problem, that is, the problem of searching for digital images in large databases."Content-based" means that the search will analyze the actual contents of the image. The term 'content' in this context might refer colors, shapes, textures, or any other information that can be derived from the image itself. Without the ability to examine image content, searches must rely on metadata such as captions or keywords. Such metadata must be generated by a human and stored alongside each image in the database.

Problems with traditional methods of image indexing [Enser,1995] have led to the rise of interest in techniques for retrieving images on the basis of automatically-derived features such as color, texture and shape – a technology now generally referred to as *Content-Based Image Retrieval* (CBIR). However, the technology still lacks maturity, and is not yet being used on a significant scale. In the absence of hard evidence on the effectiveness of CBIR techniques in practice, opinion is still sharply divided about their usefulness in handling real-life queries in large and diverse image collections. The concepts which are presently used for CBIR system are all under research.

**1.1 Scope**

Content-based image retrieval (CBIR) is currently an active research area in the computer vision community. Unfortunately, there are only few CBIR systems that can handle e-comics. All of the data of e-comics are available as multimedia documents, i.e. documents consisting of different types of data such as text and images. However, little work has been done on content-based image retrieval to specifically handle digital comics. In this project a CBIR system which demonstrates face detection and recognition techniques to allow the retrieval of comic images from queries of comic characters will be presented. As the CBIR system is mainly built on comic characters detection and recognition, the detection and recognition of comic characters will be the main scope.

**1.2 Existing System**

Despite the shortcomings of current CBIR technology, several image retrieval systems are now available as commercial packages, with demonstration versions of many others available on the Web.

**1.2.1 Commercial systems**

IBM’s QBIC system is probably the best-known of all image content retrieval systems. It is available commercially either in standalone form, or as part of other IBM products such as the DB2 Digital Library. It offers retrieval by any combination of color, texture or shape – as well as by text keyword. Image queries can be formulated by selection from a palette, specifying an example query image, or sketching a desired shape on the screen. The system extracts and stores color, shape and texture features from each image added to the database, and uses R\*-tree indexes to improve search efficiency. At search time, the system matches appropriate features from query and stored images, calculates a similarity score between the query and each stored image examined, and displays the most similar images on the screen as thumbnails. The latest version of the system incorporates more efficient indexing techniques, an improved user interface, the ability to search grey-level images, and a video storyboarding facility.

**1.2.2 Experimental systems**

A large number of experimental systems have been developed, mainly by academic institutions, in order to demonstrate the feasibility of new techniques. Many of these are available as demonstration versions on the Web.

The Netra system uses colour texture, shape and spatial location information to provide region-based searching based on local image. An interesting feature is its use of sophisticated image segmentation techniques.***.*** An example of European CBIR technology is the Surf image system from INRIA, France. This has a similar philosophy to the MARS system, using multiple types of image feature which can be combined in different ways, and offering sophisticated relevance feedback facilities.

**1.2.3    Automated System**

In the automated system, we have adopted Principal Component Analysis (PCA) for achieving dimensionality reduction in the retrieval of gray map images by calculating the column wise mean of the image and making that as the index to represent that image in the feature database.

To improve the above method we have taken the mean of the principle diagonal pixels of the image and making this as the index. Here the computational time for calculating the mean is reduced but accuracy to retrieve exact image is reduced particularly when there is a huge set of images in the database. In this method we can store up to 2560000 different images with same resolution and size.

In the histogram analysis for retrieving the gray scale images, we have given the flexibility to change the no of bins while calculating the histogram counts.

In color component analysis for retrieving the similar images we have implemented Euclidean Distance formula to calculate the distance between the query image and the images present in the database. The images that fall within the specified threshold are retrieved. We can change the value of threshold according to our needs.

**1.3 Proposed System**

The main objective of the proposed system to provide an efficient tool for efficient image retrieval from a huge content of image database using features based on Color, Texture , Binary tree structure and Canny edge detection method and retrieve the images to identify the most similar images to the query image. The terms and methods used in both the approaches are explained below.

**1.3.1 Features Used by CBIR System**

Following are the features used by proposed CBIR system

1. **Shape**:

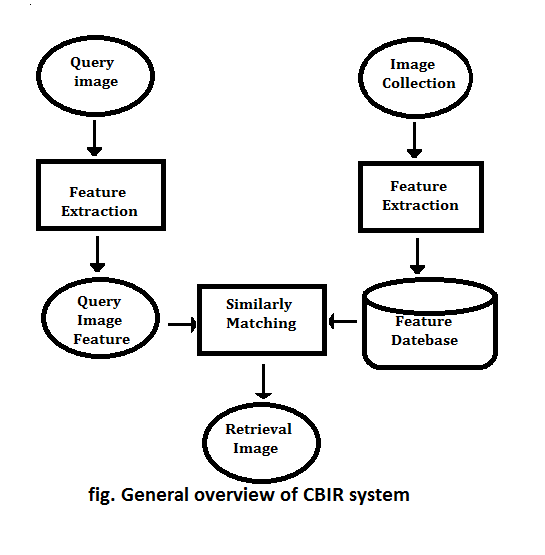
Shape is an important and most powerful feature used for image classification, indexing and retrievals. Shape information extracted using histogram of edge detection. In this paper, the edge information in the image is obtained by using the canny edge detection .Other techniques for shape feature extraction are elementary descriptor, Fourier descriptor, template matching, Quantized descriptors and so on.

1. **Color**

Color may be one of the most straightforward features utilized by humans for visual recognition and discrimination. However, people show the natural ability of using different levels of color specificity in different contexts. For example, people would typically describe an apple as being „red‟, probably implying some type of reddish hue. But in the context of describing the color of a car a person may choose to be more specific instead using the terms „dark red‟ or „maroon‟. Color extraction by computer is performed without benefit of a context. Lack of knowledge also makes it difficult to cull the color information from the color distortion .The appearance of the color of real world objects is generally altered by surface texture, lighting and shading effects, and viewing conditions. The color feature is one of the most widely used visual features in image retrieval. Images characterized by color features have many advantages like robustness, effectiveness, implementation simplicity, computational simplicity, low storage requirement.

1. **Texture**

Texture features are intended to capture the granularity and repetitive patterns of surfaces within a picture. For instance, grassland, brick walls, teddy bears, and flower petals differ in texture, by smoothness as well as patterns. This feature used in domain specific image retrieval, such as in aerial imagery and medical imaging, is particularly vital due to their close relation to the underlying semantics in these cases.



**2. SYSTEM ANALYSIS**

This System Analysis is closely related to [requirements analysis](http://en.wikipedia.org/wiki/Requirement_analysis). It is also "an explicit formal inquiry carried out to help someone (referred to as the decision maker) identify a better course of action and make a better decision than he might otherwise have made."This step involves [breaking down](http://en.wikipedia.org/wiki/Work_breakdown_structure) the system in different pieces to analyze the situation, analyzing project goals, breaking down what needs to be created and attempting to engage users so that definite requirements can be defined..

**2.1 Functional Requirement Specification**

The System after careful analysis has been identified to be present with the following modules.

**1. Feature Extraction Module:**

This module is used to extract features from the images input to it. This module is the largest module having lot of sub modules. First input the image set to this module. Then it extract features of the all the images one by one. Finally input the querying image to this module to extract features of it also. As a result of feature extraction receives a matrix of data. Matrix is a complex structure when calculating the similarity. So an averaging algorithm is used to compress the extracted feature matrix to a one dimensional array for efficiency and

Correctness. Results produce by this module is send to the Clustering and Store Module.

**2. Clustering and Storing Module:**

This module receives inputs from the Feature extraction Module. Extracted features from the Feature extraction Module are clustered and stored by this module. Features are divided into several clusters based on their similarity. Only the features extracted by the image set are stored. This module is very important, because otherwise need to extract features of the image set in each time when the application reopens. As this module stores features in a file, features are stored permanently until user extract features of another image set.

**3. Similarity Calculation Module :**

First this module calculates the nearest cluster for the queried image based on the features of it. Then this module retrieves the stored features of each image in that selected

cluster and calculates the similarity value for each image with the extracted features of the querying image. This outputs a value for each image in the selected cluster with relevance to

the similarity with the query image. Then these calculated similarity values are sorted.

**2.2 Performance Requirements**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely with the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

* The system should be able to interface with the existing system
* The system should be accurate
* The system should be better than the existing system

The existing system is completely dependent on the user to perform all the duties.

**2.3 Software Requirements**:

* Sublime Text 3 or Notepad++ for writing the Front-end Code.
* Forms and Java Applets.
* Xml
* Windows 7/8/10.
* Java programming language.

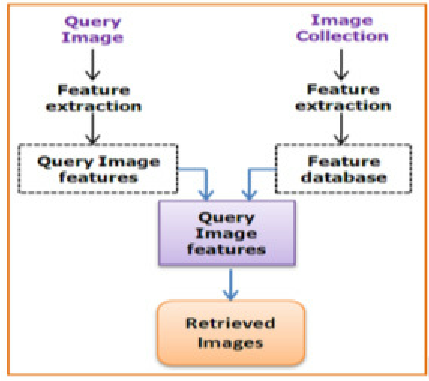
**2.4 Hardware Requirements**:

* System : Pentium IV 2.4 GHz.
* Hard Disk : 40 GB.
* Monitor : 15 VGA Colour.
* Ram : 4 Gb minimum.

**3.SYSTEM DESIGN**

Systems design is the process of defining the architecture, components, modules, interfaces, and [data](http://en.wikipedia.org/wiki/Data) for a [system](http://en.wikipedia.org/wiki/System) to satisfy specified [requirements](http://en.wikipedia.org/wiki/Requirement). One could see it as the application of [systems theory](http://en.wikipedia.org/wiki/Systems_theory) to [product development](http://en.wikipedia.org/wiki/Product_development). [Object-oriented analysis and design](http://en.wikipedia.org/wiki/Object-oriented_analysis_and_design) methods are becoming the most widely used methods for computer systems design.

**3.1 Architectural Design**



**Fig 3.1 Architectural Design**

Figure 3.1 shows a typical architecture of a content-based image retrieval system. Two main functionalities are supported: data insertion and query processing. The data insertion subsystem is responsible for extracting appropriate features from images and storing them into the image database (see dashed modules and arrows). This process is usually performed off-line. The query processing, in turn, is organized as follows: the interface allows a user to specify a query by means of a query pattern and to visualize the retrieved similar images. The query-processing module extracts a feature vector from a query pattern and applies a metric (such as the Euclidean distance) to evaluate the similarity between the query image and the database images. Next, it ranks the database images in a decreasing order of similarity to the query image and forwards the most similar images to the interface module. Note that database images are often indexed according to their feature vectors by using structures such as M-tree [11] or Slim-tree [12] to speed up retrieval and similarity computation. Note that both the data insertion and the query processing functionalities use the feature vector extraction module.

**3.2 Modules**

**1.Feature Extraction Module:**

This module is used to extract features from the images input to it. This module is the largest module having lot of sub modules. First input the image set to this module. Then it

extract features of the all the images one by one. Finally input the querying image to this module to extract features of it also. As a result of feature extraction receives a matrix of data.

Matrix is a complex structure when calculating the similarity. So an averaging algorithm is used to compress the extracted feature matrix to a one dimensional array for efficiency and

correctness. Results produce by this module is send to the Clustering and Store Module.

**2. Clustering and Storing Module**

This module receives inputs from the Feature extraction Module. Extracted features from the Feature extraction Module are clustered and stored by this module. Features are divided in to several clusters based on their similarity. Only the features extracted by the image set are stored. This module is very important, because otherwise need to extract features of the image set in each time when the application reopens. As this module stores features in a file, features are stored permanently until user extract features of another image set.

**3. Similarity Calculation Module**

First this module calculates the nearest cluster for the queried image based on the features of it. Then this module retrieves the stored features of each image in that selected

cluster and calculates the similarity value for each image with the extracted features of the querying image. This outputs a value for each image in the selected cluster with relevance to

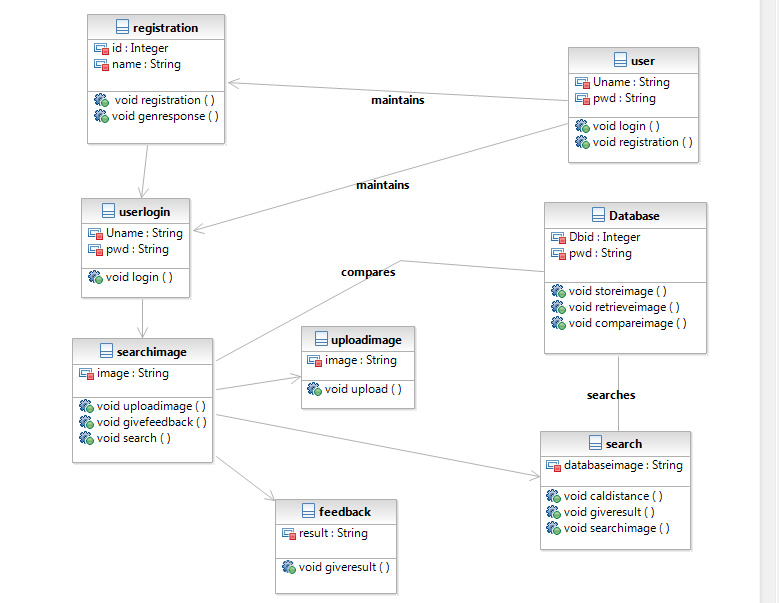
the similarity with the query image. Then these calculated similarity values are sorted.

**3.3 UML Diagrams**

[UML](http://www.omg.org/spec/UML/), short for Unified Modeling Language, is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

UML Diagrams for our application are as follows:

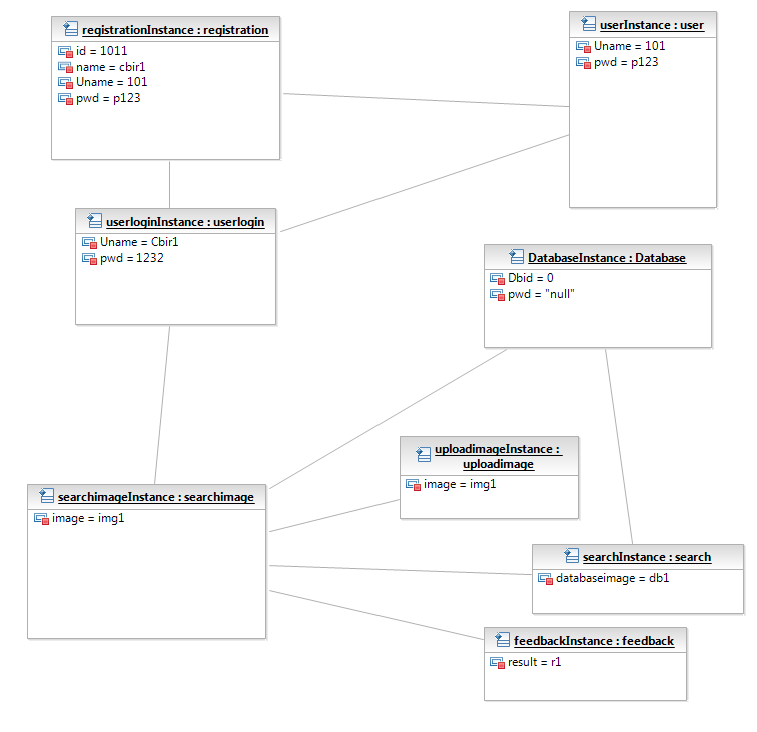
**3.3.1 Class Diagram**



Class Diagram gives us an outlook on how the responsibilities of the system are divided among themselves in terms of class.

**Class diagram** in the Unified Modeling Language (UML) is a type of static structure **diagram** that describes the structure of a system by showing the system's **classes**, their attributes, operations (or methods), and the relationships among objects.

**3.3.2 Object Diagram**



Objects are instances of class where they’re given a certain value to test its application or in some cases explain its functionality and values better.

Here,we have assigned the desired values to the object of the class and created appropriate Instances pertaining to the class, as observed above.

An **object diagram** is a graph of instances, including **objects** and data values. A static **object diagram** is an instance of a class **diagram**,it shows a snapshot of the detailed state of a system at a point in time. The use of **object diagrams** is fairly limited, namely to show examples of data structure.

**3.3.3 Use Case Diagram**

****

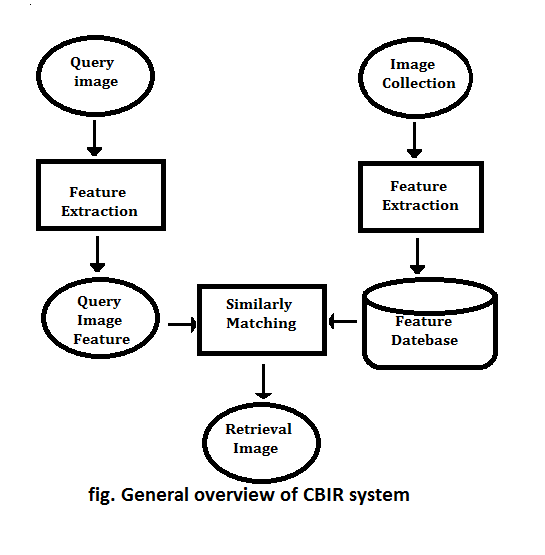
**Use case diagrams** are usually referred to as behavior **diagrams used** to describe a set of actions (**use** cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors).

In our project Actors are

1).User

2).Admin

**4 . Algorithm/System Implementation**

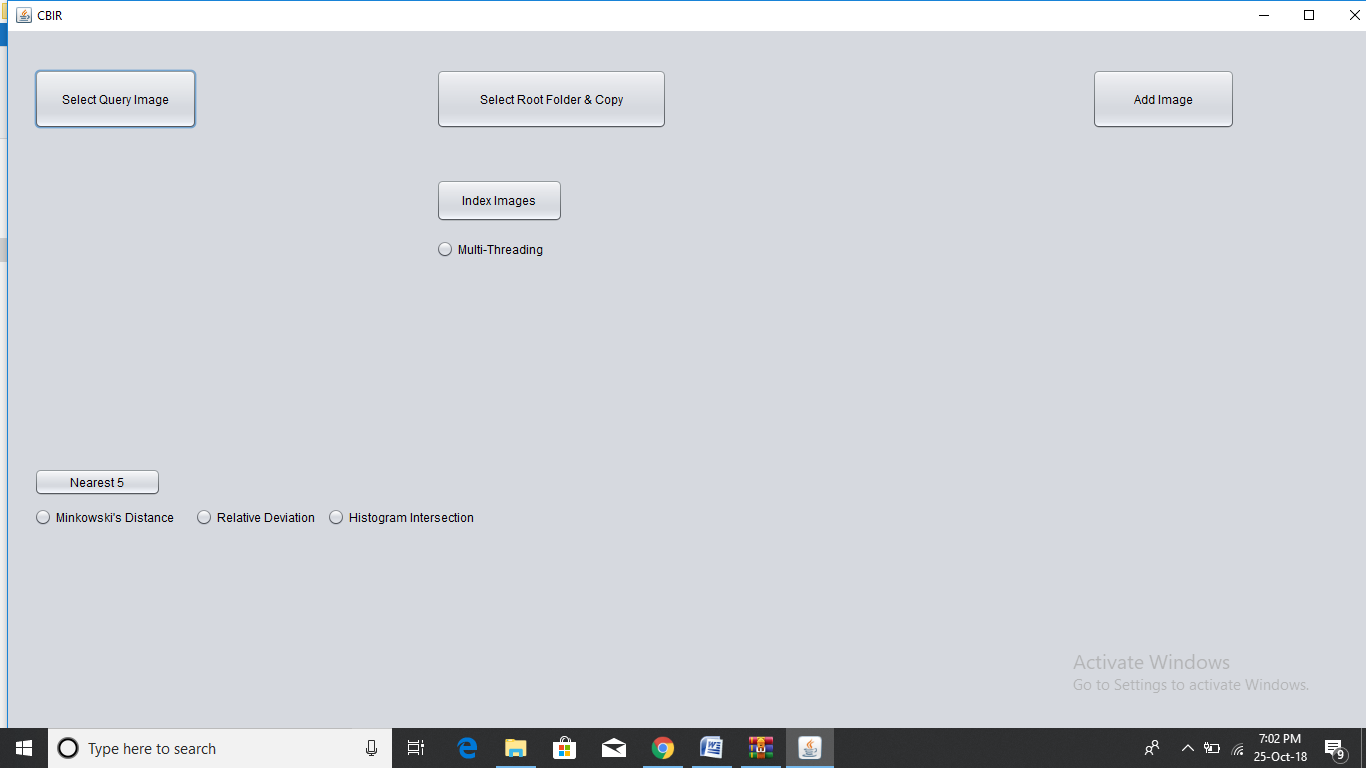
****

**5. OUTPUT SCREENS**

An **output screen** is a device used to display **output**. An **output screen** could be a separate monitor or other display device used only to display the **output** being received from the computer or other device. Output Screens of various functionalities in our project are shown over here along with the description.

**5.1** **Home Page of CBIR:**

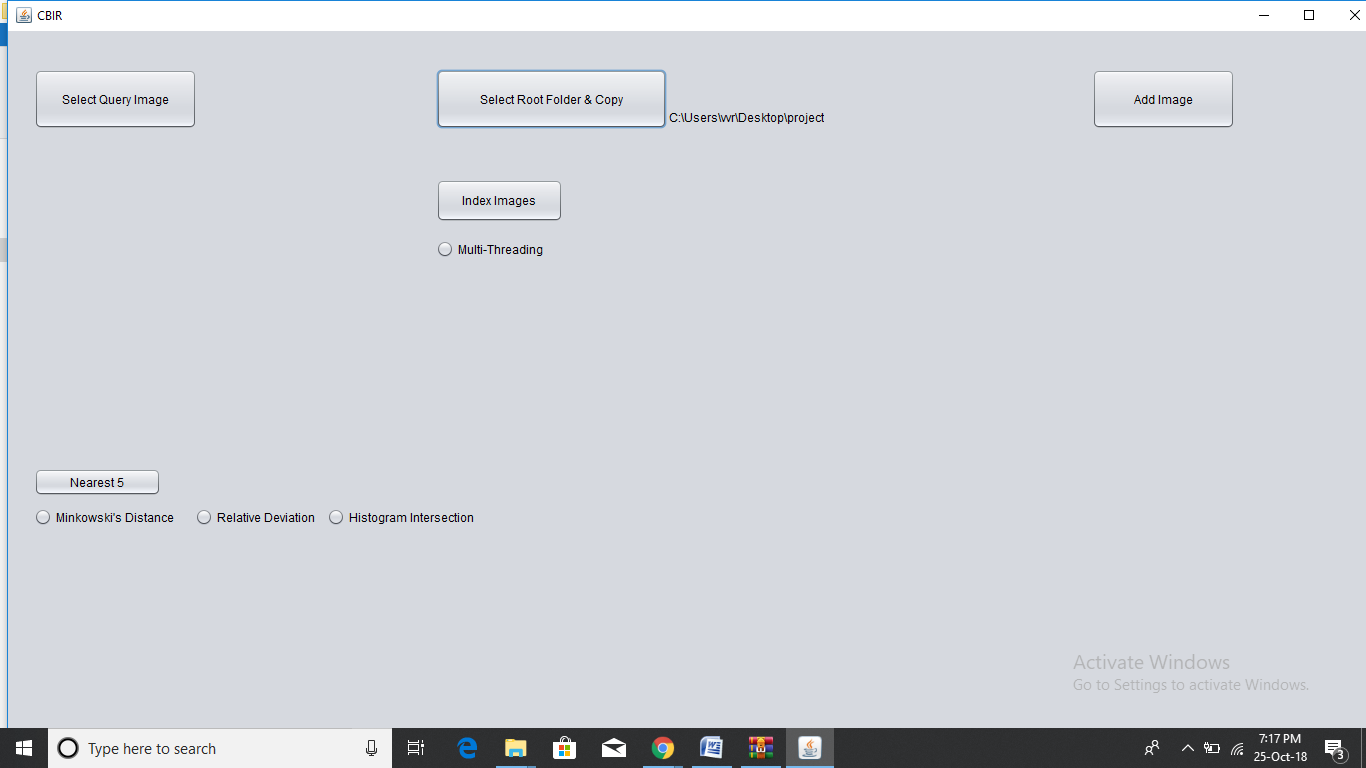
This is home page of our project content based image retrieval system.It contain buttons like select query image,select root folder and copy,add image,index images and nearest 5. Multi-Threading is kept in a radio button. Minkoswski’s Distance,Relative Deviation,Histogram Insertion are kept in radio buttons.



**Fig 5.1** **Home Page of CBIR**

**5.2 Selecting the Root Folder**

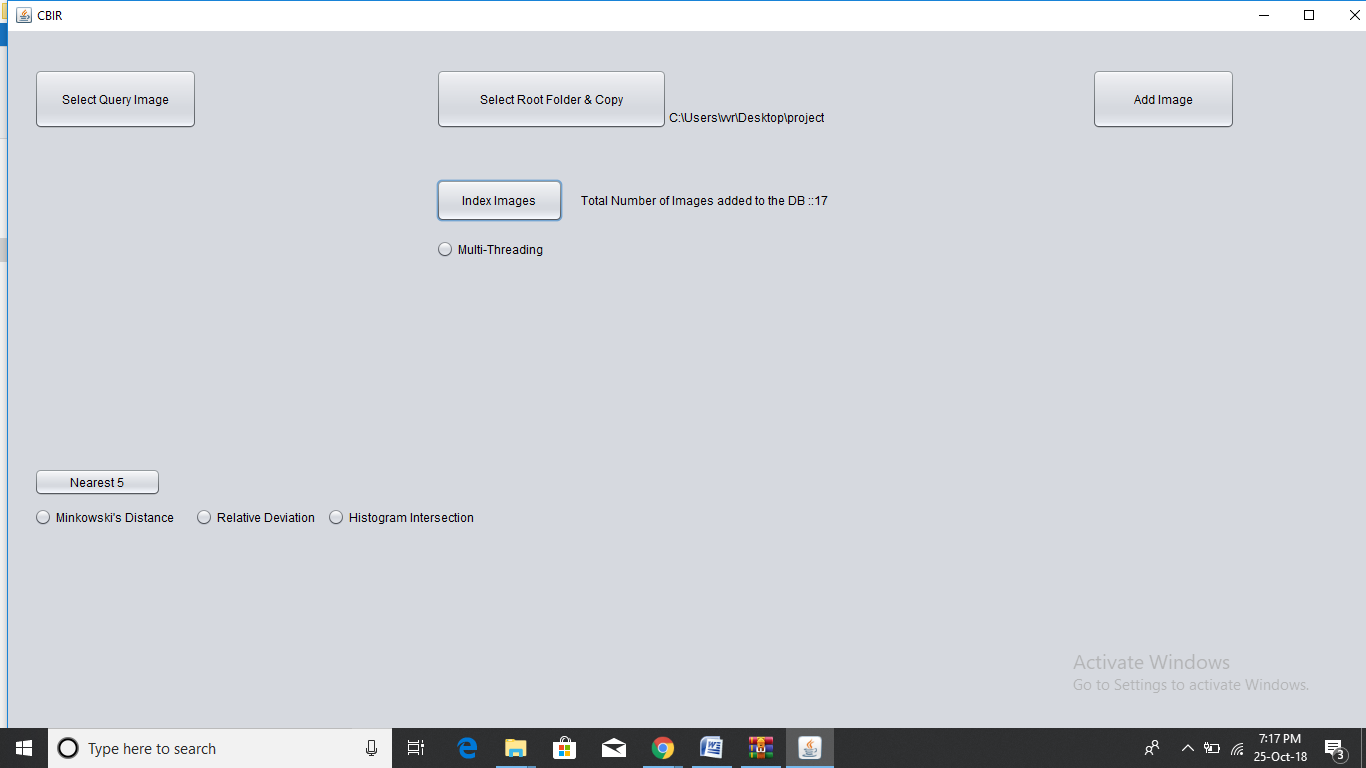
**Select** the Root Folder and Copy button , and here the root folder which contains all the images are to be selected. Initially a root folder is to be created. This folder contains all the images we want to use in the project.



**Fig 5.2 Selecting the Root Folder**

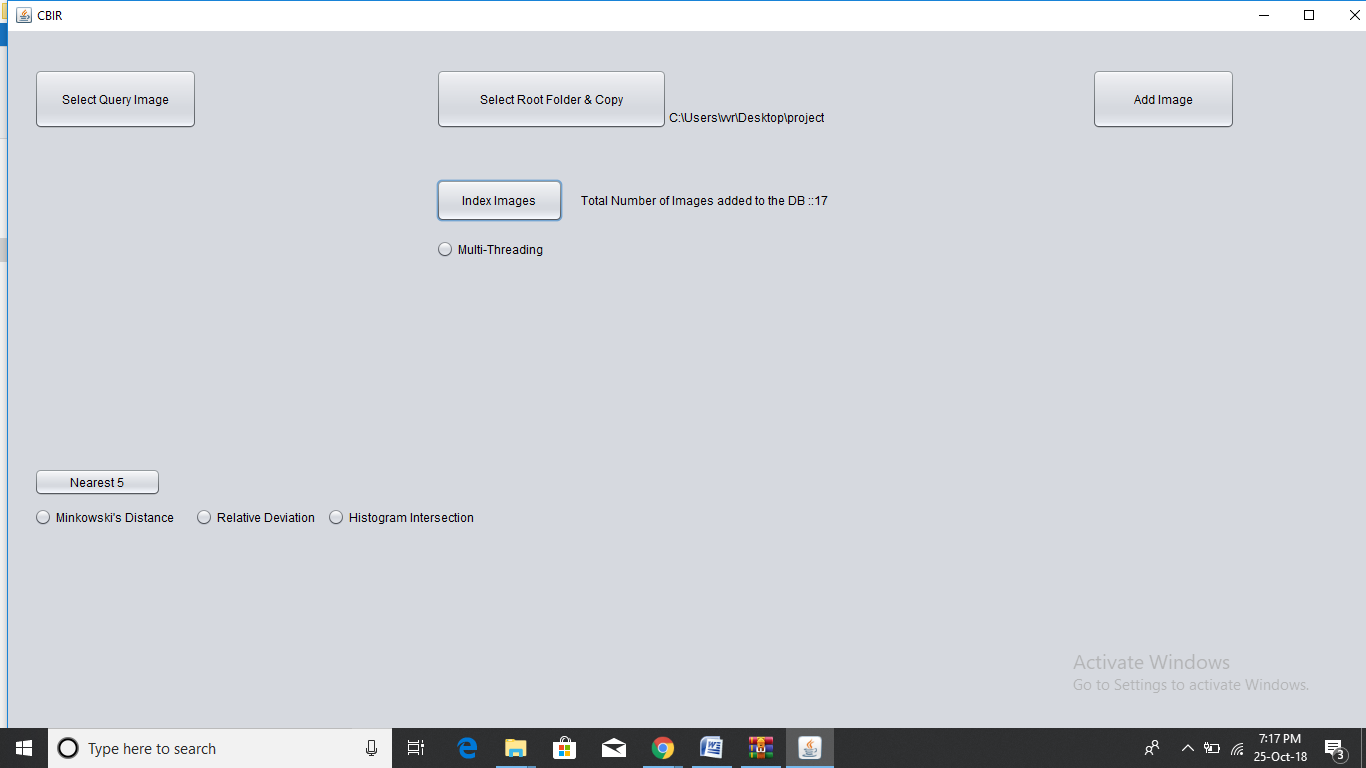
**5.3 Indexing of images**

After the root folder is selected, press the index images button so as to index the image to your Image Database and then it displays the total number of images added to the database.



**Fig 5.3 Indexing of images**

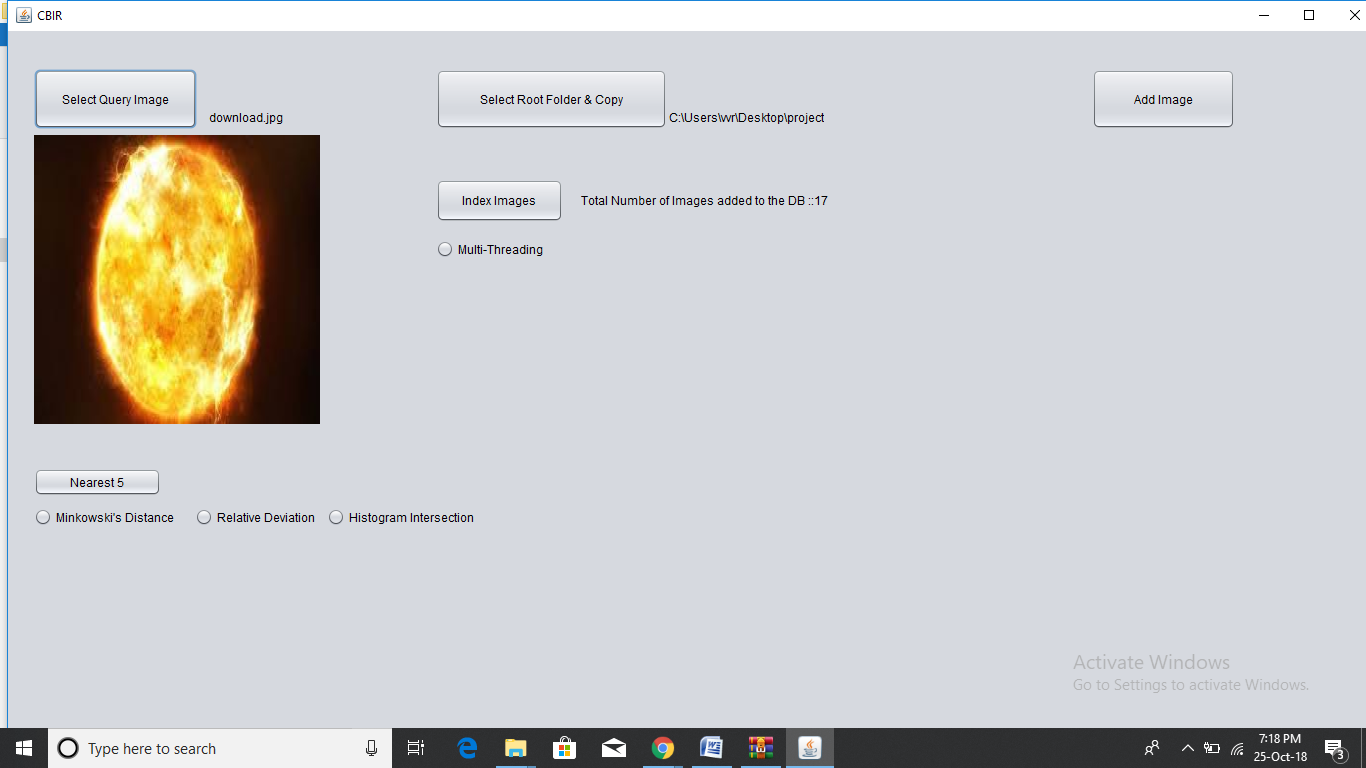
**5.4 Add image**

****

**Fig 5.4 Add image**

**5.5 Select Query image**

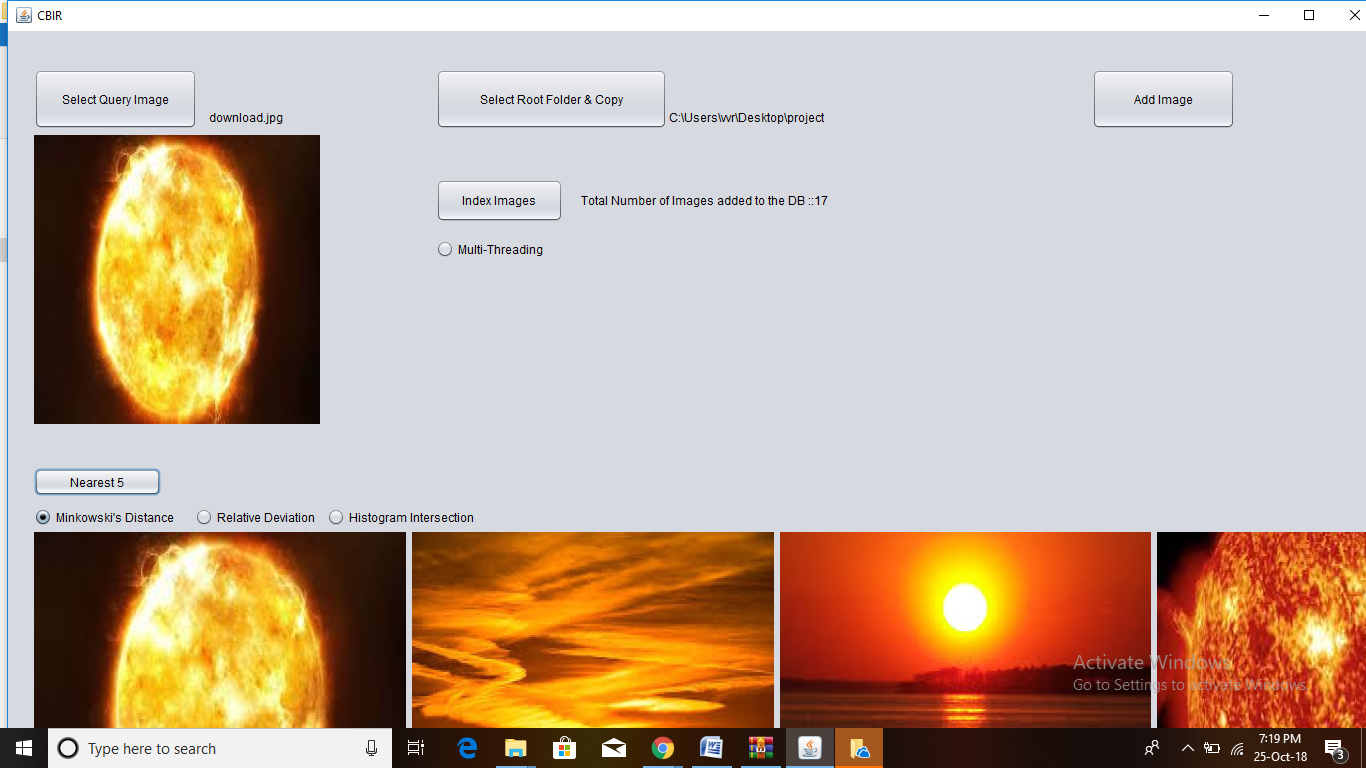
Through the Select Query Image button the Query image is inserted. You must select the image with which you want to compare the other Images.



**Fig 5.5 Select Query image**

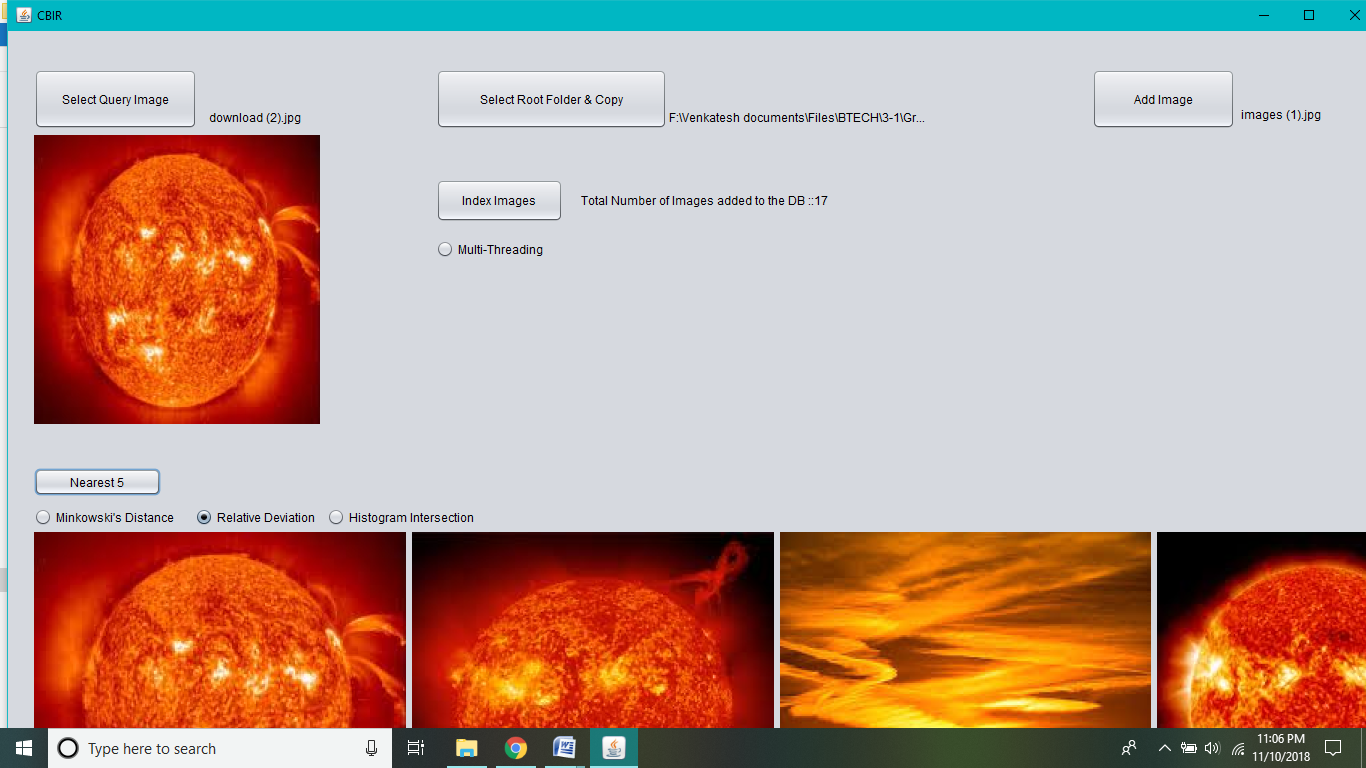
**5.6 Displaying the output of Minkowski’s distance**

The relevant images for the query image inserted are displayed here. The relevant images are displayed according to the type of distance selected.

****

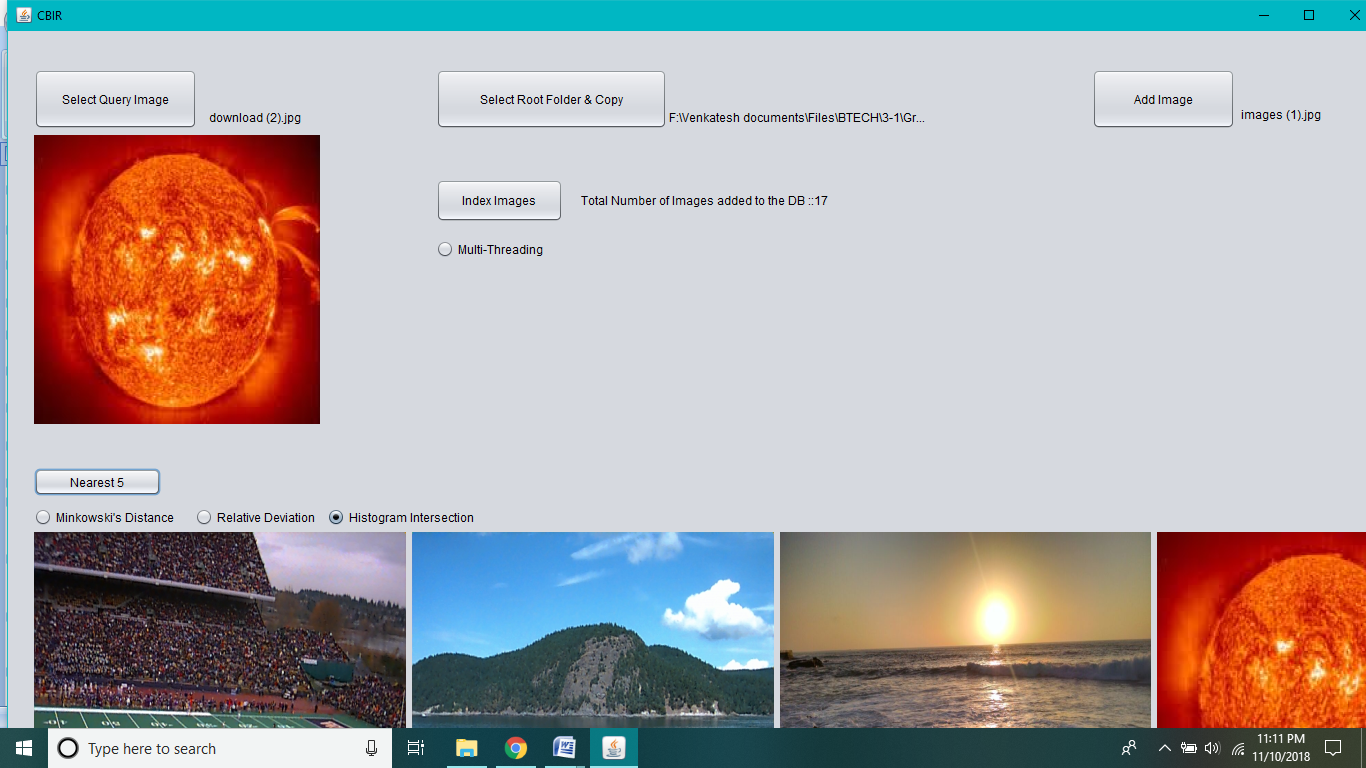
**Fig 5.6 Output of Minkowski’s distance**

**5.7 Displaying the output of Relative Deviation**



**Fig 5.7 Output of Relative Deviation**

**5.8 Displaying the output Histogram Intersection**

****

**Fig 5.8 Output of Histogram Intersection**

**6. CONCLUSION AND FUTURE SCOPE**

**Conclusion**

The scope to which CBIR expertise is currently in routine use is clearly still very limited. In particular, CBIR technology has so far had little impact on the more general applications of image searching, such as journalism or home entertainment. Only in very specialist areas such as crime prevention has CBIR technology been adopted to any significant extent. The dramatic rise in the sizes of images databases has stirred the develop ment of effective and efficient retrieval systems. The development of these systems started with retrieving images using textual connotations but later introduced image retrieval based on content. This came to be known as CBIR or Content Based Image Retrieval. Systems using CBIR retrieve images based on visual features such as color, texture and shape, as opposed to depending on image descriptions or textual indexing. In this project, we have researched various modes of representing and retrieving the image properties of color, texture and shape. Due to lack of time, we were only able to fully construct an application that retrieved image matches based on color and texture only. The submissions bring into play color correlogram and wavelet make over for indexing the metaphors and recoup them from image catalog. Color aloofness is considered using Euclidian equation and texture is premeditated using Euclidean difference involving the query image and salvaged imagery. CBIR is still a developing science. As image compression, digital image processing, and image feature extraction techniques become more developed, CBIR maintains a steady pace of development in the research field. Furthermore, the development of powerful processing power 68 and faster and cheaper memories contribute heavily to CBIR development. This development promises an immense range of future applications using CBIR. Future Enhancements Expansions and studies are going on for supplementary enhancements in intend and performance of ―content based image retrieval systems‖. In this implementation we have only done color analysis and texture analysis, the information about object location and shape is discarded. Thus this project showed that images retrieved by using the above mentioned methods may not be semantically related even though they share similar color distribution and texture in some results. In future a system that contains information about image location and its shape can be developed for improved retrieval.

**BIBLIOGRAPHY**

[1] Herbert Schildt, *The Complete Reference Java2* Fifth Edition, Tata McGraw-Hill Edition 2002.

[2] Grady Booch, James Rumbaugh, Ivar Jacobson. *The Unified Modeling Language User*

*Guide*. Addison-Wesley, Reading, Mass., 1999.

[3] *www.coderanch.com*

[4] *www.java.sun.com*

[5] *www.w3schools.com*

[6] *www.wikipedia.org*

**APPENDIX-A:JAVA TECHNOLOGY**

**About Java**:

Initially the language was called as “oak” but it was renamed as “java” in 1995.The primary motivation of this language was the need for a platform-independent(i.e. architecture neutral)language that could be used to create software to be embedded in various consumer electronic devices.

* Java is a programmer’s language
* Java is cohesive and consistent
* Except for those constraint imposed by the Internet environment. Java gives the programmer, full control

Java technology is both a programming language and a platform.

## The Java Programming Language

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

|  |  |
| --- | --- |
| * Simple * Object oriented * Distributed * Multithreaded * Dynamic | * Architecture neutral * Portable * High performance * Robust * Secure |

Each of the preceding buzzwords is explained in [The Java Language Environment](http://www.oracle.com/technetwork/java/langenv-140151.html) , a white paper written by James Gosling and Henry McGilton.

In the Java programming language, all source code is first written in plain text files ending with the .java extension. Those source files are then compiled into .class files by the javaccompiler. A .class file does not contain code that is native to your processor; it instead contains bytecodes — the machine language of the Java Virtual Machine[1](https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html#FOOT) (Java VM). The java launcher tool then runs your application with an instance of the Java Virtual Machine.



An overview of the software development process.

Because the Java VM is available on many different operating systems, the same .class files are capable of running on Microsoft Windows, the Solaris™ Operating System (Solaris OS), Linux, or Mac OS. Some virtual machines, such as the [Java SE HotSpot at a Glance](http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136373.html), perform additional steps at runtime to give your application a performance boost. This includes various tasks such as finding performance bottlenecks and recompiling (to native code) frequently used sections of code.



Through the Java VM, the same application is capable of running on multiple platforms.

## The Java Platform

A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Microsoft Windows, Linux, Solaris OS, and Mac OS. Most platforms can be described as a combination of the operating system and underlying hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

* The Java Virtual Machine
* The Java Application Programming Interface (API)

You've already been introduced to the Java Virtual Machine; it's the base for the Java platform and is ported onto various hardware-based platforms.

The API is a large collection of ready-made software components that provide many useful capabilities. It is grouped into libraries of related classes and interfaces; these libraries are known as packages. The next section, [What Can Java Technology Do?](https://docs.oracle.com/javase/tutorial/getStarted/intro/cando.html) highlights some of the functionality provided by the API.



The API and Java Virtual Machine insulate the program from the underlying hardware.

As a platform-independent environment, the Java platform can be a bit slower than native code. However, advances in compiler and virtual machine technologies are bringing performance close to that of native code without threatening portability.

The terms "Java Virtual Machine" and "JVM" mean a Virtual Machine for the Java platform.

# **Java Applet:**

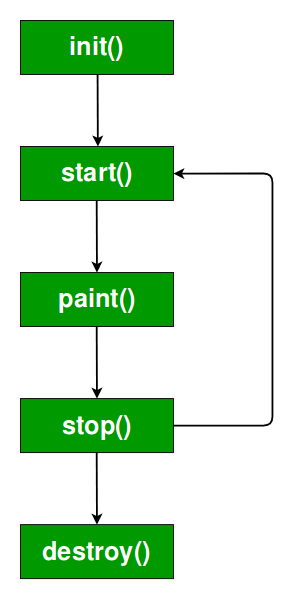
Applet is a Java program that can be embedded into a web page. It runs inside the web browser and works at client side. Applet is embedded in a HTML page using the APPLET or OBJECT tag and hosted on a web server.

Applets are used to make the web site more dynamic and entertaining.

**Some important points :**

1. All applets are sub-classes (either directly or indirectly) of [*java.applet.Applet*](https://docs.oracle.com/javase/7/docs/api/java/applet/Applet.html) class.
2. Applets are not stand-alone programs. Instead, they run within either a web browser or an applet viewer. JDK provides a standard applet viewer tool called applet viewer.
3. In general, execution of an applet does not begin at main() method.
4. Output of an applet window is not performed by System.out.println(). Rather it is handled with various AWT methods, such as drawString().

**Life cycle of an applet :**



It is important to understand the order in which the various methods shown in the above image are called. When an applet begins, the following methods are called, in this sequence:

1. init( )  
2. start( )  
3. paint( )

When an applet is terminated, the following sequence of method calls takes place:  
1. stop( )  
2. destroy( )  
Let’s look more closely at these methods.

1. **init( ) :**The **init( )** method is the first method to be called. This is where you should initialize variables. This method is called **only once** during the run time of your applet.
2. **start( ) :**The **start( )** method is called after **init( )**. It is also called to restart an applet after it has been stopped. Note that **init( )**is called once i.e. when the first time an applet is loaded whereas **start( )** is called each time an applet’s HTML document is displayed onscreen. So, if a user leaves a web page and comes back, the applet resumes execution at **start( )**.
3. **paint( ) :**The **paint( )** method is called each time an AWT-based applet’s output must be redrawn. This situation can occur for several reasons. For example, the window in which the applet is running may be overwritten by another window and then uncovered. Or the applet window may be minimized and then restored.

**paint( )** is also called when the applet begins execution. Whatever the cause, whenever the applet must redraw its output, **paint( )**is called.

The **paint( )** method has one parameter of type [Graphics](https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics.html). This parameter will contain the graphics context, which describes the graphics environment in which the applet is running. This context is used whenever output to the applet is required.

1. **stop( ) :**The **stop( )** method is called when a web browser leaves the HTML document containing the applet—when it goes to another page, for example. When **stop( )** is called, the applet is probably running. You should use **stop( )** to suspend threads that don’t need to run when the applet is not visible. You can restart them when **start( )** is called if the user returns to the page.
2. **destroy( ) :** The **destroy( )** method is called when the environment determines that your applet needs to be removed completely from memory. At this point, you should free up any resources the applet may be using. The **stop( )** method is always called before **destroy( )**.

# **Java Swings:**

**Java Swing** is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

Unlike AWT, Java Swing provides platform-independent and lightweight components.

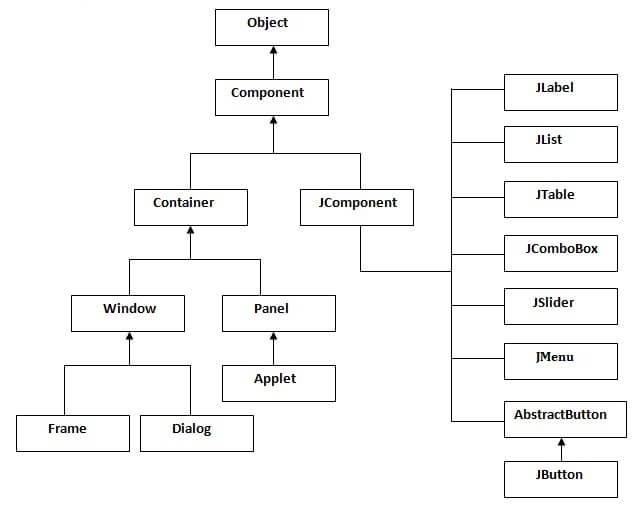
The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

### What is JFC?

The Java Foundation Classes (JFC) are a set of GUI components which simplify the development of desktop applications.

### Hierarchy of Java Swing classes:

The hierarchy of java swing API is given below.



**JButton:**

The JButton class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed. It inherits AbstractButton class.

# **JLabel:**

The object of JLabel class is a component for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly. It inherits JComponent class.

# **JRadioButton:**

The JRadioButton class is used to create a radio button. It is used to choose one option from multiple options. It is widely used in exam systems or quiz. It should be added in ButtonGroup to select one radio button only.

# **GroupLayout:**

**GroupLayout** groups its components and places them in a Container hierarchically. The grouping is done by instances of the Group class. Group is an abstract class and two concrete classes which implement this Group class are SequentialGroup and ParallelGroup.

SequentialGroup positions its child sequentially one after another where as ParallelGroup aligns its child on top of each other. The GroupLayout class provides methods such as createParallelGroup() and createSequentialGroup() to create groups. GroupLayout treats each axis independently. That is, there is a group representing the horizontal axis, and a group representing the vertical axis. Each component must exists in both a horizontal and vertical group, otherwise an IllegalStateException is thrown during layout, or when the minimum, preferred or maximum size is requested.

# **JFileChooser:**

The object of JFileChooser class represents a dialog window from which the user can select file. It inherits JComponent class.

**Hashing:**

Hashing is a process of converting an object into integer form by using the method hashCode(). Its necessary to write hashCode() method properly for better performance of HashMap. Here I am taking key of my own class so that I can override hashCode() method to show different scenarios.

**Histograms:**

**Histogram:** A graphical display of data using different heights.

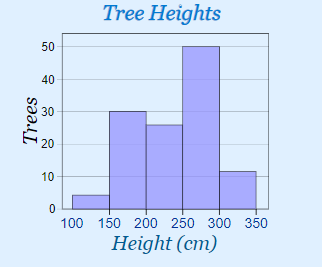
|  |  |
| --- | --- |
| Histogram | It is similar to a [Bar Chart](https://www.mathsisfun.com/data/bar-graphs.html), but a histogram groups numbers into **ranges**  And you decide what ranges to use! |



### Example: Height of Orange Trees

You measure the height of every tree in the orchard in centimeters (cm)

The heights vary from 100 cm to 340 cm



You decide to put the results into groups of 50 cm:

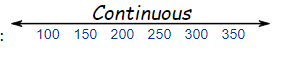
* The **100 to just below 150 cm** range,
* The **150 to just below 200 cm** range,
* etc...

So a tree that is 260 cm tall is added to the "250-300" range.

And here is the result:

You can see (for example) that there are **30** trees from **150 cm to just below 200 cm tall**

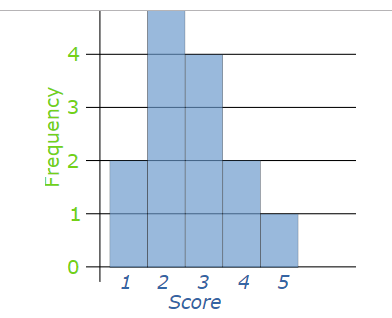
The horizontal axis is continuous like a [number line](https://www.mathsisfun.com/number-line.html):

****

The range of each bar is also called the **Class Interval**

**Frequency Histogram:**

A Frequency Histogram is a special histogram that uses vertical columns to show frequencies (how many times each score occurs):



|  |
| --- |
| Here I have added up how often 1 occurs (2 times),  how often 2 occurs (5 times), etc,  and shown them as a histogram. |
|  |
| [**Minkowski distance**](http://en.wikipedia.org/wiki/Minkowski_distance)**:**  The [Minkowski distance](http://en.wikipedia.org/wiki/Minkowski_distance) defines a distance between two points in a normed vector space.  C:\Users\kotha Rudra Tej\Pictures\Capture.PNG  Special cases:   * When p=1, the distance is known as the [Manhattan distance](https://github.com/compute-io/manhattan-distance). * When p=2, the distance is known as the [Euclidean distance](https://github.com/compute-io/euclidean-distance). * In the limit that p --> +infinity, the distance is known as the [Chebyshev distance](https://github.com/compute-io/chebyshev-distance).   **Relative standard deviation :**  Relative standard deviation is also called percentage relative standard deviation formula, is the deviation measurement that tells us how the different numbers in a particular data set are scattered around the mean. This formula shows the spread of data in percentage.  If the product comes to a higher relative standard deviation, that means the numbers are very widely spread from its mean. If the product comes lower, then the numbers are closer than its average.  It is also knows as the coefficient of variation.  The formula for the same is given as:  **RSD=s×100x¯¯¯**  Where, RSD = Relative standard deviation *s* = Standard deviation x¯¯¯ = Mean of the data. |

**APPENDIX-B:UNIFIED MODELING LANGUAGE**

The Unified Modeling Language (UML) is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts of a software system. It captures decisions and understanding about systems that must be constructed. It is used to understand, design, browse, configure, maintain, and control information about such systems. It is intended for use with all development methods, lifecycle stages, application domains, and media. The modeling language is intended to unify past experience about modeling techniques and to incorporate current software best practices into a standard approach. UML includes semantic concepts, notation, and guidelines. It has static, dynamic, environmental, and organizational parts. It is intended to be supported by interactive visual modeling tools that have code generators and report writers. The UML specification does not define a standard process but is intended to be useful with an iterative development process. It is intended to support most existing object oriented development processes.

The UML captures information about the static structure and dynamic behavior of a system. A system is modeled as a collection of discrete objects that interact to perform work that ultimately benefits an outside user. The static structure defines the kinds of objects important to a system and to its implementation, as well as the relationships among the objects. The dynamic behavior defines the history of objects over time and the communications among objects to accomplish goals.

Modeling a system from several separate but related viewpoints permits it to be understood for different purposes.

The UML also contains organizational constructs for arranging models into packages that permit software teams to partition large systems into workable pieces, to understand and control dependencies among the packages, and to manage the versioning of model units in a complex development environment. It contains constructs for representing implementation decisions and for organizing run-time elements into components.

UML is not a programming language. Tools can provide code generators from UML into a variety of programming languages, as well as construct reverse engineered models from existing programs. The UML is not a highly formal language intended for theorem proving. There are a number of such languages, but they are not easy to understand or to use for most purposes. The UML is a general-purpose modeling language. For specialized domains, such as GUI layout, VLSI circuit design, or rule-based artificial intelligence, a more specialized tool with a special language might be appropriate. UML is a discrete modeling language.It is not intended to model continuous systems such as those found in engineering and physics. UML is intended to be a universal general-purpose modeling language for discrete systems such as those made of software, firmware, or digital logic.











