CBIR for MR and CT images

by
WONG CHUNG HIONG
WEK 020285

Supervised by
Ms. Mangalam Sankupellay

Moderated by
Prof.Dr. Selvanathan Narainasamy

FACULTY OF COMPUTER SCIENCE AND
INFORMATION TECHNOLOGY
UNIVERSITY MALAYA

Sesi 2004/2005

Abstract

Image analysis has gained popularity in the medical field as there are many illnesses which require X-rays, Computerized Tomography (CT) scans and Magnetic Resonance (MR) images in order to enable accurate diagnosis of diseases, physician education and future research. The images collected through CT scans and MR imaging techniques require a database for efficient storage, search and retrieval. Medical image indexing, storage and retrieval based on image content are an important feature of image database system. This thesis explores the ability of Content-based Image Retrieval (CBIR) as a retrieve method and development in the domain of medical imaging. Content-based image retrieval (CBIR) refers to the ability to retrieve images on the basis of image content. Query by Image Content (QBIC) tool in IBM DB2 UDB will be use to apply this queries technique. In order to query the image by content in QBIC, first must create QBIC catalog for the images to catalog the images (adding entries for the images to the catalog and storing values for image features), then built a query to retrieve the images. The CBIR technique will be an indispensable component of images database.

Acknowledgement

The author would like take this opportunity to express her grateful to all who have guided and helped her in the research of this thesis topic.

First and foremost the author would like to extend her gratitude to her supervisor, Ms. Mangalam Sankupellay for her consistent effort and patience in assisting in the develpoment of this project. Without her guidance this thesis would not have propelled in the right direction to achieve the goals. Her commitment in making sure the author understands every aspect of the thesis thoroughly were of the utmost assistance that author appreciated.

Particularly mention must be made of the friends who helped author in getting the necessary resources that enable the project finish on time. The constant companionship and advices of the author's friends was greatly valued during the long time of work put into the develoment of this thesis.

Lastly but not the least the author paramountly values the help from all parties who has made this thesis become a reality.

Table of Contents

List of Figures

List of Table

List of Abbreviations

3-D	3-Dimensional
CBIR	Content-based Image Retrieval
CT	Computerized Tomography
DB2 AIV Extenders	DB2 Audio, Image and Video Extenders
DBMS	Database Management System
DICOM	Digital Imaging and Communication in Medicine
IBM DB2 UDB	IBM DB2 Universal Database
MRI	Magnetic Resonance Imaging
PET	Positron Emission Tomography
QBIC	Query by Image Content
SQL	Structured Query Language

Chapter 1 Introduction

1.1 Overview

The medical imaging field has grown substantially in recent years and has generated additional interest in methods and tools for the management, analysis, and communication of medical images.

Increasing computerization of the process of acquiring and storing diagnostic images requires more sophisticated approaches to image retrieval in order to optimally use the images for improved health-care delivery, physician education and research. Content-based image retrieval (CBIR) or content-based visual information retrieval is a vivid research area in the field of computer vision over the last 10 years. Many programs and tools have been developed to formulate and execute queries based on the visual or audio content and to help browsing large multimedia repositories. Still, no general breakthrough has been achieved with respect to large varied databases with documents of differing sorts and with varying characteristics.

CBIR refers to the ability to retrieve images on the basis of image content, as opposed to on the basis of some textual description of the images. Image content is typically defined by a set of features extracted from an image that describe the color, texture and/or shape of the entire image or of specific image regions. Given a query image, the goal of a CBIR system is to search the database and return the *n* most visually similar images to the query image. An example of a retrieval query of this sort would be "show me images from a given database that is similar to a particular image". A key element of this approach revolves around the types of patterns that can be recognized by

the computer and that can serve as the index of the data retrieval. CBIR differs fundamentally in its focus from a variety of pattern recognition and artificial intelligence techniques that have been applied to computer-aided medical image interpretation. In CBIR the emphasis is on the development of an efficient and practical database methodology for recognizing and retrieving patterns in medical images that represent pathological processes.

Technologies Computerized Tomography (CT) and Magnetic Resonance Imaging (MRI) have radically changed the way physicians diagnose and treat disease. Computed Tomography also known as computed axial tomography, or CAT scan, medical technology that uses X rays and computers to produce three-dimensional images of the human body. Unlike traditional X rays, which highlight dense body parts, such as bones, CT provides detailed views of the body's soft tissues, including blood vessels, muscle tissue, and organs, such as the brain. While conventional X rays provide flat two-dimensional images, CT images depict a cross-section of the body. Newer technology, MRI scan uses a powerful magnet to take pictures of inside the body. MR images are created by using a magnetic field, radio waves, and a computer. MRI can generate thin-section images of any part of the body—including the heart, arteries, and veins—from any angle and direction, without surgical invasion and in a relatively short period of time. [1],[2],[22]

With digital imaging and communication in medicine (DICOM), a standard for image communication has been set and patient information can be stored with the actual image, although still has a few problems prevail with respect to the standardization. Content-based access to medical images for supporting clinical decision making has been proposed that would ease the management of clinical data.

It needs to be stated as well that the goal of CBIR is not, in general to replace text-based retrieval methods as they exist at the moment but to complement them with visual search tools. It will be an indispensable component of image database.

1.2 Problem Definition

The ever-increasing volume of medical images, the economic impracticality of manually indexing these images, and the inadequacy of human language alone to describe image contents that are visually recognizable and medically significant, such as shape and geometry, color, texture of objects within images, all provide impetus for research and development toward practical *Content-Based Image Retrieval* (CBIR) systems that could become a standard offering of the medical library of the future.

Basic to a CBIR system is the ability to efficiently manage the performance, storage, and memory requirements of images. CBIR system (i) have the capability to retrieve image based on their contents; (ii) are domain independent; and (iii) can be automated. Because of their size and complexity, images have special computation and resource requirements. Some issues have identified.

- understanding image users' needs and information-seeking behavior
- identification of suitable ways of describing image content
- extracting such features from raw images
- matching query and stored images in a way that reflects human similarity judgments
- efficiently accessing stored images by content

1.3 Project Aims

Traditionally, images were stored in the flat file, which are difficult to search. This project is to develop images databases, which provide efficient acquisition and storage of medical images.

The aim of this project is to clarify some of the issues raised by CBIR technique, by reviewing its current capabilities and limitations, and its potential usefulness to users in higher education and elsewhere.

Research effort are focused to the available literature in the field of content-based access to medical images data and on the technologies used in the field to develop a medical image storage and retrieval system.

The overall goal of this project is to make a significant contribution to the image- by advancing CBIR techniques applied to the MR and CT images. Specific objectives include conduct research and identify the steps needed for content-based indexing, feature extraction, feature classification, apply the techniques for image-based retrieval and develop the algorithms needed to implement both image indexing and retrieval.

1.4 Project Scope

CBIR system for CT and MR images will be develop according to the following parameters:

a) Develop a database containing the details and medical images of the patient.

b) Provide search methods to the user for image searching

c) Create an interface as primary model of communication between the users and the database.

d) Provide a link to the 3 Dimensional data files of patient

The project is basically revolving around understanding the content-based image retrieval technique algorithm and code the algorithm using a chosen programming language, mostly Structured Query Language (SQL). SQL is a powerful tool for manipulating data. It is the de facto standard query language for relational database management systems (RDBMSs) and is used by leading RDBMS products such as Oracle, IBM DB2, Sybase and Microsoft SQL.

1.5 Project limitations

There are some constraints and limitations in this project research and development. This project design for store and retrieve only CT and MR images. These images are in grey level format, therefore cannot retrieve by color features. Most articles that propose content-based queries do give details use color features, mostly in the form of color histogram. However, as many of the images in medical domain do not contain colors, texture and shape feature should use to query the image content.

The CT and MR images need particular software to open and view its. Therefore these images should converted first to image format supported by development tools.

Due to the lack of knowledge in CBIR technique plus the complexity of the actual system, many research and study should conduct in order to discover news fact and information. Therefore I have chosen to develop the main feature of the system and have been left out some other features.

1.6 Project Development Schedule

Project scheduling cursors of the whole development activities is carefully planed out to achieve a systemic progress and ensure on-time delivery of this project. It is important to have a project schedule as it act as a time management and control to the developer. This will make sure I am in route of the direction of this project. The following Gantt chart illustrates the point.

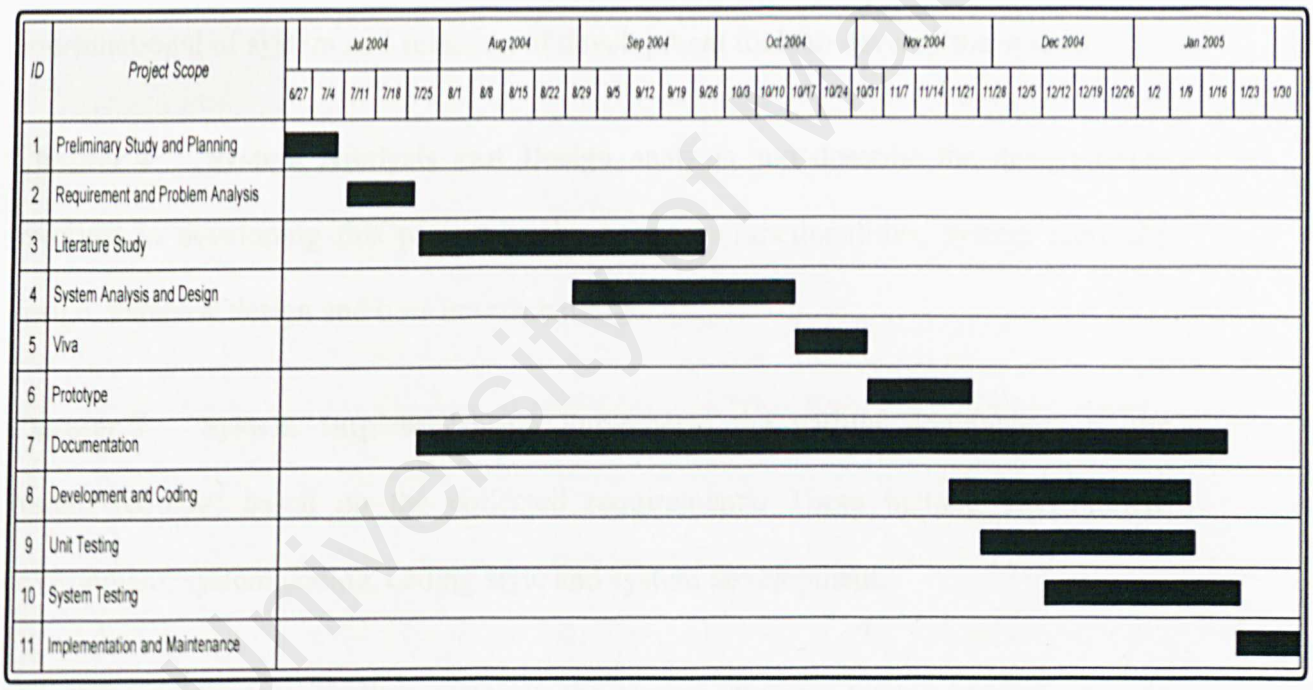


Figure 1.1 Proposed Gantt Chart

1.7 Report layout

Chapter 1 **Introduction** serves as introduction to the entire project. It overviews the project system, objectives, scopes, aims expected outcome, project schedule and facilities required.

Chapter 2 **Literature Review** focus on the survey, CBIR features, capabilities and system architecture, technologies and tools that will be apply to the project system.

Chapter 3 **Methodology** fairly discusses the development methodology, functional, non-functional of system and selection of development tools to develop the system.

Chapter 4 **System Analysis and Design** analysis and describe the design issues involved in developing this project, such as system functionalities, system hierarchy design, database design and user interface design.

Chapter 5 **System Implementation** implements the various components of the system (coding) based on the collected requirements. These include development environment, system coding, coding style and system development.

Chapter 6 **System Testing** performs the testing strategy, testing manual or code results of testing and changes made after testing.

Chapter 7 **Discussion and Conclusion** mention about the problem encountered and recommended solutions, system strengths, system limitation and future enhancements and conclusion.

Chapter 2 Literature Review

2.1 Introduction to image retrieval

Interest in the potential of digital images has increased enormously over the last few years. Users in many professional fields are exploiting the opportunities to access and manipulate remotely-stored images in many ways. However, if we are aiming at automatic retrieval of images, it should be unsurprising that encapsulating the semantics or meaning in the image is an difficult challenge. This is because visual features of images are difficult to be described using words and the process of locating a desired image in a large and varied collection of images can be a very tiring and time consuming.

The problems of image retrieval are becoming widely recognized, and the search for solutions increasingly an active area for research and development. An image retrieval system is a computer system for browsing, searching and retrieving images from a large database of digital images. It can be based on a textual description of the images in the database. In the text-based systems, first the images are annotated manually and then use a Text-based Database Management System (DBMS) for image retrieval, but the manual annotation of the images is time-consuming and ambiguous. Therefore, content-based image retrieval (CBIR) aims at avoiding a textual description and retrieving images based on their visual similarity to a user-supplied query image instead.

Increasing computerization of the process of acquiring and storing medical images such as Computerized Tomography (CT) and Magnetic Resonance (MR)

imaging requires more sophisticated approaches to image storage and retrieval in order to optimally use the images for physician education, research and etc.

The main difference between the general image formats and the medical image formats is that the latter usually contains more information such as patient's name and the settings of the equipment used to produce the image. The image data includes the pixel image data and information such as the numbers of bits per pixel, the binary storage of the pixels, the size of the image and the number of images per file. Usually medical image data includes additional information such as description of the image, results of the analysis on the data, a history of the processing of the data. There are many formats for storing medical image data. The most used are the TIFF (tagged image file format). Medical image conversion system was designed and implemented to perform conversions between the most common medical image formats. [12],[13],[16]

2.2 Content Based Image Retrieval (CBIR)

The earliest use of the term *content-based image retrieval* in the literature seems to have been by Kato[1992], to describe his experiments into automatic retrieval of images from a database by colour and shape feature. The term has since been widely used to describe the process of retrieving desired images from a large collection on the basis of features (such as colour, texture and shape) that can be automatically extracted from the images themselves. The features used for retrieval can be either primitive or semantic, but the extraction process must be predominantly automatic. Retrieval of images by manually-assigned keywords is not CBIR as the term is generally understood, even if the keywords describe image content.

CBIR differs from classical information retrieval in that image databases are essentially unstructured, since digitized images consist purely of arrays of pixel intensities, with no inherent meaning. One of the key issues with any kind of image processing is the need to extract useful information from the raw data (such as recognizing the presence of particular shapes or textures) before any kind of reasoning about the image's contents is possible.

Research and development issues in CBIR cover a range of topics, many shared with mainstream image processing and information retrieval. Currently, CBIR researches mainly focus on three topics: *feature extraction*, *efficient indexing* and *user interface*.

- **Feature extraction.** Features of image include *primitive features* and *semantic/logical features*. Primitive features such as *color*, *texture*, *shape* and *spatial relationships* are quantitative in nature, they can be extracted automatically or semi-automatically. Logical features are qualitative in nature and provide abstract representations of visual data at various levels of detail. Typically, logical features are extracted manually or semi-automatically. One or more features can be used in a specific application. For example, in a satellite information system, the texture features are important, while shape and color features are more important in trademark registration systems. The texture and shape features gain important in medical images. Once the features have been extracted, the retrieval becomes a task of measuring similarity between image features.

- **Efficient indexing.** To facilitate efficient query and searching processing, the image indexes needed to be organized into efficient data structure. Image features are

approximately represented, they may not have embedded order, and they may have inter related multiple attributes. Therefore, flexible data structures should be used in order to facilitate storage/retrieval in a image retrieval system.

- **User interface** In visual information systems, user interaction plays an important role in almost all of its functions. The user interface consists of a query processor and a browser to provide the interactive graphical tools and mechanisms for querying and browsing the database, respectively. Common query mechanisms provided by user interface are: *query by keyword, query by sketching, query by example, browsing by categories, feature selection and weighting, retrieval refining* and *relevance feedback.*

These three tasks comprise three key components of a CBIR system. Among these three main tasks, feature extraction (including similarity measurement) is the most paramount and difficult task. The majority of CBIR research goes into this challenging task. [13], [14], [15], [16], [20]

2.3 Characteristics of image queries

What kinds of query are users likely to put to an image database? To answer this question in depth requires a detailed knowledge of user needs – why users seek images, what use they make of them, and how they judge the utility of the images they retrieve. Common sense evidence suggests that images are required for a variety of reasons, including:

- illustration of text articles, conveying information or emotions difficult to describe in words,
- display of detailed data (such as radiology images) for analysis,
- formal recording of design data (such as architectural plans) for later use.

Access to a desired image from a repository might thus involve a search for images depicting specific types of object or scene, or containing a specific texture or pattern. Potentially, images have many types of attribute which could be used for retrieval, including the presence of a particular combination of colour, texture or shape features.

From the above information, it is known that image in database can be indexed and retrieve using either textual information or content information. Image content can be described using primitive features such as color, texture, shape or the combination of them. This research adopts CBIR approach that is, indexing and retrieving image by its content.

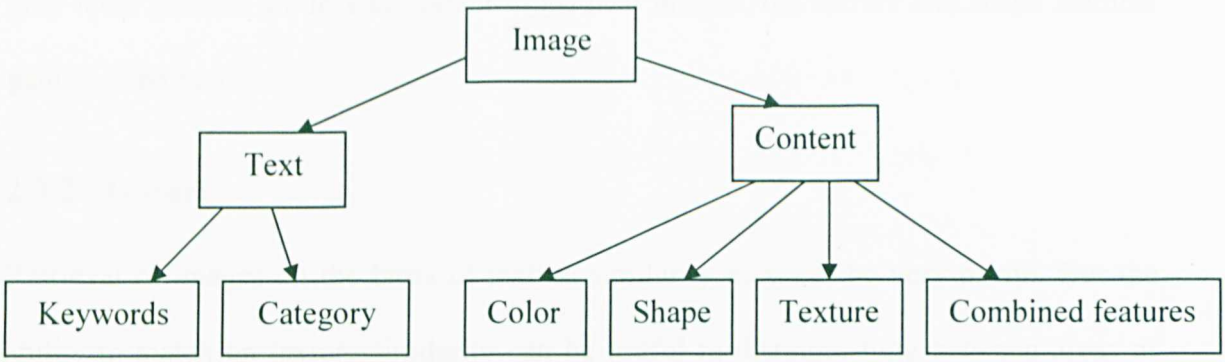


Figure 2.1 Image descriptions using different information

2.3.1 Color

In stock photography (large, varied databases for being used by artists, advertisers and journalists), color has been the most effective feature and almost all systems employ colors. Although most of the images are in red, green, blue (RGB) color space, this space is only rarely used for indexing and querying as it does not correspond well to the human color perception. It only seems reasonable to be used for images taken under exactly the same conditions each time such as trademark images. Others spaces such as hue, saturation and lightness (HSL) are much better with respect to human perception and are more frequently used. This mean that differences in the color space are similar to the differences between colors that human perceive. Much effort has also been spent on creating color spaces that are optimal with respect to lighting conditions or that are invariant to shape and other influences such as viewing position. This allows identifying colors even under varying conditions but on the other hand information about the absolute colors is lost. In specialized fields, namely in the medical domain, absolute color or grey level features are often of very limited expressive power unless exact reference points exist as it is the case of computerized tomography images. As color and

grey level features are less important in medical images, the texture and shape features gain in important.

2.3.2 Texture

Retrieval of images on the basis of texture similarity may not be very useful. But the ability to match on texture similarity can be useful in distinguishing between areas of images with similar color. Most of the techniques do calculate the relative brightness of selected pairs of pixels from each image. From these it is possible to calculate measures of image texture such as the degree of contrast, coarseness, directionality and regularity, periodicity, directionality and randomness. Texture queries can be formulated in a similar manner to color queries, by selecting examples of desired textures from a palette, or by supplying an example query image. The system then retrieves images with texture measures most similar in value to the query.

Some of the most common measures for capturing the texture of images are wavelets and Gabor filters. The Gabor filters do seem to perform better and correspond well to the properties of the human visual cortex for edge detection. These texture measures try to capture the characteristics of the image or image parts with respect to changes in certain directions and the scale of the changes. This is most useful for regions or images with homogeneous texture. [14], [15], [16]

2.3.3 Shape

Most researches on CBIR have contributed to color/texture based indexing and retrieval. Comparatively, little work has been done on image retrieval using shape.

Shape is one of key visual features used by human for distinguishing visual data along with other features of color and texture. Compare with color and texture, shape is easier for user to describe in the query, either by example or by sketch. While for color and texture feature, the query is usually presented by example, because it is impractical for ordinary users to sketch a colored or a textured image as query.

The ability to retrieve by shape is perhaps the most obvious requirement at the primitive level. Unlike texture, shape is a fairly well-defined concept – and there is considerable evidence that natural objects are primarily recognized by their shape. A number of features characteristic of object shape (but independent of size or orientation) are computed for every object identified within each stored image. Queries are then answered by computing the same set of features for the query image, and retrieving those stored images whose features most closely match those of the query. Queries to shape retrieval systems are formulated either by identifying an example image to act as the query, or as a user-drawn sketch. [3], [13], [20]

2.4 Overview of CT and MR images

Medical images are at the heart of the healthcare diagnostic procedures. They have provided not only a noninvasive mean to view anatomical cross-sections of internal organs, tissues, bone and other features of patients but also a mean for physicians to evaluate the patient's diagnosis and monitor the effects of the treatment, and for investigators to conduct research of the underlying disease.

Medical images come from a board spectrum of imaging sources such as computerized tomography (CT), magnetic resonance imaging (MRI), digital mammography and positron emission tomography (PET), and they generate a staggering amount of image data and important medical information.

2.4.1 Computerized Tomography (CT)

Computed Tomography also known as computed axial tomography, or CAT scan, medical technology that uses X rays and computers to produce three-dimensional images of the human body. Unlike traditional X rays, which highlight dense body parts, such as bones, CT provides detailed views of the body's soft tissues, including blood vessels, muscle tissue, and organs, such as the brain. While conventional X rays provide flat two-dimensional images, CT images depict a cross-section of the body.

A patient undergoing a CT scan rests on a movable table at the center of a donut-shaped scanner, which is about 2.4 m (8 ft) tall. The CT scanner contains an X-ray source, which emits beams of X rays; an X-ray detector, which monitors the number of X rays that strike various parts of its surface; and a computer. The source and detector face each other on the inside of the scanner ring and are mounted so that they rotate around the rim of the scanner. Beams from the X-ray source pass through the patient and are recorded on the other side by the detector. As the source and detector rotate in a 360° circle around the patient, X-ray emissions are recorded from many angles. The resulting data are sent to the computer, which interprets the information and translates it into images that appear as cross-sections on a television monitor. By moving the patient

within the scanner, doctors can obtain a series of parallel images, called slices. Doctors analyze a series of slices to understand the three-dimensional structure of the body.

To enhance an image, patients may be given an injection of a substance that will increase the contrast between different tissues. The patient may also be asked to drink a liquid that makes internal organs more clearly visible in the CT scan. Contrast agents are commonly used for scans of the chest, abdomen, and pelvis.

The speed at which CT slices are obtained has increased dramatically since this technology was developed during the late 1960s and early 1970s by British engineer Sir Godfrey Newbold Hounsfield and American physicist Allan MacLeod Cormack. The first CT scanner required 4.5 minutes of scanning and 1.5 minutes of computer reconstruction to generate a single slice. The latest generation of CT scanners creates a slice in about one second.

CT was first used to view the tissues of the brain, and this is still its most frequent application. CT of the brain can show if an accident victim has sustained a brain injury, or if bleeding is occurring in the brain of a stroke victim. CT is also commonly used to diagnose disorders involving the chest, abdomen, spine, and pelvis. CT may be used to determine the location and size of a cancerous tumor and whether the cancer has spread to other parts of the body. Newer applications for CT include CT angiography, which provides detailed images of the interior of small blood vessels and images of the interior of hollow structures, such as the colon and the airways of the lungs. [1], [2], [22]

2.4.2 Magnetic Resonance Imaging (MRI)

Magnetic Resonance Imaging (MRI), medical diagnostic technique that creates images of the body using the principles of nuclear magnetic resonance. A versatile, powerful, and sensitive tool, MRI can generate thin-section images of any part of the body—including the heart, arteries, and veins—from any angle and direction, without surgical invasion and in a relatively short period of time. MRI also creates "maps" of biochemical compounds within any cross section of the human body. These maps give basic biomedical and anatomical information that provides new knowledge and may allow early diagnosis of many diseases.

MRI is possible in the human body because the body is filled with small biological "magnets," the most abundant and responsive of which is the proton, the nucleus of the hydrogen atom. The principles of MRI take advantage of the random distribution of protons, which possess fundamental magnetic properties. Once the patient is placed in the cylindrical magnet, the diagnostic process follows three basic steps. First, MRI creates a steady state within the body by placing the body in a steady magnetic field that is 30,000 times stronger than the earth's magnetic field. Then MRI stimulates the body with radio waves to change the steady-state orientation of protons. It then stops the radio waves and "listens" to the body's electromagnetic transmissions at a selected frequency. The transmitted signal is used to construct internal images of the body using principles similar to those developed for computerized axial tomography, or CAT scanners.

In current medical practice, MRI is preferred for diagnosing most diseases of the brain and central nervous system. MRI scanners provide equivalent anatomical

resolution and superior contrast resolution to that of X-ray CAT scanners. They produce functional information similar to that of positron emission tomography (PET) scanners but with superior anatomical detail. MRI scanners also provide imaging complementary to X-ray images because MRI can distinguish soft tissue in both normal and diseased states. Although an MRI scan is relatively expensive, it may actually reduce costs to patients and hospitals by providing diagnostic evaluation to outpatients and thereby frequently limiting more expensive hospitalization. Because it does not use ionizing radiation, MRI is risk free except for patients with cardiac pacemakers, patients who might have an iron filings next to their eyes (for example, sheet metal workers), patients with inner ear transplants, and patients with aneurysm clips in their brains.

Some experts once expected that CT would be replaced by the newer technology of magnetic resonance imaging (MRI), which uses magnetically charged substances within the body to create diagnostic images. Nevertheless, CT continues to have several advantages over MRI, including lower cost, wider availability, shorter examination time, superior imaging of the chest and abdomen, and the ability to create images without the strong magnetic field required by MRI. At present, CT and MRI are complementary, rather than competitive technologies. [1], [6], [8], [12], [22]

2.4.3 The Anatomical Planes

Four imaginary planes pass through the body in the anatomical position, which help to describe movements and body positions. These are the median, sagittal, coronal, and horizontal planes.

The median plane is an imaginary vertical plane that passes through the middle of the body, dividing it into left and right halves. The sagittal plane is a vertical plane which runs parallel to the median plane, but does not necessarily pass through the body's midline. In effect, the median plane is a specific type of sagittal plane. The coronal plane is a vertical plane which is perpendicular to the median and sagittal planes, and is sometimes also referred to as the frontal plane. The horizontal plane, or transverse plane, is a plane which splits the body or body part in question into upper and lower parts. The diagram below shows each of these planes.

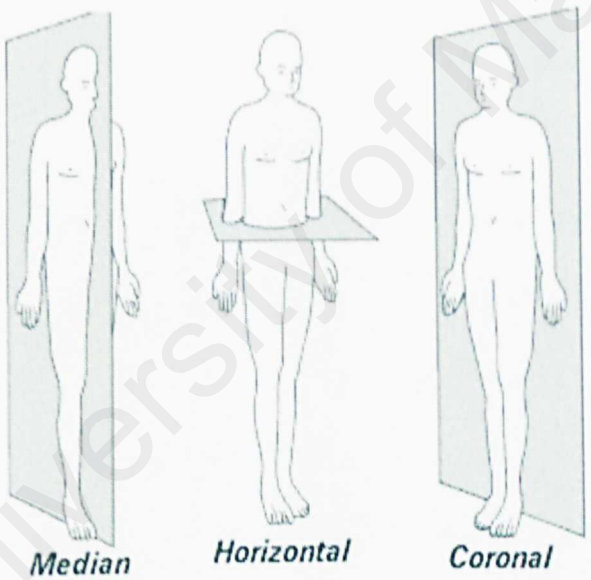


Figure 2.2 Median, Horizontal and Coronal planes

There is a language used by medical practitioners of all kinds (chiropractors, surgeons, nurses, etc). When discussing the body and its ailments the basic terms need to know are: **a) Sagittal:** left and right b) **Transverse:** divides top and bottom. c) **Coronal:** divides front and back. . [1],[2],[22]

Sagittal View Transverse View Coronal View

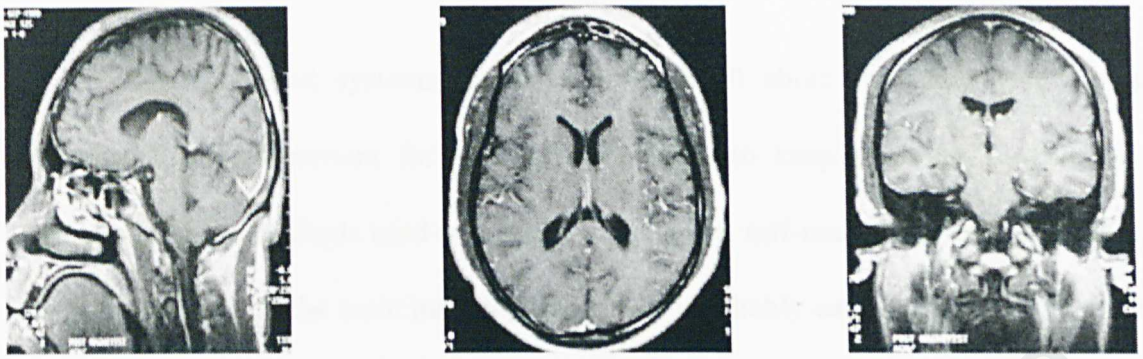


Figure 2.3 Anatomy of the Brain - MR images in 3 planes

2.5 Storage and Access methods

Although most systems do not talk in detail about the underlying storage, however this is important for interactive systems to keep response times at bay. Common storage methods used are relational database, self-made structures or simply to keep entire index in the main memory which will inevitably cause problems when using large databases.

A database management system gives the user access to data that is transformed into information. This system allow users to create, update and extract information from their database Compare to a manual filling system, the advantages of a computerized database system are speed, accuracy and accessibility. Relational database system is good for managing large amount of data. It is based on two-dimensional tables in which each item appears as a raw. The table was originally called as relations which led to the name relational database. The relationships among the data are expressed by comparing the values stored in the tables. Query languages like Structured Query Language (SQL) are used to retrieve information from the tables. The relational database provides important features like fast querying, sharing of objects among program and permanent storage.

2.6 Review of Shape Representation and Description Techniques

Shape is an important visual feature and it is one of the primitive features for image content description. However, shape content description is a difficult task. Because it is difficult to define perceptual shape features and measures the similarity between shapes. To make the problem more complex, shape is often corrupted with noise, defection, arbitrary distortion and occlusion.

Shape research has been an active research area for over 30 years. In the past, shape research has been driven mainly by object recognition. As a result, techniques of shape representation and description mostly target at particular applications. Effectiveness or accuracy is the main concern of these techniques. In contrast, in the newly emerged multimedia application CBIR, efficiency is envisaged equally important as effectiveness due to online retrieval demand. A new world standard MPEG-7, known as "Multimedia Content Description Interface" has been developed to address the issue of audio/visual information description. In the development of MPEG-7, six requirements has been set to measure a shape descriptor, they are: good retrieval accuracy, compact features, general application, low computation complexity, robust retrieval performance and hierarchical coarse to fine representation.

Generally, there are two types of approaches in shape representation. One is the contour based shape method and the other is the region based method. Region based method is a general method which can be applied to generic shapes.

2.6.1 Contour-based Shape Representation Techniques

Contour shape techniques only exploit shape boundary information. There are generally two types of very different approaches for contour shape modeling: continuous approach (global) and discrete approach (structural). Continuous approaches do not divide shape into sub-parts; a feature vector derived from the integral boundary is used to describe the shape. The measure of shape similarity is either point-based matching or feature-based matching. Discrete approaches break the shape boundary into segments, called primitives using a particular criterion. The final representation is usually a string or a graph (or tree); the similarity measure is done by string matching or graph matching. In the following, these two types of approaches are discussed.

2.6.1.1 Global Methods

Global contour shape representation techniques usually compute a multi-dimensional numeric feature vector from shape boundary information. The matching between shapes is a straightforward process, which is usually conducted by using a metric distance, such as Euclidean distance or city block distance. Point feature based matching is also used in particular applications.

2.6.1.2 Structural Methods

Another member in shape analysis family is structural shape representation. With structural approach, shapes are broken down into boundary segments called primitives. Structural methods differ in the selection of primitives and the organization of the primitives for shape representation. Common methods of boundary decomposition are based on polygonal approximation, curvature decomposition and curve fitting.

2.6.2 Region-based Shape Representation Techniques

In region based methods, all the pixels within a shape region are taken into account to obtain the shape representation. Common region based methods use moment descriptors to describe shape. Other region based methods include grid method, shape matrix, convex hull and media axis.

2.6.2.1 Global Methods

Global methods treat shape as a whole; the result representation is a numeric feature vector which can be used for shape description.

2.6.2.2 Structural Methods

Similar to the contour structural methods, region-based structural methods decompose shape region into parts which are then used for shape representation and description.

Contour-based approaches are more popular than region-based approaches. Because it is believed that human being discriminates shapes mainly by their contour features. The majority of real world objects have clear contours which are readily available. Therefore, contour methods can easily find applications. Contour based methods usually involve less computation than their region-based counterparts. However, contour shape descriptors are more easily affected by noise and variations than region-based shape descriptors because they use less shape information than region-based methods. Region based methods are usually more robust and application

independent. Generally, every shape technique has its advantages and disadvantages. [13], [20]

2.7 Overview of existing Content-based image retrieval systems

The best-known and representative commercial CBIR system is the Query by Image Content (*QBIC*) system developed by Niblack and colleagues at IBM Almaden Research Center in San Jose. 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 image, or sketching a desired shape on the screen. *QBIC* has been developed further since its initial version and now forms an essential part of IBM's suite of Digital Library products. These aim at providing a complete media-collection management system. An interesting development in the *QBIC* research effort at IBM is the attempt to include grayscale imagery in its domain, a difficult retrieval task. [10]

The other well-known commercial CBIR systems are Visual Information Retrieval (*Virage*) which is used by Alta Visa to facilitate image searching, and *Excalibur* which is adopted by Yahoo! for image searching.

Photobook was one of the representative earlier CBIR system developed by MIT Media Laboratory. This system provides retrieval of textures, 2-D shapes and human faces.

MARS system was developed at University of Illinois by Huang et al. The system characterizes each object within an image by a variety of features, and uses a range of

different similarity measures to compare query and stored objects. User feedback is used to adjust feature weights, and if necessary to invoke different similarity measures.

2.8 Review of Development Tools and Technologies

2.8.1 MySQL

The MySQL database server is the world's most popular open source database, which means that it is free. MYSQL is a relational database management system. MySQL is a small, compact, easy to use database server, ideal for small and medium sized applications. By the way, it is available on a variety of UNIX platforms, Linux, Windows NT, Windows 95/98 and Windows 2000.With more than five million active installations, MySQL has quickly become the core of many high-volume, business-critical applications.

Customers such as **Yahoo!, Google, Cisco, Sabre Holdings, HP** and **NASA** are realizing significant cost savings by using MySQL's high performance, reliable database management software to power large Web sites, business-critical enterprise applications and packaged software applications

2.8.2 Oracle

Oracle Database is an object-relational database. Oracle Database is the most scalable and full-featured database available. Whether driving the web site, packaged applications, data warehouses or OLTP applications, Oracle Database is a foundation technology for any professional computing environment. Oracle Database includes many features to improve data protection. Performance, scalability, and manageability

are basic requirements for business intelligence applications. As in previous database releases, Oracle9*i* offers considerable new enhancements in each of these areas. Oracle9*i*'s partitioning capabilities have been expanded to support list partitioning, and base partitioning capabilities have been extended to cover all data types available in Oracle8*i*, including index organized tables, objects and nested tables. This allows organizations to effectively store manage and search very large amounts of any type of information.

Searching and Indexing

Oracle9*i* builds on the database's already powerful capabilities to search all kinds of content, including text and multimedia. Ultra Search in Oracle9*i* unifies search areas across heterogeneous corporate repositories, websites and groupware content. Ultra Search includes a web interface, web crawling and search administration facilities, as well as a programmable Java API, to provide a unified interface for enterprise and vertical portal search applications.

In order to meet the demands of eBusiness applications, Oracle Text indexing has been improved with a new index type designed to perform very fast search across volumes of short textual descriptions. This is ideal for catalog and metadata search as well as searching of auction data and resumes. With Oracle9*i*, text search of nested XML elements, search attribute values, XPath query syntax, and other advanced XML structures are also supported.

Media and document metadata can now also be extracted, indexed, and mapped to XML documents or database schema through Java APIs to interMedia Annotator. These APIs allow for programmatic invocation of metadata services by any application

or scripting language that can use Java APIs including JavaScript, VBscript, and Apple Script.

2.8.3 Microsoft SQL Server

SQL Server 2000 is a powerful tool for turning information into opportunity. Industry-leading support for XML, enhanced tools for system management and tuning, and exceptional scalability and reliability make SQL Server 2000 the best choice for the agile enterprise.

2.8.4 IBM DB2 Universal Database (UDB) VERSION 8.0

IBM DB2 Universal Database (UDB) reduces the cost of developing e-business applications with its Java support and web connectivity. DB2 provides native, high performance interface to OLE DB data sources. Through this enhanced support, DB2 is a natural engine for e-business in Microsoft environments. DB2 UDB makes use of more features in the Windows to deliver the best performance results.

DB2 Universal Database (DB2 UDB) for UNIX and Windows can help improve the performance of database applications. Many solution developers have already chosen

DB2 UDB as their primary development database environment, and have ported and continue to enable applications to support it in order to take advantage of its unique features.

DB2 UDB is a true cross-platform database management system (DBMS), running on a wide variety of systems including Windows 98, Windows NT, Windows 2000, Solaris, HP-UX, AIX and Linux.

DB2 is a database leader in several technologies, and offers true multi-platform support and scalability. The same database is able to handle mixed types of workloads on a single server. The DB2 design handles varying workloads from high-volume online transaction processing (OLTP) to complex multi-user queries while maintaining an excellent performance.

DB2 support the ability to create user-defined data types. Using DB2 Extenders, rich data types such as audio, video, image and character can be supported. Most DB2 application programs use static SQL to access tables. A static SQL statement is a complete, unchanging statement hand-coded into an application program. It cannot be modified during the program's execution except for changes to the host variables. Static SQL is a powerful and more than adequate for most application. Any SQL statement can be embedded in a program and executed as static SQL.[5],[10],[18],[19]

2.8.4.1 DB2 Supported Programming Interfaces

Several different programming interfaces can use to manage or access DB2 databases.

Embedded SQL

SQL statements that is precompiled before a program is compiled. The SQL statements can be static or dynamic. Using DB2 embedded SQL applications can be coded in the C/C++, COBOL, FORTRAN, Java (SQLJ), and REXX programming languages.

DB2 Call Level Interface (CLI)

A callable SQL interface based on the X/Open CLI specification; compatible with the Microsoft Corporation's Open Database Connectivity (ODBC).

DB2 Application Programming Interfaces (APIs)

APIs that perform database administration tasks, such as create, activate, back up, and restore. DB2 APIs can be called from applications, including embedded SQL and DB2 CLI applications.

Java Development Kit

Tools and environment for develop Java applications and applets. The kit includes driver support for client applications and applets written in Java using JDBC. It also provides support for embedded SQL for Java (SQLJ), Java user-defined functions (UDFs), and Java stored procedures.

Microsoft Visual Basic and Visual C++

Programming environments used to develop applications conforming to Data Access Object (DAO) and Remote Data Object (RDO) specifications, and ActiveX Data Object (ADO) applications that use the Object Linking and Embedding Database (OLE DB) bridge or the IBM OLE DB Provider for DB2.

IBM or third-party tools

Applications can also be developed using tools such as Net.Data, Excel, Perl, PHP, and Open Database Connectivity (ODBC), as well as end-user tools such as Lotus Approach and its programming language, Lotus Script.

The way the application accesses DB2 databases will depend on the type of application you want to develop. For example, a data entry application might choose to embed static SQL statements in the application, application that performs queries over the World Wide Web might choose Net.Data, Perl, or Java.[5]

2.8.4.2 IBM DB2 Audio, Image, Video (AIV) Extenders

DB2 (DB2) Universal Database (UDB) is a powerful, object-relational database manager. It stores and protects traditional numeric and character data, as well as large, complex objects (LOBs). DB2 Extenders help to exploit DB2's object-relational features. The extenders define distinct data types and special functions for image, audio, video, and text objects. By doing this, the Extenders save developers the time and effort of defining these data types and functions in the applications.

DB2 AIV Extenders take the database applications beyond traditional numeric and character data to leverage images, video, voice and complex documents. These data types and functions are available through SQL query. DB2 Extenders make it easy to bring this information together on demand. In addition, the AIV Extenders give the applications new ways to search for information. For example, applications can search for images by their visual characteristics, using visual examples of color or texture.

The AIV Extenders create and use administrative support tables and handles to store and access image, audio, and video data. Here, I concentrate to image extender.

DB2 Image Extender supports a wide variety of image formats, such as GIF, JPEG, BMP and TIFF. With DB2 Image Extender, applications can:

- Import and export images and their attributes into and out of a database. When import an image, DB2 Image Extender stores and maintains image attributes such as size in bytes, format, height, width and number of colors.
- Control access to images with the same level of protection as traditional business data.

- Convert the format of images when importing or exporting or still in its source format. You can also scale an image, rotate it, do black-white image inversion or change representational characteristics such as bits per sample and compression type.
- Secure and recover images. Images and their attributes that you store in a DB2 database are afforded the same security and recovery protection as traditional business data.
- Query images based on related business data or by image attributes. You can search for images based on data that you maintain, such as a name, number or description; or by data that the DB2 AIV Extenders maintains, such as the format of the image or its distribution of colors.
- Generate and display image thumbnails and full images. A thumbnail is a miniature version of an image. When import an image into a database, DB2 Image Extender creates and stores a thumbnail of the image.

With DB2 Image Extender's Query by Image Content (QBIC) technology, images can be search easily. QBIC use visual examples of colors or texture patterns as search criteria. The DB2 Image Extender will find the images whose visual characteristics are closest to the examples that choose.[5], [10],[17],[18],[19]

2.8.5 Data Access Technology

2.8.5.1 Java Database Connectivity (JDBC)

JDBC technology is an API that lets the developer access virtually any tabular data source from the Java programming language. It provides cross-DBMS connectivity to a

wide range of SQL databases, and now, with the new JDBC API, it also provides access to other tabular data sources, such as spreadsheets or flat files. The JDBC API allows developers to take advantage of the Java platform's "Write Once, Run Anywhere" capabilities for industrial strength, cross-platform applications that require access to enterprise data.

2.8.5.2 Open Database Connectivity (ODBC)

Open Database Connectivity (ODBC) is a widely accepted application programming interface (API) for database access. It is based on the Call-Level Interface (CLI) specifications from X/Open and ISO/IEC for database APIs and uses Structured Query Language (SQL) as its database access language.

ODBC is designed for maximum interoperability – that is, the ability of a single application to access different database management systems (DBMSs) with the same source code. Database applications call functions in the ODBC interface, which are implemented in database-specific modules called drivers. The use of drivers isolates applications from database-specific calls in the same way that printer drivers isolate word processing programs from printer-specific commands.

Chapter 3 Methodology

3.1 Introduction

Thoroughly planning is a must in a project for effective development in order to achieve the project goals. Methodology is an organized, documented set of procedures and guidelines for one or more phases of the software life cycle, such as analysis or design. It is usually presented as a series of steps, with techniques and notation associated with each step.

Process model is a very important during the software development process or software life cycle. It can form a common understanding of the activities, resources and constraints involved in software development. When a process model is created, it helps to find the inconstancies, redundancies and omissions in the process. As the problems are noted and corrected, the process becomes more effective and focused on building the final system.

In the development process, several different models for system development have been used, such as Waterfall Model, V model, Waterfall Model with prototyping, Spiral Model, Extreme Programming and etc. These models provide guidance on the order in which a project should carry out its major tasks.

3.2 Waterfall Model with Prototyping

Pfleeger (1991) extended the waterfall model by placing a larger emphasis on the testing component. In this model all steps are interconnected to allow a prototyping approach to be used in conjunction with the model. Every development stage should be completed before another the next begins.

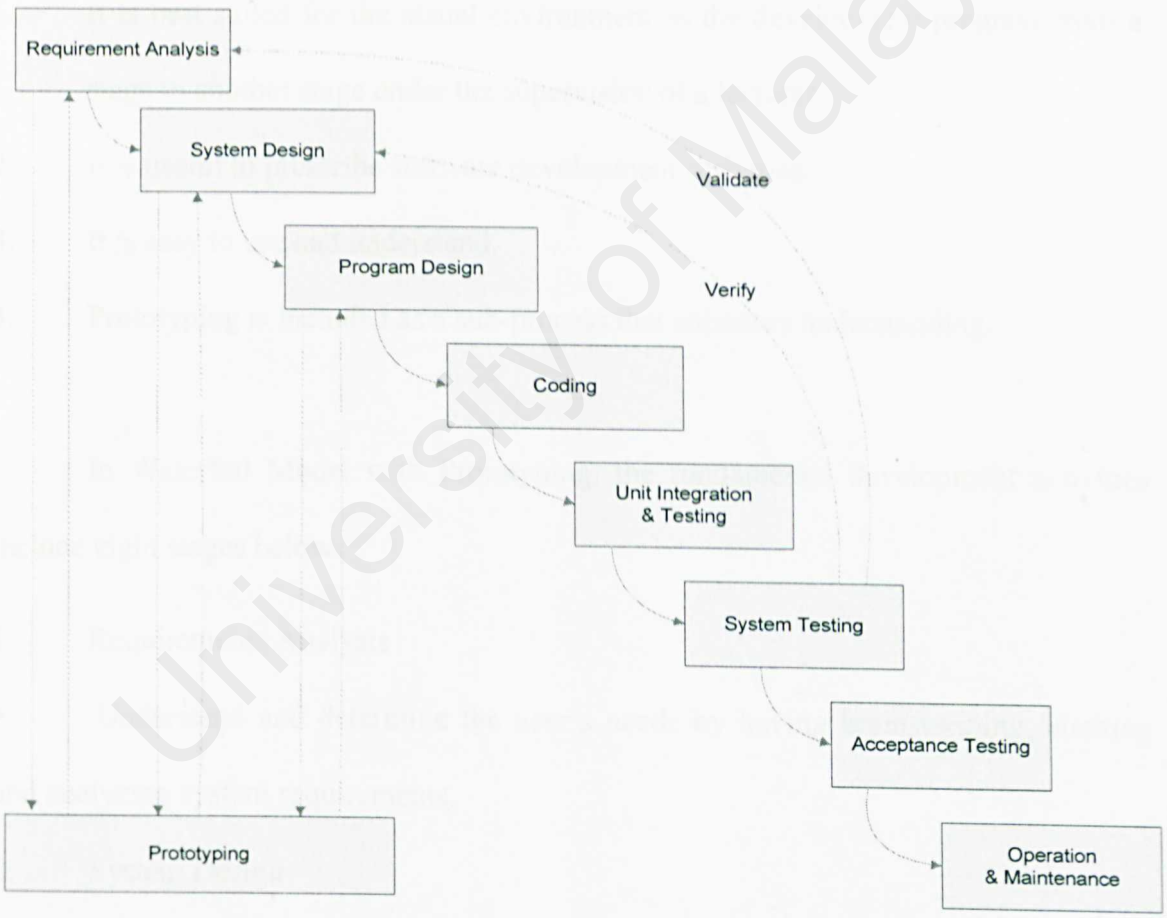


Figure 3.1 Waterfall Model with Prototyping

Waterfall Model is amended with prototyping activities to improve the understanding. Requirements prototyping is to ensure that the requirement are feasible and practical, if not revisions are made at the requirements stage. Design prototyping helps developer asses alternative design strategies and decide which is best for a particular project.

The Waterfall Model with Prototyping has been chosen to model the development of CBIR system. The reasons are that: -

1. It is best suited for the actual environment as the development progress from a stage to another stage under the supervision of a lecturer.
2. It is useful to prescribe software development activities.
3. It is easy to use and understand.
4. Prototyping is included as a sub-process that enhances understanding.

In Waterfall Model with Prototyping, the fundamental development activities include eight stages below:

1. Requirements Analysis

- Understand and determine the user's needs by having brainstorming, eliciting and analyzing system requirements.

2. System Design

- Outlining system functionalities by having feasibility studies and case studies on current systems.
- Determine and specify hardware and software architecture.
- Verify the system design.

3. Program Design

- Determine and specify the program design and database design.

4. Coding

- Involve programming, personal planning, tool acquisition, database development and component level documentation.

5. Unit and Integration Testing

- Test the modules separately and integrate the tested modules.
- Test on the integrated modules.

6. System Testing

- Combine all the integrated modules into a system and test on the system.
- Specify and review the result of system test.
- Evaluate the system to meet the requirements.

7. Acceptance Testing

- Testing on system is completed. The system is delivered.

8. Operation and Maintenance

- Control and maintain the system.

The validation during system testing is to ensure that the system has implemented all the requirements, so that each system function can be tracked back to a particular requirement in the specification. As for the verification, it ensures that each function works correctly.

The aim of prototyping is to enable input from the end user at an early stage by giving them the look and feel of the application. This is achieved by modeling the user

interface while having little or no content behind that interface. Prototyping is especially valuable where requirements cannot be specified clearly.

3.3 Facts finding techniques

In developing this thesis project, information and fact-finding are done in many kind of techniques such as interview, research and etc. The objectives of fact finding are to obtain the great view about project system that is ready to use within requirements.

3.3.1 Documentation and printed media

The main reference resource for this project thesis report writing is past year student's report in the documentation room of FSKTM. These previous years' thesis reports are the fastest way to understand the whole system development layout.

3.3.2 Web searching

Web searching is the wonderful fact finding method. Internet is a platform where a lot of info can be acquired. The existing similar project and software can be easily found via World Wide Web. It is greatly helpful in giving ideas and guidance to develop the features image retrieval besides, the detail information of authoring tools for CBIR system can be found abundantly through the search engine especially Google and Yahoo

3.3.3 Guidance by lecturer

Discussion had been took place with project supervisor to determine the scopes and the limitations of my project. She had spent her time in guiding and assisting the project directly and indirectly of the way to develop this report of system.

3.3.4 Discussion and Internet Forum

Most of the information that gathered in this project is based on interviews or conversation with people concern. The interviews were not strictly structured because they were more like conversations. Besides, discussion also took place within my friends to gather more information about the needs of system. Internet forum also plays an important role in obtaining the ideas.

3.3.5 Research (written materials)

Research has been done by reading some articles, magazines and other textual mediums in order to gather information related to project as a foundation for the discussion in this project. Reading materials provide a lot of information about the internal structure of medical term that are usually used in medical field.

3.4 Functional Requirement

Functional requirement specifies a function that a system or a system's component must be able to perform. These are software requirements that define behaviors of a system, that is, the fundamental process or transformation that software and hardware components of the system perform on inputs to produce expected outputs. Functional requirement are handles while the requirements and analysis workflows are being performed.

As for this CBIR for medical images database system, images retrieval is the major module of this project system. Therefore the system could provide certain types of search criteria and search option to enable users to perform a search. The main functional requirement that will contribute and form the whole system is the abilities to store and retrieve the medical images based on the shape required by the users. At this section, the information required by the users is displayed automatically according to their requirements. Thus, users are only required to choose the given option.

There is another main functional module, which is the Graphical User Interface (GUI) system. The GUI must provide users an easy mean or platform of viewing, editing, and inputting images data besides editing the system's configuration and inputting the user's details.

3.5 Non-Functional Requirement

Non-functional requirement does not describe what a system or software will do or process, but HOW it does. It specifies properties of the target product itself. For example, software performance requirements, some external interface requirements, software design constraints, and software quality attributes. Non-functional requirements are difficult to test; they are usually, or most of the time, evaluated subjectively. Non-functional requirements have been recognized and acknowledged as a vital and crucial determinant to the success of many software projects. To handle certain nonfunctional requirements, detailed knowledge about the target software product may be needed.

For this simulation project, the following requirements have been set:

1. Clarity and Simplicity

The query criteria must be clear and simple.

2. Maintainability

System maintenance always requires more effort and time if the system is not well planned and designed at the beginning. System maintenance is a must for this CBIR system, just like any other systems, as it allows certain changes or modifications to be made over the system.

3. Efficiency

Efficiency is the ability of a process procedure to be called or accessed unlimitedly to produce similar performance outcomes at an acceptable or credible speed. In this CBIR system, efficiency comes into picture as how fast the system can retrieval the medical

images as what is require. It is also regarding how well and fast the system can handle and load the medical images.

4. Flexibility

The CBIR system is a flexible system as it is a standalone system and can be actually executed over many Microsoft and UNIX operating systems.

5. User Friendliness

In this CBIR system, the user interface design aims to fulfill three golden rules of user friendliness, which are:

- Place User in Control

This will define interaction modes in a way that does not force a user into unnecessary or undesired actions or situations. Besides, it also provides flexible interaction, for instance, via mouse movement and keyboard commands.

- Make the interface consistent

The interface design should conform to consistent fashion where all visual information must be organized according to a design standard that is maintained throughout all screen displays. Apart from that, input mechanisms are restricted to limited sets that are used consistently throughout the application.

- Accuracy and Correctness

Accuracy means how close or near an output produced by a system to a desired or perfect output according to theory. Correctness is the degree to which the software performs its required functions. To ensure that this CBIR system meets these requirements, many testing should do to variety shape of medical images.

3.6 Selection of Development Tool

Database management software is now the core of enterprise computing. New ways of working bring new requirements, including digital rights management. For this project, IBM DB2 Universal Database (UDB) version 8.0 will be use to develop the images database system. Query by Image Content (*QBIC*) system developed by Niblack and colleagues at IBM Almaden Research Centre are a technology that permits users to catalog and retrieve images from databases without having to describe them verbally.

IBM has incorporated the *QBIC* technology into two of its commercial products; Ultimedia Manager and DB2 Extensions. The company is also licensing the core search engine to developers and resellers. On the other hands, the combination of IBM's DB2 database foundation along with Java GUI images is very successful.[10]

In addition to scalability and performance, DB2 offers the following advantages:

- . Integrated support for native environments
- . Integrated system management tools and multiplatform tools
- . Self-managing and resource tuning capability
- . Data replication service
- . Integrated Web access
- . Web services applications
- . Integrated support for development environments
- . Data warehousing functionality
- . High Availability support
- IBM Program for Assistance to Developers

DB2 GUI Tools overview

Figure 3.1 shows the different tools that can be invoked from the IBM DB2 Menu.

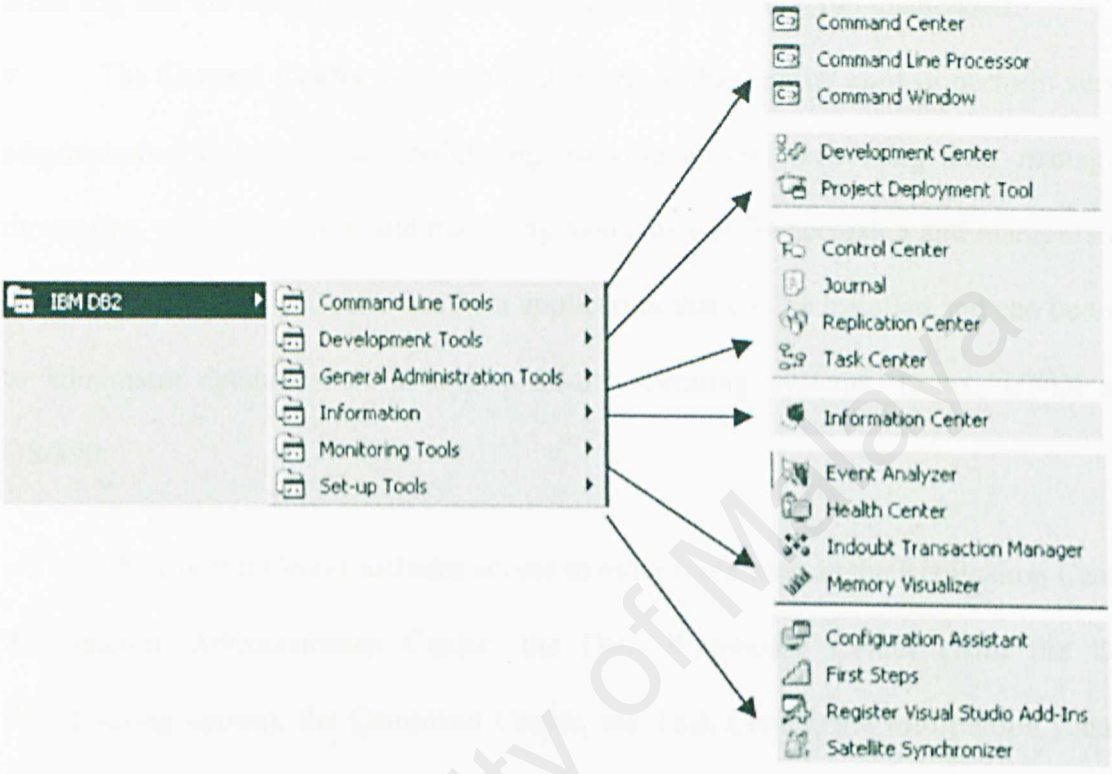


Figure 3.1 IBM DB2 Menus

Here are some of the capabilities provided by DB2 UDB GUI tools:

- The **Command Center** is used to enter DB2 commands or SQL statements in an interactive window, and to see the execution result in a result window. You can scroll through the results and save the output to a file.
- The **Health Monitor** and the **Health Center** help to monitor the health of DB2 systems. They receive alerts about potential system health issues and address those health issues before they become real problems that can affect the system's performance.
- . The **Configuration Assistant** has options to configure both local and remote servers, including DB2 Connect servers, or to create configuration templates.

- The **Journal** is used to view all available information about jobs that are pending execution, executing, or that have completed execution; the recovery history log; the alerts log; and the messages log and also the results of jobs that run unattended.
- . The **Control Center** is a graphical interface that can be used to perform server administrative tasks such as configuring, backing up and recovering data, managing directories, scheduling jobs, and managing media as well as accessing and manipulating databases. The Control Center is a Java application that can be installed and can be used to administer databases on Windows 32-bit operating systems, Linux, UNIX and OS/390.

The Control Center includes access to other tools such as the Replication Center, the Satellite Administration Center, the Data Warehouse Center (with the Data Warehousing option), the Command Center, the Task Center, the Information Catalog Center, the Health Center, the Journal, the License Center, the Developer Center, and the Information Center.[9],[11],[22]

3.7 Hardware and Software Requirements

Hardware includes any physical (that can be touched) device or peripheral that is connected to a computer and is controlled by the computer's microprocessor. In this CBIR system, no special or specific high-end hardware is required or needed. It just requires an IBM-compatible personal computer (PC), which is powerful enough to support Microsoft's operating systems, IBM DB2 Universal Database and Java Development Toolkit (Java 2 SDK v1.4),. Nevertheless, a PC with at least an

Intel or AMD 1.0 GHz processor and 256 MB of RAM, is very much desired and recommended.

Software is a general term for the various kinds of programs used to operate computers and related devices. Software is often divided into system software and application software. System software is usually operating systems that support application software. Meanwhile, application software is programs that do work users are directly interested in.

Microsoft Windows XP Professional or Microsoft Windows 2000 has been chosen as the development platform. The main reason is that most computers in campus and everywhere are using the Microsoft's operating system. However the CBIR system will be implementing in UNIX platform.

The Application Development Tool needed to build and program CBIR system is IBM DB2 Universal Database version 8.0, and DB2 AIV Extenders. The Graphic User Interface will be writing in Java language that is supported by DB2.

Chapter 4 System Analysis and Design

The aim of the system analysis workflow is to obtain a deeper understanding of the requirements and describe those requirements in such a way that the resulting design and implementation are easy to maintain.

A requirement is a feature of a system, or a description of something a system is capable of doing in order to fulfill its purposes. Requirements give a detailed explanation not only the flow of information to and from a system and, the transformation and processing of data by the system, but also the constraints on the system's performance and capabilities. Specifying requirements serves three main objectives:

- Allow developers or programmers to explain their understanding of how users want a system to work and function.
- Tell and instruct designers what functionalities and characteristics a resultant system is to have.
- Tell the developers what to demonstrate to convince the users that a particular system being delivered or developed is indeed as what was needed or ordered.

The design specifications describe the features of a system, the components or element of the system and their appearance to the users. The input to the design process is the specification document, a description of what the product system is to do. The

output is the design document, a description of how the product system is to achieve this.

There are 3 major stages in the design process for CBIR medical images system, which are system hierarchy design, database design and user interface design.

In architecture design, the whole system consists of several components. These components related to each other in certain manners. Their relationship are analyzed and recorded. For the database design, data structures implemented in the system are planned and mention expressly. Finally, the design of user interface takes place. The results are display graphically.

4.1 System Hierarchy

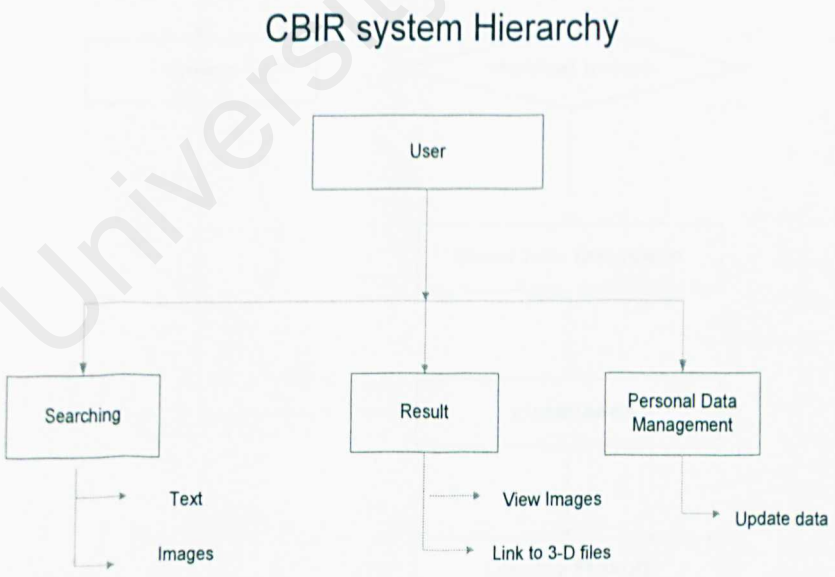


Figure 4.1 System Hierarchy

4.2 Flow Chart of System

Flowchart is one type of process model to show the sequence of processes or operation in program. The flow chart of CBIR system is illustrated in Figure 4.2.

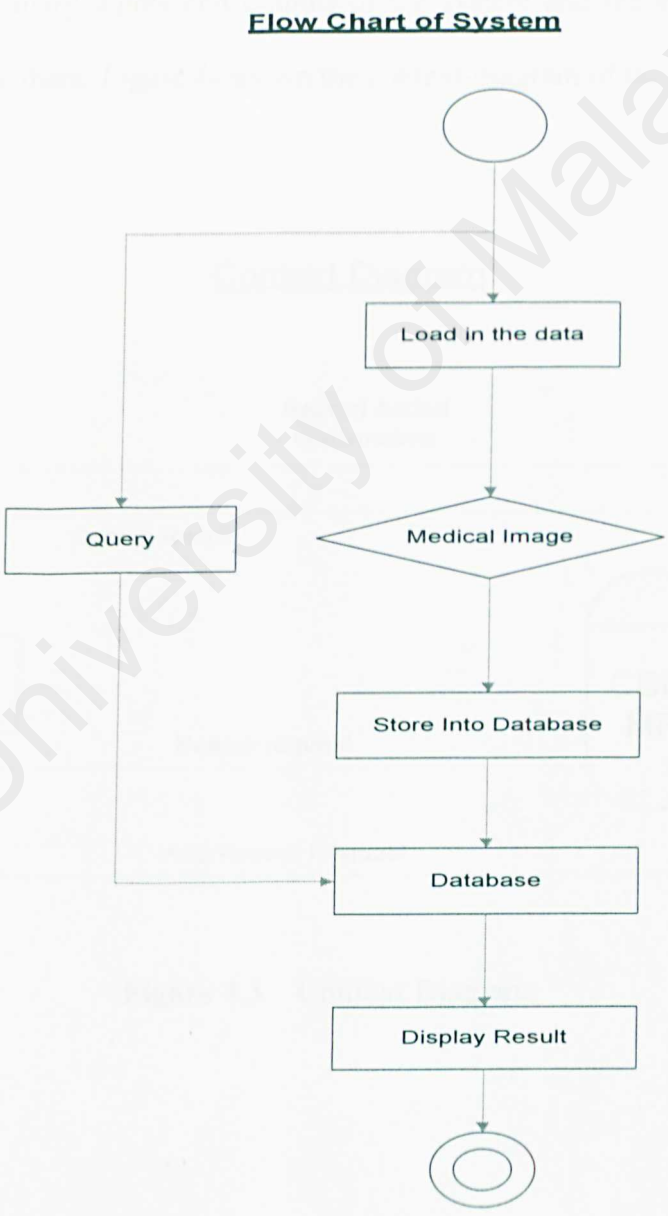


Figure 4.2 Flow chart of CBIR system

4.3 Context Diagram

A context diagram is an excellent source for analyzing actors and finding potential uses cases. We can identify potential use cases by looking at the diagram and identifying the primary inputs and outputs of the system and the external parties that submit and receive them. Figure 4- shown the context-diagram of the CBIR system

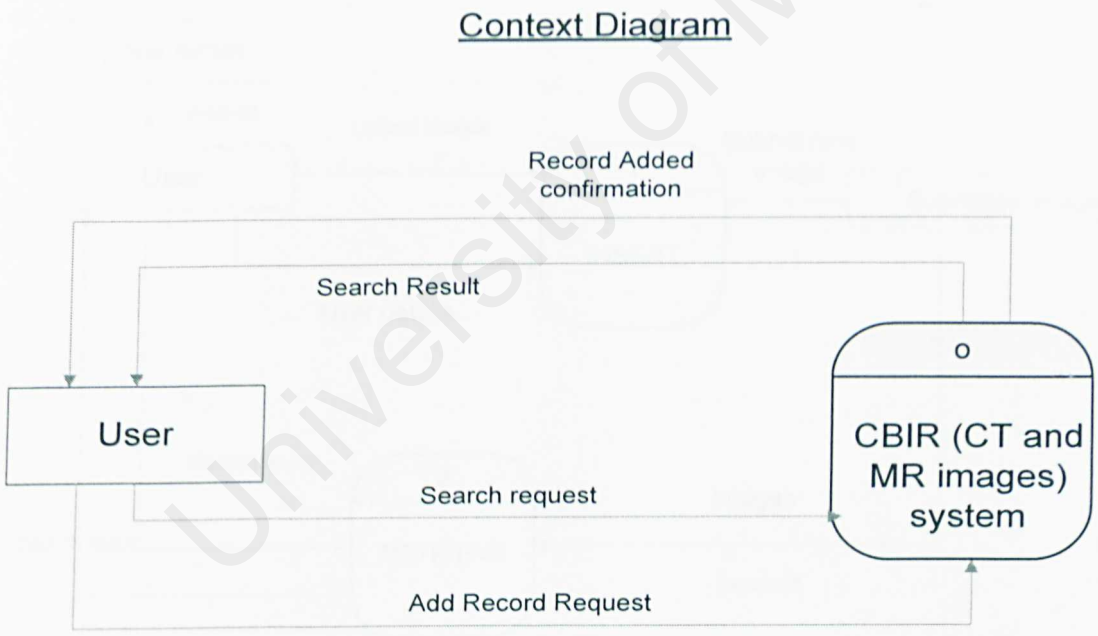


Figure 4.3 Context Diagram

4.4 Data Flow Diagram

A data flow diagram (DFD) is a tool that depicts the flow of data through a system and the work or processing performed by the system.

A simple data flow diagram is illustrated in Figure 4.4. This data flow diagram shows the flow of data through CBIR system.

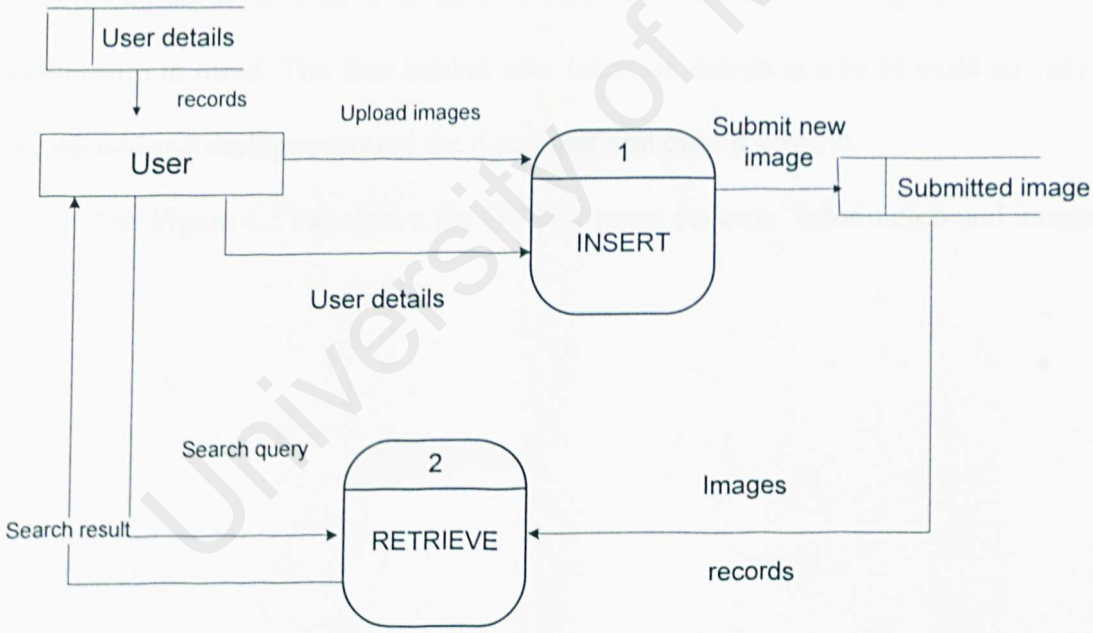


Figure 4.4 Data Flow Diagram

4.5 User Interface Design

Software interface is part of an application that the user sees and interacts with. It acts as an intermediary between the user and the underlying structure, architecture and code that makes the software work (GUI design essentials).

User interface design is not an easy task. The first step is to define the overall look and functions of the system. The quality of the system input determines the quality of system output. It is vital that input forms and screens be designed with this critical relationship in mind. The idea behind user interface design is also to build an easy-to-learn and easy-to-use dialogue around the data input and output screens.

The Figure 4.5 has shown the GUI for insert patients' information and images.

Figure 4.5 GUI for insert patients' information and images

The Figure 4.6 showed the main screen for retrieve database of CBIR system.

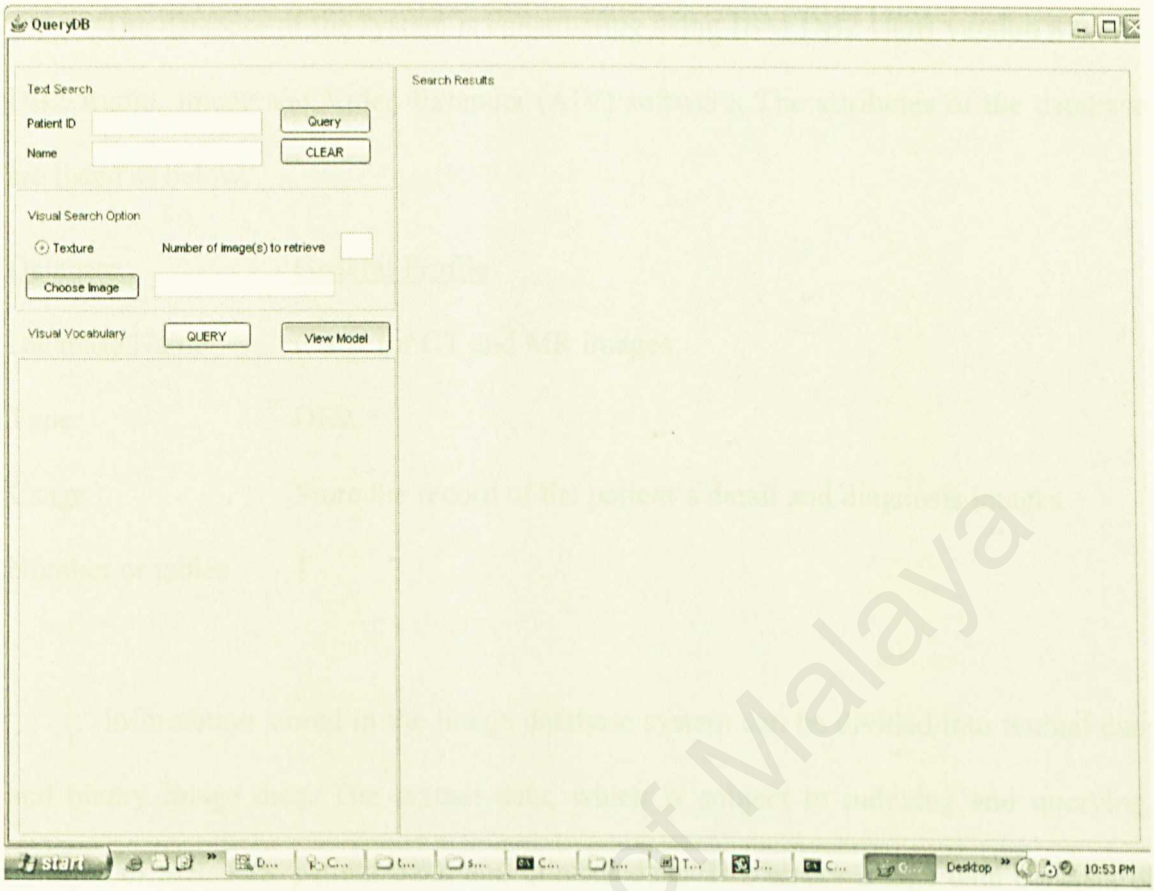


Figure 4.6 GUI for images retrieval

4.6 Database design

The database of the system is constructing using IBM DB2 UDB version 8.0 and DB2 Audio, Image and Video Extender (AIV) software. The attributes of the database are listed as below.

Database	General Profile
Database Name	CBIR for CT and MR images
Type	DB2
Usage	Store the record of the patient's detail and diagnosis images
Number of tables	1

Information stored in the image database system can be divided into textual data and binary image data. The textual data, which is subject to indexing and querying, consists of metadata, patient data, and clinical data. The binary image data consists of MRI and CT images.

Database primarily organized the textual data and the binary image data by patient. Each patient is assigned a unique subject ID number to avoid a situation where a patient is assigned multiple patient record numbers.

4.6.1 Data dictionary

The table used in CBIR system database is as follow.

Patient table

Field Name	Data Type	Length	Description
PatientID	CHAR not null primary key	10	Unique User ID
Name	LONG VARCHAR	30	Patient's name
Age	CHAR	4	Age of patient
Race	VARCHAR	12	Race of patient
CoronalVIEW	mmdbsys.db2image		
SagittalVIEW	mmdbsys.db2image		
TransverseVIEW	mmdbsys.db2image		

Table 4.1 Patient detail

4.6.2 Simple E-R Diagram

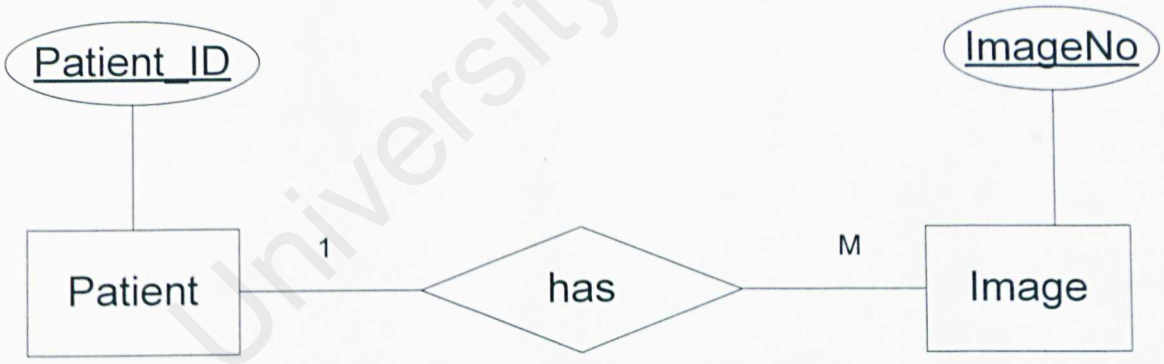


Figure 4.8 E-R Diagram

4.6.3 E-R Diagram of Images Database

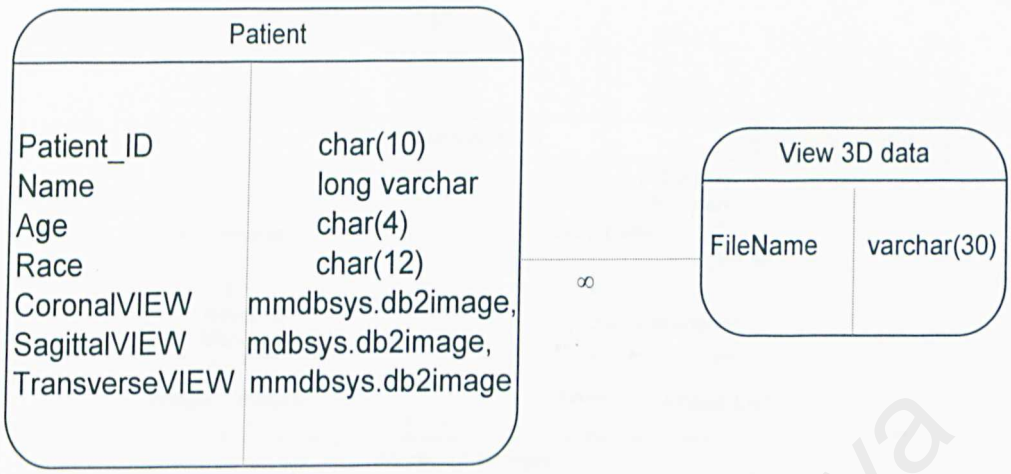


Figure 4.9 E-R Diagram of Image Database

4.7 System Architecture

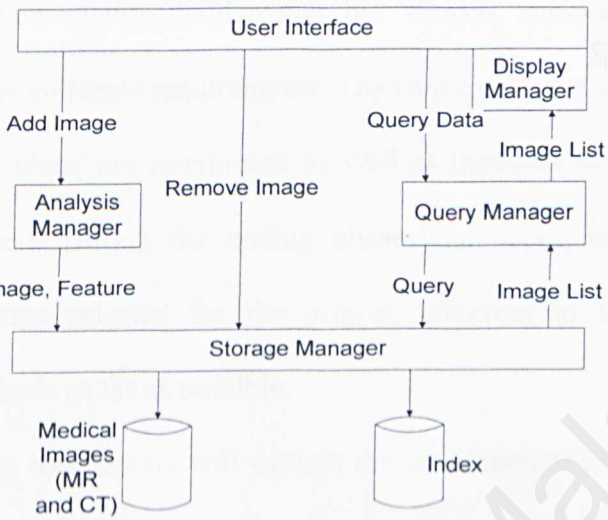


Figure 5.0 System Architecture

4.8 Expected Outcome

The expected outcome is an images retrieval and monitoring system that provides facilities on searching images. Hopefully the results are sufficient and match user requirements.

Chapter 5 System Implementation

System implementation implements the various components of the system (coding) based on the collected requirements. The conceptual and technical designs from the system analysis phase are interpreted as well as modeled to become the physical working system itself. During the coding phase, each component is coded in the programming language selected for the project, adhering to a set of coding and documentation standards as far as possible.

The following subchapters will explain the development environment as well as the development of the system itself, system coding, coding methodology and the coding style and approach applied in the CBIR for CT and MRI system.

5.1 Development Environment

CBIR for CT and MR images system was developed modularly using top-down approach which involves implementing the high-level and software modules that were of priority. These are then further refined into functions and procedures otherwise known as stepwise refinement.

- Before any functionality can be provided, the skeletal system consisting of the hardware and software must be made available before any work is to proceed. From there onwards, starting with a list of all the functions to be provided by the system, I have derived a prioritized list. Given this, the time and resources of

each function were estimated, which not only includes those with the highest priorities.

5.1.1 Hardware in the Development Environment

The hardware configured for the development environment is the underlying element of the whole system. The hardware used in the system implementation phase plays an important role in realizing the final system architecture.

The hardware configuration of the development environment is as below:

- Intel Pentium IV Processor 1.8GHz.
- Memory – 256MB
- Storage – 3 GB of Hard disk space is reserved.
- Other standard desktop PC component.

5.1.2 Software in the Development Environment

Hardware and software form a tightly coupled cohesion that operates in unison to performed programmed tasks. Without software, the fastest, biggest or the most powerful computer will also be inoperative and idling in the corner.

The software tools utilized in the development environment are listed as follow:

- Operating System – Microsoft Windows XP Professional Service Pack 1.
- Java 2 Development Kit with Java Run Time (jdk1.5.0_01).
- Xinox Software's JCreator LE v2.50.
- Notepad
- Documentation – Microsoft Office XP.

- IBM DB2 UDB version 8.0 and DB2 Audio, Image and Video Extender(AIV) software

5.2 System Development

Development of the CBIR system began with acquisition of knowledge and experience with the Java programming language. The following subchapters will detail the explanation of object oriented programming (OOP) approach, classes that are defined and created for the simulation system and other related coding parts of the entire system.

5.2.1 Object Oriented Programming (OOP)

To be able to use Java as the programming language to code or to develop a program or a system, one needs to gain an understanding of the concepts of object oriented. Understanding of what is an object; need to understand core concepts behind object-oriented programming: objects, messages and classes. The import thing is to learn and practice in how these concepts translate into code.

Object - In definition, an object is a software bundle of related variables and methods. "Objects" is a key to understanding object-oriented technology. Software objects are often used to model real-world objects found in everyday life.

Message - Software objects interact and communicate with each other using messages.

Class - A class is a blueprint or prototype that defines the variables and the methods common to all objects of a certain kind.

Interface - An interface is a contract in the form of a collection of method and constant declarations. An interface defines a protocol of behavior that can be implemented by any class anywhere in the class hierarchy. Interface defines a set of methods but does not implement them. A class that implements the interface agrees to implement all the methods defined in the interface, thereby agreeing to certain behavior.

5.3 System Coding

After researches and studies have been done, then start to code the CBIR system using the Java programming language, and to be able to run the system as a standalone windows application and can be executed using Java 2 Runtime environment. The coding phase was done using the Xinox Software's JCreator Light Edition (LE) v2.50 integrated development environment.

The CBIR system utilizes Java 2 Platform package. The graphical user interfaces (GUIs) is created using JFC/Swing. JFC is short for Java Foundation Classes, which encompass a group of features for building GUIs and adding rich graphics functionality and interactivity to Java applications. JFC was first announced at the 1997 JavaOne developer conference. It is defined as containing the features shown in the table below.

Features of the Java Foundation Classes	
Feature	**Description**
Swing GUI Components	Includes everything from buttons to split panes to tables.
Pluggable Look-and-Feel Support	Gives any program that uses Swing components a choice of look and feel. For example, the same program can use either the Java or the Windows look and feel. Many more look-and-feel packages are available from various sources. As of v1.4.2, the Java platform supports the GTK+ look and feel, which makes hundreds of existing look and feels available to Swing programs.
Accessibility API	Enables assistive technologies, such as screen readers and Braille displays, to get information from the user interface.
Java 2D API	Enables developers to easily incorporate high-quality 2D graphics, text, and images in applications and applets. Java 2D includes extensive APIs for generating and sending high-quality output to printing devices.
Drag-and-Drop Support	Provides the ability to drag and drop between Java applications and native applications.
Internationalization	Allows developers to build applications that can interact with users worldwide in their own languages and cultural conventions. With the input method framework developers can build applications that accept text in languages that use thousands of different characters, such as Japanese, Chinese, or Korean.

Table 5.1 Features of the Java Foundation Classes

Refer to Appendix A for full source code of the CBIR for MR and CT system.

5.4 Coding Style

5.4.1 Formatting and Indenting Codes

Formatting and indenting codes is constantly associated to good coding practice. A code that is written without proper formatting or indenting will function or what as well as a formatted code. However, this can make exceptionally difficult to see where an error is coming from. Indentation principally makes the structure of the code stand out and

easier to be read. This eventually will help in detecting and removing the common programming errors.

JCreator provides user a good formatting and indenting facility where user is associated to format and indent codes automatically in the environment while coding a Java source.

5.4.2 Commenting Codes

Comment is not part of the program code and it does not command any program execution. However, comments will slow down the program execution because the compiler has to read and then skip the comment lines each time.

In spite of such shortcoming, comments are still applied as part of common practice in documenting the system's coding. And indeed it is a good and recommended practice. Commenting will help the author or the reader of the code to understand what and why the coding was written. In addition, this also makes it easier for others especially collaborating programmers to understand the coding.

Comments are usually included before or at the side of each block of the code describing it purposes. In the Java programming language, the // is used as the prefix of a single line comment while /* and */ are used as prefix and postfix of a multiple lines comment.

Figure below shows the formatted, indented and commented code in the JCreator development environment:

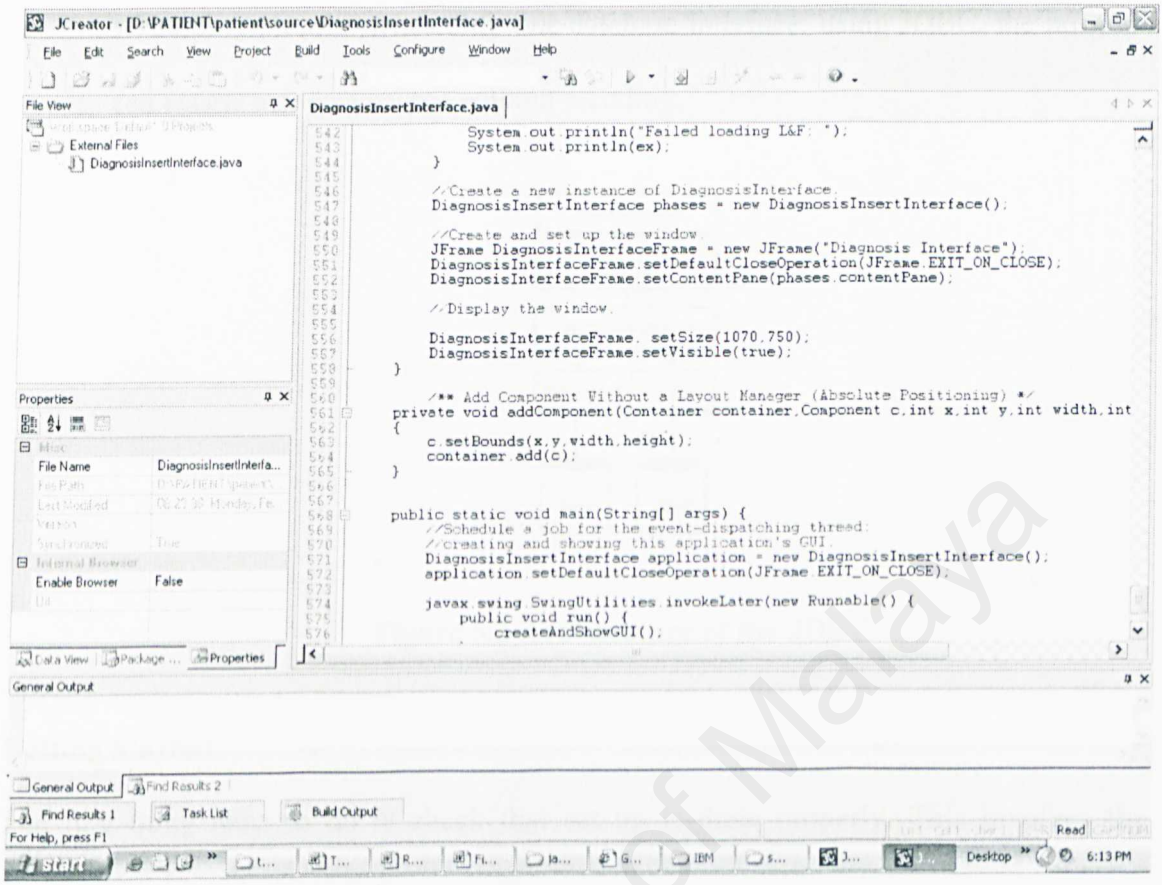


Figure 5.1 Commented code in JCreator

5.4.3 Implementing the DB2 Universal Database

5.3.3.1 Setting up the Database

The CBIR system uses JDBC to manage and control database access. The CBIR database contains a table which stores data for the application. JDBC was designed to keep simple things simple. This means that the JDBC API makes everyday database tasks, like simple SELECT statements. The structure of the JDBC includes these key concepts:

• The goal of the JDBC is a DBMS independent interface, a "generic SQL database access framework," and a uniform interface to different data sources.

• The programmer writes only *one* database interface; using JDBC, the program can access any data source without recoding.

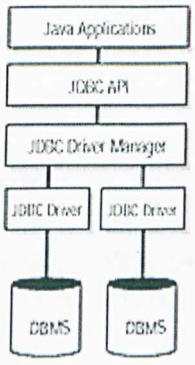


Figure 5.2 Architecture of the JDBC

Getting Started

The first thing need to do is check that set up is done properly. This involves the following steps:

1. Install Java and JDBC on machine.

   To install both the Java *tm* platform and the JDBC API, simply follow the instructions for downloading the latest release of the JDK *tm* (Java Development Kit *tm*). When install the DB2 UDB, you will get the JDBC and Java JDK in the java folder as well.

2. Install a driver on your machine.

   For JDBC drivers written for specific DBMSs, such as DB2, installation consists of just copying the driver onto your machine; there is no special configuration needed.

5.4.3.2 Establishing a Connection

In order to connect to DB2 database, the first thing need to do is establish a connection with the DBMS. This involves two steps: (1) loading the driver and (2) making the connection.

Step 1 Loading Drivers

Loading the driver or drivers that want to use is very simple and involves just one line of code. If, for example, in order to use the JDBC driver in the DB2, the following code will load it:

```
Class.forName("COM.ibm.db2.jdbc.app.DB2Driver");//line 154 in Util.java
```

User does not need to create an instance of a driver and register it with the `DriverManager` because calling `Class.forName` will do that automatically. If you were to create your own instance, you would be creating an unnecessary duplicate, but it would do no harm.

When a driver has loaded, it is available for making a connection with a DBMS.

Step 2 Making the Connection

The second step in establishing a connection is to have the appropriate driver connect to the DBMS. The following line of code illustrates the general idea:

```
Connection con = DriverManager.getConnection(url, myLogin, myPassword);
```

The GUIs of the CBIR system is connected to DB2 UDB using a JDBC driver developed by a third party – IBM DB2, which not belongs to Java itself. Therefore need to put subprotocol after `jdbc:` in the JDBC URL. For example, if the driver developer has registered the name acme as the subprotocol, the first and second parts of the JDBC URL will be `jdbc:acme`. In the DB2, the subprotocol name is db2.

In the CBIR system which using JDBC to access a JDBC data source called "Medical", the JDBC URL could be jdbc: db2: medical. For this system, I did't place the login name and password, therefore, it is null in place of "myLogin" and "myPassword" for the DBMS. The following two lines of code will establish a connection:

```
public static String jdbcURL = "jdbc:db2:medical";//line 14 - Util.java
conn = DriverManager.getConnection(jdbcURL); //line 155 - Util.java
```

If the driver which loaded recognizes the JDBC URL supplied to the method DriverManager.getConnection, that driver will establish a connection to the DBMS specified in the JDBC URL. The DriverManager class, true to its name, manages all of the details of establishing the connection for you behind the scenes.

The connection returned by the method DriverManager.getConnection is an open connection you can use to create JDBC statements that pass your SQL statements to the DBMS. In the code above, con is an open connection.

Creating database and table

DB2 UDB provides an easy way to create database and table. These were created using the command line processor (CLP) which is shown below.

```
db2 => db2start
SQL1026N  The database manager is active.
db2 => create database medical
DB20000I  The CREATE DATABASE command completed successfully.
db2 => db2 => create table patient(patientID char(10) not null primary key, Name long
varchar,age char(4), Race char(12), CoronalVIEW mmdbsys.db2image, SagittalVIEW
mmdbsys.db2image,TransverseVIEW mmdbsys.db2image)
DB20000I  The SQL command completed successfully.
db2 =>
```

The further procedure to enable database for db2image is show in Appendix B

Inserting Data Using Parameters

The system use SQL statements such as INSERT to insert data into table. First, create a stored procedure and then call the stored procedure from within the code supply the necessary values if parameters are needed. The following example illustrates how to insert a new user in a users table.

```
String sql = "INSERT INTO " + patientTable + "(PATIENTID,coronalview";

                if (sagittalview != null) sql += ",sagittalview";
                if (transverseview != null) sql += ",transverseview";
                if (name != null) sql += ",name";
                if (age != null) sql += ",age";
                if (race != null) sql += ",race";
// code in ProcessPatientEntry.java
```

Retrieve data from table PATIENT

The follow lines of code show part of SQL language to retrieve image data from the database. Also notice that have called the `getDBConnection()` method of the Connection object.

```
conn = getDBConnection();

sql = "SELECT "          + "Thumbnail(" + imageColumn +")"
                         + ",Comment(" + imageColumn +")"
                         + ",Filename(" + imageColumn +")"
                         + ",QBScoreFromStr(" + qbicQuery + ","
                         + imageColumn + ") AS SCORE" +
                         " FROM " + tableName +
                         " ORDER BY SCORE"
                         ;
       Util.log(this, "executing query: " + sql);
       rs = stmt.executeQuery(sql);
       displayScoredImages(rs);

// code in QueryDB.java
```

Chapter 6 System Testing

Testing is an activity that must be carried out throughout the system development life cycle. It is an integral component of the software. Testing done only at the end of a product is a recipe for disaster as it will be hard to pin point the exact error location. In reality, testing actually takes place at each phase, from requirements gathering to analysis to implementation. During the requirement phase, the requirements must be checked; design phase requires careful checking as every other phase. During implementation, each module must be tested and the final product as a whole needs testing at the integration phase, as it tells whether the coding of the system is successfully implemented, whether the executing system visualizes accurately and according to the algorithm and whether the code needed to be modified, enhanced, deleted, added or debugged.

6.1 Testing Strategy

Five test strategies will be use to determine the level of success of CBIR system. The five strategies are:

- Unit Testing
- Data Validation Testing
- Integration Testing
- System Testing
- Performance testing

Unit Testing

Unit testing concentrates on the smallest unit of software design – the module. It consists of testing an individual component before it is combined with other components. The test criterion is each statement should be executed at least once, and that every possible outcome of each branch or loop should be tested as well. In addition, checks are made to the internal data structures, logic and boundary conditions for input and output data.

In the development of CBIR, tests are performed to each button and link to ensure that it functions as required, every condition statement is feed with all possible combination of data so that none are left out and appropriate actions can be taken in the course of erroneous input data, links are correctly linked to others, and the functionality of each module are tested.

For example, insert and retrieve module tested under this strategy for its functionality of inserting and retrieving with correctly.

Data Validation Testing

This testing is used to test the input data such as input length, input data type as well as required field. For example, the PatientID field must have 8 characters and all the data and images must key in or upload correctly in order to perform the insert function.

Integration Testing

Integration testing involves combining tested modules into more complex groupings of modules, and testing these groupings until the whole software system has been put

together and the interfaces tested. Testing performed at this level is usually black-box testing. This testing activity can be considered as testing of the design, and hence the emphasis on testing modules interactions.

The integration testing of the CBIR system can be viewed from an incremental perspective, whereby tests started from the insert function which is written in the ProcessPatientEnty.java. Using this paradigm, the DiagnosisInsertInterface.java GUIs is built piece-by-piece, and tested little-by-little which is integrated and using some code written in ProcessPatientEnty.java and PatientData.java

```
private void processSummitAction() {
                                :
        ProcessPatientEntry test = new ProcessPatientEntry();
                                :
        PatientData entry = null;
        entry = new PatientData();
                                :
}
```

After integrated the GUIs with ProcessPatientEntry.java, the data and images can be insert using GUIs rather than using source code before in coding and testing phase. Actually, there have many integration testing in order to run the system, but it is time consuming to write down in the plain text.

For the CBIR, this test is also used to ensure all modules are connected and linked according to plan to uncover errors associated with interfacing.

System Testing

All software packages, custom-built programs and any existing programs that comprise the new system must be tested to ensure they all work together. This task normally involves analyst, owners, users and builders. [SAD Mcgraw Hill]

System testing is the process of testing an integrated hardware and software system to verify that the system meets the specified requirements as described in the requirements specification. A system test is a series of different test designed to fully exercise the system to uncover its limitations and measure its capabilities. Because system testing takes place at a higher level, the testing focuses on behavior rather than function or functional structure. This include compile and debug the system.

i) Function Testing

During system testing of CBIR, functional testing was carried out to determine that the system performs the functions as described in the requirements specification.

Further testing of the system includes exercising all possible triggers and the expected results compared with the actual results, tracking through the software system to ascertain that they function correctly in sequence from input to output. Subsequently, every error message which can be generated by the system is tested for appropriateness and understandability from the user's perspective, and examples used in the user's manual are tested for correctness. All the functionalities are available throughout the manual, and through an effective and accurate index.

ii) Compiling and Executing

Once the coding of the system is completed, where all the classes designed are fully coded, the Java source code needs to be compiled to see whether there are any bugs or errors in the coding. If the Java application can be successfully compiled without any error, then the detail testing phase can proceed to executing the application. Otherwise, if the Java application is compiled with errors, the testing phase needs to jump to the debugging phase, before it is recompiled again.

iii) Debugging

Once the application is compiled with errors, the error messages need to be scrutinized to identify where the errors have occurred in the source code. The errors might caused by syntax mistakes, such as the left out of semi-colons, curly braces and other symbols used in the Java programming language. Some errors might also caused by logical errors such as errors in referencing, errors in calling methods, or errors in passing arguments.

The process of debugging is to check on those mistakes and correct them. It is s a process of eliminating errors or bugs from the source code in order for the system to compile successfully.

The compiling-debugging process is an iterative process, which is very common and normal in system programming. Only a successfully compiled piece of source code is able to proceed to the execution process in the testing. Therefore compiling-debugging the source code is a tedious process.

Performance Testing

Performance testing is phase to see if the performance of the system meets the required specifications i.e. the non-functional requirements. Several testing that were conducted is as followed.

- **Usability**
  - Usability is based on building user interfaces that have patterns already familiar to the typical user. Thus, the usability of the system is tested first because if the system is not easy to use, it will not be used at all or it will be used incorrectly and was done with end users.
- **Reliability**
  - Reliability is the probability of some function of the system failing within a specified time. Reliability tests are conducted to ensure that the system can be used for periods of time before a fault is detected. Testing is conducted in terms of whether for a given same input, does the system perform in exactly the same way, and does a change in various parameters still provide the same expected outputs.

Chapter 7 Discussion and Conclusion

Overall I would classify this final year project as a success and insightful. Through this project, I have gained a lot of knowledge and exposure regarding the findings, researches and the coming of a new era in human life, and more specifically the computing and CBIR world. CBIR is an interesting, challenging and brain-quenching altogether.

Nevertheless, the outcomes will greatly bring a tremendous revolution to the working and daily lifestyle of mankind. Processing of tasks, whether they are now done electronically or manually, will definitely see a huge advancement and leap in speed and accuracy once CBIR based on shape feature can be successfully implemented.

There are still a lot of works, endeavors, struggles and barriers to break and go through to bring the theories and hypotheses into real applications, which the image retrieval can be more accurate.

7.1 Problems Encountered and Recommended Solutions

The success of the CBIR system does not come without problems and constraints. Much effort had been placed into understanding, eradicating, and solving problems encountered through the whole course of the system development.

7.1.1 Unfamiliarity with Setting and Configuration of Environment

During the initial stages of implementation, setting up of the database server consumed a lot of time, due to lack of experience. Being a first-hand in this matter, the installation and configuration were met with many failures. Discussion with peers and resources from the internet were sourced repeatedly before the desired results are achieved.

7.1.2 Unfamiliarity with programming language or poor mastery

The seriously problems faced in the system development were to understanding the working process of DB2 database, java language and SQL statement.

Because I have no much knowledge in the Java programming language before the implementation of the system started, I needed to get a deep detailed idea of object oriented programming in Java before coming out with the architectural basis design of the CBIR system. After that, there still need some research and coding in order to make correct and success communication between java language, SQL statement and DB2.

Objects of the system were to be identified and verified of their applicability and flexibility to be coded. Once the coding of the system started, I encountered a lot of syntax and logical flow errors, which consequently, forced me to spend a huge amount of time in debugging, referring to documentation, ask friends and even seeking help from experts in forums.

Last but not least, after the final code was successfully compiled and executed, I was taken aback by the non-aligned and jumbled up arrangement of nodes, links and finally the entire system. Works to align and position nodes properly and tidily to better visualize the coding were carried out.

7.1.3 Limited resource and knowledge in the field CBIR

Through the internet, I found that there have many existing system developed using CBIR method with apply the color, shape, texture method, but there can't found source code used or the results are not accurate.

Through the surfing net, I found eVe SDK on www.evisionglobal.com which is suitable to develop this CBIR system. Unfortunately, I failed to download this SDK which cause by the webpage error. Furthermore, I found this site quite late which the deadline of this system is coming.

The eVe (eVision Visual Engine) Software Developers Kit (SDK) Software Developers Kit (SDK) is a programming toolkit for building image analysis and visual search applications for the recognition and retrieval of images. The components include Java, C++, and COM class libraries, sample programs with source code, sample images, and reference documentation.

Developers and software engineers can use eVe as a building block for developing a visual search application.it is not an end-user application in itself. eVe provides an intelligent infrastructure to acquire, process, and manage visual assets while assisting the development of natural and intuitive user interfaces that perform searches in the same way that people think.

Some example applications using eVe are:

Online shopping. Locating a specific product within an e-Commerce site can sometimes be a difficult task. For example, to locate a particular type of shirt on a clothing site, a user must first specify the type, purpose, color, and material of that shirt using text-based prompts and menu listings. After clicking through and choosing

options from these numerous prompts and menus, the user is then presented with a number of unsorted shirt images from which to choose. Utilizing eVe.s Visual Search technology, you can enable a user to find the desired shirt using a more visually intuitive interface that requires fewer clicks and produces faster results.

Medical imaging. Medical imaging is critical to patient diagnosis and care across a broad range of health care procedures and disease states. For example, to locate radiology images from different patients who share common symptoms is difficult, if not impossible. Utilizing eVe.s Visual Search technology, you can enable a medical professional to quickly find visually similar radiology images from different patients to help with diagnosis.

If this SDK can be used to develop this CBIR system, the result could be different.

7.2 System Strengths

The system has a view model function which can view STL (Stereolithography) and 3D CAD models using the free ModelPress Reader.

The free ModelPress Reader views the highly-compressed ISF and 3DF files created by ModelPress_Publisher or MYRIAD. It also views VRML, STL, HSF and ICS formatted files.

ModelPress Reader can be a standalone desktop application or be embedded in another desktop application; or, it can be embedded in custom Web pages to be a Web-delivered control, designed to work with Microsoft Internet Explorer.

ModelPress does not use an application server, so scalability is not an issue. The published file sizes are so manageable, formerly huge models begin to display in just seconds. Because ModelPress does not require a live session server, ISF and 3DF files can be downloaded or e-mailed and viewed off-line, just as with other popular Web publishing formats. Through this ModelPress Reader, the model can be save in JPEG format and insert into database of the system.

7.3 System Limitations and Future Enhancement

Although the CBIR system project is rated, overall, as a success, it still comes with a number of limitations, and there are still a lot of areas which can be further enhanced.

The system was developed utilizing the Java Swing. There have much more Swing Component which can be implemented. Actually there have scroll bars for the image panes, but after implement all the required component together, the scroll bars can't function. Future enhancement for this constraint, by using other add component method, is to place a vertical and a horizontal scroll bars on the image panel. This will enable the user to scroll left and right, up and down to see the images being upload if size of the images bigger than the image panes.

Besides placing scroll bars, the java GUIs of the system only support image with JPEG format. This can be enhance if have much more time to find the expert solution to convert the image's format.

There have another limitation which the system can just insert 3 images simultaneously which each image is insert into different column in the patient table.

Last but not least, although the completed system is just an insert and retrieves image system; it will be a great achievement in human history to be able to put the content-based image retrieval algorithm into real world application, such as web search. Again, all these are just brilliant ideas, there are much more effort, research and development to be funded and carried out to see the reality.

7.4 Conclusion

CBIR for MR and CT images has fulfilled its objectives and requirements as stated during the findings of the Analysis Phase.

Future enhancements can be made to the system to ensure the application provides a better integrated solution for the interface and retrieve method.

Nevertheless, knowledge and experienced were gained in the course of developing the system from initial planning to system evaluation. Concepts such as JDBC architecture, system development methodologies, etc were captured in the course of literature reviews, and programming skills was put to good use. This project has open the pathway for improvement in communication skills, project management and other personal building skills such as self-management and more learned-oriented.

References

Websites

1. "Anatomical Planes - Physio-nethttp" <http://www.physio-net.com/reference/glossary/misc/anatomicalplanes.htm> (27 Seprember 2004).

2. "Basic Anatomic Terms". <http://www.footmaxx.com/clinicians/anatomic.html> (27 September 2004).

3. "Content-based image retrieval", 23 January 2004. <http://www.unn.ac.uk/iidr/CBIR/cbir.html> (06 July 2004).

4. "Design Considerations for Medical ImageArchive System". <http://hpcio.cit.nih.gov/file/repository.pdf>(4 July 2004).

5. "Image, Audio, and Video Extenders Administration and Programming". <http://csbackup.it.bton.ac.uk/db2/dmba6/htqry.htm> (06 July 2004).

6. "MRI" <http://www.mayfieldclinic.com/PE/PE-MRI.HTM>(30 June 2004)

7. "Scanning of the Body (Computed Tomography scan)", 17 June 2004. <http://www.radiologyinfo.com/content/ct_of_the_body.htm>(30June 2004).

8. "Virtual Hospital: Diagnostic Radiology: Magnetic Resonance Imaging (MRI)". <http://www.vh.org/adult/patient/radiology/magneticresonanceimaging/index.html > (8 August 2004).

9. "Wireless Developer Network - Java GUI Tutorial". <http://www.wirelessdevnet.com/channels/java/training/javagui.html> (28 July 2004).

10. <http://wwwqbic.almaden.ibm.com/> (06 July 2004).

11. Andrew Binstock, "Quick Tour of IBM DB2", 23 October 2003. (<http://www.devx.com/ibm/Article/17647> (10 September 2004).

12. Hemant D.Tagare, PhD, C. Carl Jaffe, MD, and James Duncan, PhDMedical, "Image Databases A Content-based Retrieval Approach", 21January, 1997. (<http://www.pubmedcentral.gov/articlerender.fcgi?pubmedid=9147338> (26 July 2004).

13. Henning Müller, Nicolas Michoux , David Bandon and Antoine Geissbuhler, "ScienceDirect - International Journal of Medical Informatics : A review of content-based image retrieval systems in medical applications—clinical benefits and future directions," 7 February 2004. <http://www.sciencedirect.com/science> (29/7/2004).

14. John P Eakins and Margaret E Graham, "Northumbria Image Data Rresearch Insititute", January 1999. <http://www.unn.ac.uk/iidr/research/cbir/report.html > (4 July 2004).

15. K. L. Chelule, Dr. T. Coole and D.G. Cheshire, "Fabrication of Medical Models from scan data via Rapid Prototyping Techniques". <http://www.deskartes.com/news/fabrication_of_medical_models_fr.htm> (28 August 2004).

16. L. Rodney Long, S. Antani, D. J. Lee Daniel M. Krainak, George R.Thoma, "CEB - Biomedical Information from a National Collection of Spine X-rays: Film to Content-based Retrieval", 07 April 2003. <http://archive.nlm.nih.gov/pubs/long/spie-sd2003/spie-sd2003.php> (26 August 2004).

17. Raul F. Chong, "An Introduction to DB2 UDB Express GUI tools (Part 1)", 17 July 2003.

<http://www106.ibm.com/developerworks/db2/library/techarticle/0307chong/0307chong.html> (30 June 2004).

18. Ryan Cox, "Developing Java applications using DB2 Image and Audio Extenders (Part 1)," 12 November 2001. <http://www-106.ibm.com/developerworks/db2/library/techarticle/cox/0201cox.html> (06 July 2004).

19. Ryan Cox, "Developing Java applications using DB2 Image and Audio Extenders (Part 2)," 12 November 2001. <http://www-106.ibm.com/developerworks/db2/library/techarticle/0202cox/0202cox.html> (06 July 2004).

20. Sean Landis, "Sean Landis' Fall 718 Context-Based Image Retrieval Project Page", 12 July 1995. <http://www.nbb.cornell.edu/neurobio/land/OldStudentProjects/cs718/fall1995/Landis> (20 July 2004).

21. http://members.shaw.ca/mikerennie/DB%20Library%20Tutorial.htm (2 November 2004)

22. Database Programming with JDBC and Java http://www.oreilly.com/catalog/javadata/chapter/ch04.html (2 November 2004)

23. The Java Tutorial -A practical guide for programmers http://java.sun.com/docs/books/tutorial (5 November 2004)

Electronic Book

24. World Book 2004 Deluxe

Appendix A SAMPLE SOURCE CODE

```
// DiagnosisInsertInterfaca.java

package patient.request.process;

import patient.util.*;
import patient.data.*;
import patient.exception.*;

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.sql.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.border.*;
import javax.swing.filechooser.*;
import javax.swing.BorderFactory;
import javax.swing.border.Border;


public class DiagnosisInsertInterface extends JFrame {
        private JTextField PatientIDTxtFld,
                CoroNameTxtFld, SagittNameTxtFld,TransNameTxtFld,
                NameTxtFld,
                AgeTxtFld,
                RaceTxtFld;

        private JButton CoroBrowse,SagittBrowse, TransBrowse, Clear, Summit;

        JLabel Im,Im2,Im3,CoroNameLabel,SagittNameLabel,TransNameLabel,
        PatientIDLabel,
        NameLabel,
        AgeLabel,
        RaceLabel;

        JPanel imagePanel =new JPanel();
        JPanel dataPanel = new JPanel();
        JPanel contentPane = new JPanel();

        JScrollPane imagePane = new JScrollPane(imagePanel) ;
        JComboBox race = null;
        JLabel phaseIconLabel = null;
    JLabel phaseIconLabel2 = null;
```

```
    JLabel phaseIconLabel3 = null;

        private JFileChooser chooser, chooser2, chooser3;

        int TextFieldSize = 10 ;
        String fileSelected;

//set up GUI
public DiagnosisInsertInterface() {

        super();

     //Add various widgets to the sub panels.
     addWidgets();

     //Create the contentPane to contain the two sub panels.
     contentPane = new JPanel();

     contentPane = (JPanel)this.getContentPane();
     //Add the data and image panels to the contentPane.
     contentPane.setLayout(null);
     addComponent(contentPane, imagePane, 6,3,1000,460);
     addComponent(contentPane, dataPanel, 6,480,1000,200);
     }

 /*
 * Get the images and set up the widgets.
 */
 private void addWidgets() {

      imagePane.createHorizontalScrollBar();
      imagePane.createVerticalScrollBar();

     phaseIconLabel = new JLabel();
  phaseIconLabel.setHorizontalAlignment(JLabel.CENTER);
  phaseIconLabel.setVerticalAlignment(JLabel.CENTER);
  phaseIconLabel.setVerticalTextPosition(JLabel.CENTER);
  phaseIconLabel.setHorizontalTextPosition(JLabel.CENTER);
  phaseIconLabel.setBorder(BorderFactory.createCompoundBorder(
     BorderFactory.createLoweredBevelBorder(),
     BorderFactory.createEmptyBorder(5,5,5,5)));

    phaseIconLabel2 = new JLabel();
 phaseIconLabel2.setHorizontalAlignment(JLabel.CENTER);
 phaseIconLabel2.setVerticalAlignment(JLabel.CENTER);
 phaseIconLabel2.setVerticalTextPosition(JLabel.CENTER);
 phaseIconLabel2.setHorizontalTextPosition(JLabel.CENTER);
```

```
phaseIconLabel2.setBorder(BorderFactory.createCompoundBorder(
    BorderFactory.createLoweredBevelBorder(),
    BorderFactory.createEmptyBorder(5,5,5,5)));


phaseIconLabel3 = new JLabel();
phaseIconLabel3.setHorizontalAlignment(JLabel.CENTER);
phaseIconLabel3.setVerticalAlignment(JLabel.CENTER);
phaseIconLabel3.setVerticalTextPosition(JLabel.CENTER);
phaseIconLabel3.setHorizontalTextPosition(JLabel.CENTER);
phaseIconLabel3.setBorder(BorderFactory.createCompoundBorder(
    BorderFactory.createLoweredBevelBorder(),
    BorderFactory.createEmptyBorder(5,5,5,5)));

    CoroNameTxtFld = new JTextField( 10 );
    JLabel CoroNameLabelLabel  =new JLabel( "Cororal Image File" );

    SagittNameTxtFld = new JTextField( 10);
    JLabel SagittNameLabel  =new JLabel( "     Sagittal Image File" );

    TransNameTxtFld = new JTextField( 10 );
    JLabel TransNameLabel  =new JLabel( "Transverse Image File" );

    //CoroBrowse Button
    CoroBrowse = new JButton();
    CoroBrowse.setText("Coronal Image");
    CoroBrowse.setSelected(true);
    CoroBrowse.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent ev)
                { processSelectFileAction();
                }
        });

    //CoroBrowse Button
    SagittBrowse = new JButton( "Sagittal Image" );
    SagittBrowse.addActionListener(new ActionListener() {

        public void     actionPerformed( ActionEvent ev )
        {processSelectFileAction2();
        }
});

    //TransBrowse Button
    TransBrowse = new JButton( "Transverse Image" );
    TransBrowse.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent ev)
```

```
                {
processSelectFileAction3();
                }
                });

        //Clear Button
        Clear = new JButton( "Clear" );
        Clear.addActionListener (new ActionListener() {

                public void     actionPerformed( ActionEvent ev )
                {       System.out.println( "Clear" );
                        processClearAction() ;
                }
                });

        //Summit Button
        Summit = new JButton( "Summit" );
        Summit.addActionListener (new ActionListener() {

                public void     actionPerformed( ActionEvent ev )

                {       System.out.println( "Summit" );
                        processSummitAction() ;
                }
                });


  //PatientIDTxtFlde line

        JLabel PatientIDLabel  =new JLabel("Patient ID  " );
        PatientIDLabel.setBackground(new Color(255, 255, 255));
        PatientIDLabel.setForeground(new Color(0, 51, 255));
        PatientIDTxtFld = new JTextField( 30 );

        // NameTxtFlde line
        JLabel NameLabel  =new JLabel( "Name   " );
        NameLabel.setBackground(new Color(255, 255, 255));
        NameLabel.setForeground(new Color(0, 51, 255));
        NameTxtFld = new JTextField( 30);

        // AgeTxtFlde line
        JLabel AgeLabel  =new JLabel( "Age " );
        AgeLabel.setBackground(new Color(255, 255, 255));
        AgeLabel.setForeground(new Color(0, 51, 255));
        AgeTxtFld = new JTextField( 20 );

        // RaceTxtFlde line
```

```
            JLabel RaceLabel  =new JLabel( "Race " );
            RaceLabel.setBackground(new Color(255, 255, 255));
            RaceLabel.setForeground(new Color(0, 51, 255));
            RaceTxtFld = new JTextField( 20 );

            //Add a border around the image panel.
          imagePane.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createTitledBorder("Images Phase"),
            BorderFactory.createEmptyBorder(5,5,5,5)));

        //Add a border around the data panel.
          dataPanel.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createTitledBorder("Data Phase"),
            BorderFactory.createEmptyBorder(5,5,5,5)));

         //Add imagePanel and dataPanel

         imagePanel.setLayout(null);
         addComponent(imagePanel, phaseIconLabel,  8,1,320,350);
         addComponent(imagePanel, phaseIconLabel2, 330,1,320,350);
         addComponent(imagePanel, phaseIconLabel3, 652,1,320,350);
         phaseIconLabel.setLayout(new FlowLayout(FlowLayout.CENTER, 5, 5));
         phaseIconLabel2.setLayout(new FlowLayout(FlowLayout.CENTER, 5, 5));
         phaseIconLabel3.setLayout(new FlowLayout(FlowLayout.CENTER, 5, 5));
            addComponent(imagePanel, CoroBrowse, 20,360,140,24);
            addComponent(imagePanel, SagittBrowse, 340,360,140,24);
            addComponent(imagePanel, TransBrowse, 660,360,140,24);
            addComponent(imagePanel, CoroNameTxtFld,    20,390,250,24);
        addComponent(imagePanel, SagittNameTxtFld, 340,390,250,24);
        addComponent(imagePanel, TransNameTxtFld, 660,390,250,24);

        dataPanel.setLayout(null);
        addComponent(dataPanel, PatientIDLabel,  20,25,140,24);
        addComponent(dataPanel, PatientIDTxtFld, 100,25,140,24);
        addComponent(dataPanel, NameLabel, 20,55,140,24);
        addComponent(dataPanel, NameTxtFld,100,55,140,24);
        addComponent(dataPanel, AgeLabel,20,85,140,24);
        addComponent(dataPanel, AgeTxtFld,100,85,140,24);
        addComponent(dataPanel, RaceLabel,20,115,140,24);
        addComponent(dataPanel, RaceTxtFld,100,115,140,24);
        addComponent(dataPanel, Clear, 350,80,140,24);
        addComponent(dataPanel, Summit, 350,110,140,24);
            }

private void processClearAction() {

        // clear images from GUI
```

```
        phaseIconLabel.setIcon(null);
        CoroNameTxtFld.setText("");
        SagittNameTxtFld.setText("");
        TransNameTxtFld.setText("");
        phaseIconLabel2.setIcon(null);
        phaseIconLabel3.setIcon(null);
        PatientIDTxtFld.setText("");
        NameTxtFld.setText("");
        AgeTxtFld.setText("");
        RaceTxtFld.setText("");
}

private void processSelectFileAction() {

        JFileChooser chooser = new JFileChooser();
        chooser.setDialogTitle("Select an image file");
        int returnVal = chooser.showOpenDialog(this);
        if( returnVal == JFileChooser.APPROVE_OPTION ) {
           String fileSelected = chooser.getCurrentDirectory() + "\\" +
chooser.getSelectedFile().getName();
                CoroNameTxtFld.setText( fileSelected );
                ImageIcon icon = new ImageIcon( fileSelected );
                phaseIconLabel.setIcon(icon);
                }
        }

private void processSelectFileAction2() {

        JFileChooser chooser2 = new JFileChooser();
        chooser2.setDialogTitle("Select an image file");
        int returnVal = chooser2.showOpenDialog(this);
        if( returnVal == JFileChooser.APPROVE_OPTION ) {
           String fileSelected2 = chooser2.getCurrentDirectory() + "\\" +
chooser2.getSelectedFile().getName();
                SagittNameTxtFld.setText( fileSelected2 );
        ImageIcon icon = new ImageIcon( fileSelected2 );
                phaseIconLabel2.setIcon(icon);
                }
        }

private void processSelectFileAction3() {

         JFileChooser chooser3 = new JFileChooser();
        chooser3.setDialogTitle("Select an image file");
        int returnVal = chooser3.showOpenDialog(this);
```

```
        if( returnVal == JFileChooser.APPROVE_OPTION ) {
           String fileSelected3 = chooser3.getCurrentDirectory() + "\\" +
chooser3.getSelectedFile().getName();

                TransNameTxtFld.setText( fileSelected3 );
                ImageIcon icon = new ImageIcon( fileSelected3 );
                phaseIconLabel3.setIcon(icon);
                }
        }

private void processSummitAction() {

        Connection conn = null;
        Statement stmt = null;
        boolean useDataSource = false;
        String providerURL = null;
        String initialContextFactory = null;
                ProcessPatientEntry test = new ProcessPatientEntry();

                // get context
        if ( useDataSource ) {
                System.out.println( "using DataSource" );
                test.useDataSource = true;
                if  (providerURL != null ) {
                        if ( initialContextFactory == null )
                                initialContextFactory = Util.initialContextFactory;

                        test.ctx = Util.getContext( providerURL, initialContextFactory );
                }
        } else {
                System.out.println( "using straight JDBC connection" );
                test.useDataSource = false;
        }

        java.util.Vector entriesV = new java.util.Vector(2);

        byte[] coroImage = null;
        byte[] sagitImage = null;
        byte[] transImage = null;
        ImageData coronalview = null;
        ImageData sagittalview = null;
        ImageData transverseview = null;

        //ImageData imageData = null;
        String patientID =null;
        String name = null;
        String age = null;
```

```
String race = null;
        String filCoro = CoroNameTxtFld.getText().trim();
        String fileC = parseFileName( filCoro);
        String filSaggit = SagittNameTxtFld.getText().trim();
        String fileS = parseFileName( filSaggit );
        String filTrans = TransNameTxtFld.getText().trim();
        String fileT = parseFileName( filTrans );
PatientData entry = null;
entry = new PatientData();

try{    // image
        coroImage = Util.getContent(filCoro);
        coronalview = new ImageData(coroImage);

        // image
        if ( fileC.toUpperCase().endsWith( "JPG" ) ) {
                coronalview.setImageFormat( "JPG" );
        }       else if ( fileC.toUpperCase().endsWith( "PNG" ) ) {
                coronalview.setImageFormat("PNG");
        }       else if ( fileC.toUpperCase().endsWith( "GIF" ) ) {
                coronalview.setImageFormat("GIF");
        }
                entry.setCoronalview(coronalview);

        try{

        sagitImage = Util.getContent(filSaggit);
        sagittalview = new ImageData(sagitImage);

                // image
        if ( fileS.toUpperCase().endsWith( "JPG" ) ) {
                sagittalview.setImageFormat( "JPG" );
        }       else if ( fileS.toUpperCase().endsWith( "PNG" ) ) {
                sagittalview.setImageFormat("PNG");
        }       else if ( fileS.toUpperCase().endsWith( "GIF" ) ) {
                sagittalview.setImageFormat("GIF");
        }
                entry.setSagittalview(sagittalview);

        }catch (java.io.IOException e) {

                                        e.printStackTrace();
                                        } catch (Exception e) {
                                                e.printStackTrace();
                                                }
```

```
try{


transImage = Util.getContent(filTrans);
transverseview = new ImageData(transImage);
        // image
if ( fileT.toUpperCase().endsWith( "JPG" ) ) {
        transverseview.setImageFormat( "JPG" );
}       else if ( fileT.toUpperCase().endsWith( "PNG" ) ) {
        transverseview.setImageFormat("PNG");
}       else if ( fileT.toUpperCase().endsWith( "GIF" ) ) {
        transverseview.setImageFormat("GIF");
}
        entry.setTransverseview(transverseview);
        }catch (java.io.IOException e) {
                                e.printStackTrace();
                                } catch (Exception e) {
                                        e.printStackTrace();
                                        }

        // Patient ID
                try {
                        patientID =  PatientIDTxtFld.getText().trim();
                        entry.setPatientID(patientID);
                        } catch (ArrayIndexOutOfBoundsException a) {}

                // Name
                try {
                        name =  NameTxtFld.getText().trim();
                        entry.setName(name);
                        } catch (ArrayIndexOutOfBoundsException a) {}

                // Age
                try {
                        age = AgeTxtFld.getText().trim();
                        entry.setAge(age);
                        } catch (ArrayIndexOutOfBoundsException a) {}

                // Race
                try {
                        race = RaceTxtFld.getText().trim();
                        entry.setRace(race);
                        } catch (ArrayIndexOutOfBoundsException a) {}
                        } catch (java.io.IOException e) {
                         e.printStackTrace();
                                } catch (Exception e) {
                                        e.printStackTrace();
```

```
                                        }
                        try {
                                test.execute(entry);
                                System.out.println("Patient table populated");
                                } catch (ProcessPatientCommandException e) {
                                        e.printStackTrace();
                                        }
                }

        private String parseFileName( String fileName ) {

        java.util.StringTokenizer tok = new java.util.StringTokenizer( fileName, "\\" );
        int count = tok.countTokens();
        String token = null;
        for (int i = 0; i < (count-1); ++i) token = tok.nextToken();
        fileName = tok.nextToken();
        return fileName;
}
/**
    * Create the GUI and show it.
    */
   private static void createAndShowGUI() {
//      Make sure we have nice window decorations.
      JFrame.setDefaultLookAndFeelDecorated(true);
      JDialog.setDefaultLookAndFeelDecorated(true);
                try
                {

        UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLook
AndFeel");
                }
                catch (Exception ex)
                {
                        System.out.println("Failed loading L&F: ");
                        System.out.println(ex);
                }

     //Create a new instance of DiagnosisInterface.
     DiagnosisInsertInterface phases = new DiagnosisInsertInterface();

     //Create and set up the window.
     JFrame DiagnosisInterfaceFrame = new JFrame("Diagnosis Interface");
     DiagnosisInterfaceFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     DiagnosisInterfaceFrame.setContentPane(phases.contentPane);

     //Display the window.
```

```
    DiagnosisInterfaceFrame. setSize(1070,750);
    DiagnosisInterfaceFrame.setVisible(true);
  }

      /** Add Component Without a Layout Manager (Absolute Positioning) */
      private void addComponent(Container container,Component c,int x,int y,int
width,int height)
      {
            c.setBounds(x,y,width,height);
            container.add(c);
      }

  public static void main(String[] args) {
    //Schedule a job for the event-dispatching thread:
    //creating and showing this application's GUI.
    DiagnosisInsertInterface application = new DiagnosisInsertInterface();
    application.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    javax.swing.SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        createAndShowGUI();
      }
    });
  }

}
```

```
// QueryDB.java

/*************************************************************/
/*                QueryDB                                    */
/*                                                           */
/*************************************************************/
package patient;
import patient.util.*;
import patient.data.*;
import patient.exception.*;
import patient.request.process.*;
import java.awt.*;
import java.io.*;
import java.sql.*;
import java.io.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
/**
 * Summary description for QueryDB
 *
 */
public class QueryDB extends JFrame
{

        // Variables declaration
        private JLabel textSearchLabel;
        private JLabel PatientIDLabel;
        private JLabel nameLabel;
        private JLabel VisualSearchOptionLabel;
        private JLabel VisualVocabularyLabel;
        private JLabel SearchResultsLabel;
        private JLabel compareToImage;
        private JRadioButton jRadioButton1;
        private JTextField PatientIDTextField;
        private JTextField NameField;
        private JTextField ChooseImageTextField;
        private JButton jButton1;
        private JButton jButton2;
        private JButton QueryCompare;
        private JButton viewModel;
        private JButton clear;

        ButtonGroup grp = new ButtonGroup();
        private JPanel contentPane;
```

```
    private JPanel textSearchPanel;
    private JPanel VisualSearchPanel;
    private JPanel SearchResultsPanel;
    JScrollPane jScrollPane1 = new JScrollPane(SearchResultsPanel) ;
    private String retrievedDir = "\\PATIENT\\patient\\retrieved\\";
    private String ModelFile=null;
    JLabel icon_label = new JLabel(new ImageIcon("bestfit.gif"));
    private String[] retrievedFiles;
    JTextField numberOfImage_val = new JTextField("");
    private String tableName = "PATIENT";
    private String imageColumn = "CORONALVIEW";
    private String imageColumn2 = "SAGITTALVIEW";
    private String imageColumn3 = "TRANSVERSEVIEW";
    int error = 0;

    // End of variables declaration

    public QueryDB()
    {
        super();
        initializeComponent();
        this.setVisible(true);
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
regenerated
     * by the Windows Form Designer. Otherwise, retrieving disign might not work
properly.
     */
    private void initializeComponent()
    {
        textSearchLabel = new JLabel();
        PatientIDLabel = new JLabel();
        nameLabel = new JLabel();
        VisualSearchOptionLabel = new JLabel();
        VisualVocabularyLabel = new JLabel();
        SearchResultsLabel = new JLabel();

        jRadioButton1 = new JRadioButton();
        PatientIDTextField = new JTextField();
        NameField = new JTextField();
        ChooseImageTextField = new JTextField();
        jButton1 = new JButton();
        jButton2 = new JButton();
```

```
            clear = new JButton();
            QueryCompare = new JButton();
            viewModel = new JButton();
            JLabel numberOfImage = new JLabel("Number of image(s) to retrieve ");

            textSearchPanel = new JPanel();
            VisualSearchPanel = new JPanel();
            SearchResultsPanel = new JPanel();

            jScrollPane1.createHorizontalScrollBar();
            jScrollPane1.createVerticalScrollBar();
            contentPane = (JPanel)this.getContentPane();

            textSearchLabel.setForeground(new Color(0, 51, 255));
            textSearchLabel.setText("Text Search");

            PatientIDLabel.setText("Patient ID ");
            nameLabel.setText("Name");

            VisualSearchOptionLabel.setForeground(new Color(0, 51, 255));
            VisualSearchOptionLabel.setText("Visual Search Option");

            // VisualVocabularyLabel
            VisualVocabularyLabel.setForeground(new Color(0, 51, 255));
            VisualVocabularyLabel.setText("Visual Vocabulary");

            //compareToImage JLabel
compareToImage = new JLabel();
    compareToImage.setHorizontalAlignment(JLabel.CENTER);
    compareToImage.setVerticalAlignment(JLabel.CENTER);
    compareToImage.setVerticalTextPosition(JLabel.CENTER);
    compareToImage.setHorizontalTextPosition(JLabel.CENTER);
    compareToImage.setBorder(BorderFactory.createCompoundBorder(
      BorderFactory.createLoweredBevelBorder(),
      BorderFactory.createEmptyBorder(5,5,5,5)));

    compareToImage.setBorder(BorderFactory.createCompoundBorder(
      BorderFactory.createEmptyBorder(0,0,10,0),
      compareToImage.getBorder()));

            // SearchResultsLabel
            //
            SearchResultsLabel.setBackground(new Color(255, 255, 255));
            SearchResultsLabel.setForeground(new Color(0, 51, 255));
            SearchResultsLabel.setText("Search Results");
```

```
// jRadioButton1
//
jRadioButton1.setText("Texture");
jRadioButton1.setSelected(true);
jRadioButton1.setActionCommand("TEXTURE");
grp.add(jRadioButton1);

viewModel.setText("View Model");
viewModel.setSelected(true);
viewModel.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e)
        {
                processViewModelAction();
        }

});

// Query jButton1
jButton1.setText("Query");
jButton1.setSelected(true);
jButton1.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e)
        {

        //      qbicQuery();
        processRetrieveCORONALVIEW();
        processRetrieveSAGITTALVIEW();
        processRetrieveTRANSVERSEVIEW();
                        }

});
//
// jButton2 choose file image button
//
jButton2.setText("Choose Image");
jButton2.setSelected(true);
jButton2.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e)
        {
                processSelectFileAction();
        }

});

//      Query image by Compare  jButton2
QueryCompare.setText("QUERY");
QueryCompare.addActionListener(new ActionListener() {
```

```
            public void actionPerformed(ActionEvent e)
            {
                processQueryAction();
            }

        });
        // clear button
        clear.setText("CLEAR");
//      clear.setSelected(true);
        clear.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e)
            {
            processClearAction();
            compareToImage.setIcon(null);
            ChooseImageTextField.setText("");
            }

        });

        // contentPane
        //
        contentPane.setLayout(null);
        contentPane.setBorder(BorderFactory.createEtchedBorder());

        addComponent(contentPane, textSearchLabel, 18,34,60,18);
        addComponent(contentPane, PatientIDLabel, 18,64,60,18);
        addComponent(contentPane, nameLabel, 18,90,60,18);
        addComponent(contentPane, VisualSearchOptionLabel, 18,145,114,18);
        addComponent(contentPane, VisualVocabularyLabel, 18,248,107,18);
        addComponent(contentPane, QueryCompare, 140,248,80,24);
        addComponent(contentPane, viewModel, 246,248,100,28);
        addComponent(contentPane, SearchResultsLabel, 367,27,85,18);
        addComponent(contentPane, jRadioButton1, 20,170,80,24);
        addComponent(contentPane, numberOfImage, 140,170,200,24);
        addComponent(contentPane, numberOfImage_val, 300,170,30,24);
        addComponent(contentPane, PatientIDTextField, 76,63,154,22);
        addComponent(contentPane, NameField, 76,90,154,22);
        addComponent(contentPane, jButton1, 246,60,83,24);
        addComponent(contentPane, clear, 246,87,83,24);
        addComponent(contentPane, jButton2, 15,205,104,24);
        addComponent(contentPane, ChooseImageTextField, 132,205,163,22);
        addComponent(contentPane, textSearchPanel, 6,24,345,110);
        addComponent(contentPane, VisualSearchPanel, 6,130,345,110);
        addComponent(contentPane, compareToImage, 9,280,342,420);
        addComponent(contentPane, jScrollPane1, 351,24,660,676);
```

```
            addComponent(contentPane, SearchResultsPanel, 351,24,660,676);

            // jPanel
            //
        textSearchPanel.setLayout(new FlowLayout(FlowLayout.CENTER, 5, 5));
        textSearchPanel.setBorder(BorderFactory.createEtchedBorder());
        VisualSearchPanel.setLayout(new FlowLayout(FlowLayout.CENTER, 5, 5));
        VisualSearchPanel.setBorder(BorderFactory.createEtchedBorder());
        SearchResultsPanel.setLayout(new FlowLayout(FlowLayout.CENTER, 5, 5));
        SearchResultsPanel.setBorder(BorderFactory.createEtchedBorder());
        compareToImage.setLayout(new FlowLayout(FlowLayout.CENTER, 5, 5));
        compareToImage.setBorder(BorderFactory.createEtchedBorder());
        // QueryDB
        //
        this.setTitle("QueryDB");
        this.setLocation(new Point(0, 0));
            this.setSize(new Dimension(1070, 740));
        }

        /** Add Component Without a Layout Manager (Absolute Positioning) */
        private void addComponent(Container container,Component c,int x,int y,int
width,int height)
        {
            c.setBounds(x,y,width,height);
            container.add(c);
        }
        // view model button action

        public void processViewModelAction() {

        ModelFile = "\\3.stl";
            {
                String as[] = {
                "cmd.exe",
                "/C",
                "D:\\PATIENT\\VIEWstl\\MR\\MPReader
D:\\PATIENT\\VIEWstl\\MODELS\\" + ModelFile
            };

                //DEBUG:
                    //System.out.println(as[0]);
                Runtime runtime = Runtime.getRuntime();
            try
                {
                    runtime.exec(as);
            }
                catch(IOException ioexception)
```

```
                {
                        System.err.println("IO exception: " +
ioexception.getMessage());
                }
            }

        }

        private void processRetrieveCORONALVIEW(){

        String patientID =null;
        String name =null;
        patientID = PatientIDTextField.getText().trim();
   name = NameField.getText().trim();
        PatientData patientData = null;
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs= null;

try {
                conn = getDBConnection();
                Util.log(this, "connection opened");
                String sql = null;

// try to set the function path, if it fails, it just means that it was already set for that
pooled connection
try {
                stmt = conn.createStatement();
                sql = "SET CURRENT FUNCTION PATH = mmdbsys, CURRENT
FUNCTION PATH";
                        Util.log(this, "executing update: " + sql);
                        stmt.executeUpdate(sql);

                } catch (java.sql.SQLException se) {
                        //Util.log(this, "benign!!! " + se.toString());
                } catch (java.lang.UnknownError ue) {
                        //Util.log(this, "benign!!! " + ue.toString());
                }

                sql = "SELECT "
                                + "Thumbnail(" + imageColumn +")"
                                + ",Comment(" + imageColumn +")"
                                + ",Filename(" + imageColumn +")"
                                + " FROM " + tableName
                                + " WHERE PATIENTID = \'" + patientID + "\'";
                                // AND NAME =\'" + name + "\'";
```

```
                Util.log(this, "executing query: " + sql);
                rs= stmt.executeQuery(sql);
                displayScoredDBImages(rs);

                } catch (Throwable e) {
                e.printStackTrace();
        } finally {
                try {
                        if (rs != null) {
                                rs.close();
                                Util.log(this,"result set closed");
                        }
                        if (conn != null) {
                                conn.close();
                                Util.log(this,"connection closed");
                        }

                }catch (Throwable e) {
                e.printStackTrace();
                }
        }
}

private void processRetrieveSAGITTALVIEW(){

        String patientID =null;
        String name =null;
         patientID = PatientIDTextField.getText().trim();
         name = NameField.getText().trim();
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs= null;
        PatientData patientData = null;

        try {
                conn = getDBConnection();
                Util.log(this, "connection opened");
                String sql = null;
// try to set the function path, if it fails, it just means that it was already set for that
pooled connection
        try {
        stmt = conn.createStatement();
        sql = "SET CURRENT FUNCTION PATH = mmdbsys, CURRENT
FUNCTION PATH";
                        Util.log(this, "executing update: " + sql);
                        stmt.executeUpdate(sql);
```

```
            } catch (java.sql.SQLException se) {
                    //Util.log(this, "benign!!! " + se.toString());
            } catch (java.lang.UnknownError ue) {
                    //Util.log(this, "benign!!! " + ue.toString());
            }

            sql = "SELECT "
                                    + "Thumbnail(" + imageColumn2 +")"
                                    + ",Comment(" + imageColumn2 +")"
                                    + ",Filename(" + imageColumn2 +")"
                                    + " FROM " + tableName
                                    + " WHERE PATIENTID = \'" + patientID + "\'"
                                    ;

            Util.log(this, "executing query: " + sql);
            rs= stmt.executeQuery(sql);
            displayScoredDBImages(rs);



    } catch (Throwable e) {
            e.printStackTrace();
    } finally {
            try {
                    if (rs != null) {
                            rs.close();
                            Util.log(this,"result set closed");
                    }

                    if (conn != null) {
                            conn.close();
                            Util.log(this,"connection closed");
                    }

            }catch (Throwable e) {
            e.printStackTrace();
            }
    }
}

private void processRetrieveTRANSVERSEVIEW(){

        String patientID =null;
        String name =null;
         patientID = PatientIDTextField.getText().trim();
         name = NameField.getText().trim();
```

```
        PatientData patientData = null;
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs= null;
        try {
                conn = getDBConnection();
                Util.log(this, "connection opened");
                String sql = null;
// try to set the function path, if it fails, it just means that it was already set for that
pooled connection
                try {
                        stmt = conn.createStatement();

                        sql = "SET CURRENT FUNCTION PATH = mmdbsys,
CURRENT FUNCTION PATH";

                        Util.log(this, "executing update: " + sql);
                        stmt.executeUpdate(sql);

                } catch (java.sql.SQLException se) {
                        //Util.log(this, "benign!!! " + se.toString());
                } catch (java.lang.UnknownError ue) {
                        //Util.log(this, "benign!!! " + ue.toString());
                }

                sql = "SELECT "
                                + "Thumbnail(" + imageColumn3 +")"
                                + ",Comment(" + imageColumn3 +")"
                                + ",Filename(" + imageColumn3 +")"
                                + " FROM " + tableName
                                + " WHERE PATIENTID = \'" + patientID + "\'";

                Util.log(this, "executing query: " + sql);
                rs= stmt.executeQuery(sql);
                displayScoredDBImages(rs);

        } catch (Throwable e) {
                e.printStackTrace();
        } finally {
                try {
                        if (rs != null) {
                                rs.close();
                                Util.log(this,"result set closed");
                        }
                        if (conn != null) {
                                conn.close();
                                Util.log(this,"connection closed");
```

```
                }

            }catch (Throwable e) {
            e.printStackTrace();
            }
        }
}
        private void qbicQuery() {
        Util.log(this, "qbicQuery1");
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;
        Statement stmt2 = null;
        ResultSet rs2 = null;
        Statement stmt3 = null;
        ResultSet rs3 = null;
        String patientID =null;
        String name =null;
         patientID = PatientIDTextField.getText().trim();
         name = NameField.getText().trim();

        try {
                conn = getDBConnection();
                Util.log(this, "connection opened");

                String sql = null;
                String sql2 = null;
                String sql3 = null;

                // try to set the function path, if it fails, it just means that it was already
set for that pooled connection
                try {
                        stmt = conn.createStatement();
        sql = "SET CURRENT FUNCTION PATH = mmdbsys, CURRENT
FUNCTION PATH";
                        Util.log(this, "executing update: " + sql);
                        stmt.executeUpdate(sql);

                } catch (java.sql.SQLException se) {
                        //Util.log(this, "benign!!! " + se.toString());
                } catch (java.lang.UnknownError ue) {
                        //Util.log(this, "benign!!! " + ue.toString());
                }

                sql = "SELECT "
                                        + "Thumbnail(" + imageColumn +")"
                                        + ",Comment(" + imageColumn +")"
```

```
                                + ",Filename(" + imageColumn +")"
                                + " FROM " + tableName
                                + " WHERE PATIENTID = \'" + patientID + "\'"
                                ;
                Util.log(this, "executing query: " + sql);
                rs = stmt.executeQuery(sql);
                displayScoredDBImages(rs);

                try {
                stmt2 = conn.createStatement();
        sql2 = "SET CURRENT FUNCTION PATH = mmdbsys, CURRENT
FUNCTION PATH";

                        Util.log(this, "executing update: " + sql2);
                        stmt2.executeUpdate(sql2);
                } catch (java.sql.SQLException se) {
                        //Util.log(this, "benign!!! " + se.toString());
                } catch (java.lang.UnknownError ue) {
                        //Util.log(this, "benign!!! " + ue.toString());
                }
                        sql2 = "SELECT "
                                + "Thumbnail(" + imageColumn2 +")"
                                + ",Comment(" + imageColumn2 +")"
                                + ",Filename(" + imageColumn2 +")"
                                + " FROM " + tableName
                                + " WHERE PATIENTID = \'" + patientID + "\'"
                                ;

                Util.log(this, "executing query: " + sql2);
                rs2 = stmt2.executeQuery(sql2);
                displayScoredDBImages(rs2);

        try {
                stmt3 = conn.createStatement();
sql3 = "SET CURRENT FUNCTION PATH = mmdbsys, CURRENT FUNCTION
PATH";
                Util.log(this, "executing update: " + sql3);
                        stmt3.executeUpdate(sql3);
                } catch (java.sql.SQLException se) {
                        //Util.log(this, "benign!!! " + se.toString());
                } catch (java.lang.UnknownError ue) {
                        //Util.log(this, "benign!!! " + ue.toString());
                }
                sql3 = "SELECT "
                                + "Thumbnail(" + imageColumn3 +")"
                                + ",Comment(" + imageColumn3 +")"
                                + ",Filename(" + imageColumn3 +")"
```

```
                                + " FROM " + tableName
                                + " WHERE PATIENTID = \'" + patientID + "\'";

            Util.log(this, "executing query: " + sql3);
            rs3 = stmt3.executeQuery(sql3);
            displayScoredDBImages(rs3);
        } catch (Throwable e) {
            e.printStackTrace();
        } finally {
            try {
                if (rs != null) {
                    rs.close();
                    Util.log(this,"result set closed");
                }
                if (rs2 != null) {
                    rs2.close();
                    Util.log(this,"result2 set closed");
                }
                if (rs3 != null) {
                    rs3.close();
                    Util.log(this,"result3 set closed");
                }
                if (stmt != null) {
                    stmt.close();
                    Util.log(this,"statement closed");
                }
                if (stmt2 != null) {
                    stmt2.close();
                    Util.log(this,"statement closed");
                }
                if (stmt3 != null) {
                    stmt3.close();
                    Util.log(this,"statement closed");
                }
                if (conn != null) {
                    conn.close();
                    Util.log(this,"connection closed");
                }
            }catch (Throwable e) {
            e.printStackTrace();
            }
        }

}

private void displayScoredDBImages(ResultSet rs) throws java.sql.SQLException {
```

```
        Util.log(this, "displayScoredDBImages");
        java.util.Vector filesV = new java.util.Vector();
        String[] files = null;

        try {
                int row = 0;
                while (rs.next()) {
                        System.out.println("row[" + (++row) + "]");
                // clear and delete previous retrieved files
           // if (row == 1) {
//                      processClearAction();
            //      }
                        int column = 0;
                        byte[] thumbnail = rs.getBytes(++column);
                        String comment = rs.getString(++column);
                        String fileName = rs.getString(++column);
                //      double score = rs.getDouble(++column);

                        System.out.println(
                                "\t thumbnail: "
                                        + ((thumbnail == null) ? "null" : "byte array")
                                        + "\t comment: "
                                        + comment
                                        + "\t fileName: "
                                        + fileName);
                if (fileName != null)
                                fileName = parseFileName(fileName);

                        if (thumbnail != null) {
                                //** thumbnails are always returned as GIF's **/
                                String thumbnail_format = "GIF";
                                String useFileName =
                                        retrievedDir    + row + "thumb_" + ((fileName !=
null)
? (fileName + "." + thumbnail_format)
: (comment + "." + thumbnail_format));
                                writeImageToFile(thumbnail, useFileName);
                                filesV.add(useFileName);
                                        }
                                }
        } catch (SQLException e) {
                throw e;
}

        //************************************************************
        //
        //  Add the bestfit picture to the bottom panel
```

```
        //
        //*************************************************************

        if (filesV.size() > 0) {
                files = new String[filesV.size()];
                filesV.copyInto(files);
        }
        if (files != null) {
                retrievedFiles = files;

                for (int i = 0; i < files.length; ++i) {
                        icon_label = new JLabel(new ImageIcon(files[i]));
                        SearchResultsPanel.add(icon_label);
                }
                SearchResultsPanel.add(icon_label);
                SearchResultsPanel.repaint();
                SearchResultsPanel.invalidate();
                SearchResultsPanel.doLayout();
                }
}
        private void processQueryAction() {
        int error = 0;
        String msg = new String("");
        // build qbic query string
        String file = ChooseImageTextField.getText().trim();
        String qbicQuery = null;
        if (grp.getSelection().getActionCommand() == "TEXTURE") {
                qbicQuery =
                        "\'texture file=<server,"
                                + file
                                + ">\'";

        }
        // execute the query
        if (qbicQuery != null) {

           int numberToRetrieve = 0;
           String val = numberOfImage_val.getText().trim();

           if ( (val != null) && (val.length() > 0) ) {
                try {
                        numberToRetrieve = Integer.parseInt(val);
                      } catch (NumberFormatException e) {
                        error = 1;
                  showError(this, "the number of images to retrieve field does not contain
a number (" + val + ")");
                      }
           }
```

```
        if (numberToRetrieve < 0) {
                error = 1;
                showError(this, "the number of images to retrieve field cannot be
less than 0 (" + val + ")");
        }
            try {
                if (numberToRetrieve > 0) {
                    qbicQuery2(qbicQuery, numberToRetrieve);
                } else {
                    qbicQuery1(qbicQuery);
                }
            } catch (SQLException ex) {
                error = 1;
                showError(this, ex.getMessage());
            }
    }
}

public void showError(JFrame f, String msg)
 {
        final JDialog d = new JDialog(f, "Exception Caught", true);
        d.setSize(400,150);
        JLabel l = new JLabel();
        JTextArea ta = new JTextArea();

        ta.setLineWrap(true);
        ta.setWrapStyleWord(true);
        ta.setEditable(false);
        ta.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
        ta.getBorder();
        ta.setBackground(new Color(220, 220, 220));
        ta.append(msg);

        d.getContentPane().setLayout(new BorderLayout());
        d.getContentPane().add(ta, BorderLayout.CENTER);
        JButton b = new JButton("OK");
        b.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent ev) {
              d.setVisible(false);
              d.dispose();
            }
        });

        JPanel p = new JPanel();
        p.add (b);
        p.setBackground(new Color(220, 220, 220));
        d.getContentPane().add(p, BorderLayout.SOUTH);
```

```
        d.setLocationRelativeTo (f);
        d.setVisible(true);
    }
private void processSelectFileAction() {
        JFileChooser chooser = new JFileChooser();
        chooser.setDialogTitle("Select an image file");
        int returnVal = chooser.showOpenDialog(this);
        if(returnVal == JFileChooser.APPROVE_OPTION) {
           String fileSelected = chooser.getCurrentDirectory() + "\\" +
chooser.getSelectedFile().getName();
                Util.log(this, "file selected: " + fileSelected);
                ChooseImageTextField.setText(fileSelected);
                ImageIcon icon = new ImageIcon(fileSelected);
                compareToImage.setIcon(icon);
        }

}
private void qbicQuery1(String qbicQuery) throws java.sql.SQLException {

        Util.log(this, "qbicQuery1");

        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;
        Statement stmt2 = null;
        ResultSet rs2 = null;
        Statement stmt3 = null;
        ResultSet rs3 = null;


        try {
                conn = getDBConnection();
                Util.log(this, "connection opened");
                String sql = null;
                String sql2 = null;
                String sql3 = null;

                // try to set the function path, if it fails, it just means that it was already
set for that pooled connection
                try {
                        stmt = conn.createStatement();

                        sql = "SET CURRENT FUNCTION PATH = mmdbsys,
CURRENT FUNCTION PATH";

                Util.log(this, "executing update: " + sql);
                        stmt.executeUpdate(sql);
```

```
            } catch (java.sql.SQLException se) {
                    //Util.log(this, "benign!!! " + se.toString());
            } catch (java.lang.UnknownError ue) {
                    //Util.log(this, "benign!!! " + ue.toString());
            }
                    sql = "SELECT "
                                    + "Thumbnail(" + imageColumn +")"
                                    + ",Comment(" + imageColumn +")"
                                    + ",Filename(" + imageColumn +")"
                                    + ",QBScoreFromStr(" + qbicQuery + "," +
imageColumn + ") AS SCORE" +
                                    " FROM " + tableName +
                                    " ORDER BY SCORE"
                                    ;
            Util.log(this, "executing query: " + sql);
            rs = stmt.executeQuery(sql);
            displayScoredImages(rs);

            try {

                    stmt2 = conn.createStatement();
                    sql2 = "SET CURRENT FUNCTION PATH = mmdbsys,
CURRENT FUNCTION PATH";

                    Util.log(this, "executing update: " + sql2);
                    stmt2.executeUpdate(sql2);
            } catch (java.sql.SQLException se) {
                    //Util.log(this, "benign!!! " + se.toString());
            } catch (java.lang.UnknownError ue) {
                    //Util.log(this, "benign!!! " + ue.toString());
            }

            sql2 = "SELECT "
                                    + "Thumbnail(" + imageColumn2 +")"
                                    + ",Comment(" + imageColumn2 +")"
                                    + ",Filename(" + imageColumn2 +")"
                                    + ",QBScoreFromStr(" + qbicQuery + "," +
imageColumn2 + ") AS SCORE" +
                                    " FROM " + tableName +
                                    " ORDER BY SCORE"
                                    ;

                                    try {

                    stmt3 = conn.createStatement();
```

```
                sql3 = "SET CURRENT FUNCTION PATH = mmdbsys,
CURRENT FUNCTION PATH";

                Util.log(this, "executing update: " + sql3);
                stmt3.executeUpdate(sql3);
            } catch (java.sql.SQLException se) {
                //Util.log(this, "benign!!! " + se.toString());
            } catch (java.lang.UnknownError ue) {
                //Util.log(this, "benign!!! " + ue.toString());
            }

            Util.log(this, "executing query: " + sql2);
            rs2 = stmt2.executeQuery(sql2);
            displayScoredImages(rs2);

            sql3 = "SELECT "
                                + "Thumbnail(" + imageColumn3 +")"
                                + ",Comment(" + imageColumn3 +")"
                                + ",Filename(" + imageColumn3 +")"
                                + ",QBScoreFromStr(" + qbicQuery + "," +
imageColumn3 + ") AS SCORE" +
                                " FROM " + tableName +
                                " ORDER BY SCORE"
                                ;

            Util.log(this, "executing query: " + sql3);
            rs3 = stmt3.executeQuery(sql3);

            displayScoredImages(rs3);

        } catch (java.sql.SQLException e) {
            Util.log(this, e);
            throw e;
        } catch (Throwable e) {
            Util.log(this, e);
            throw new java.sql.SQLException(e.toString());
        } finally {
            try {
                if (rs != null) {
                    rs.close();
                    Util.log(this,"result set closed");
                }
                if (rs2 != null) {
                    rs2.close();
                    Util.log(this,"result2 set closed");
                }
                if (rs3 != null) {
```

```
                    rs3.close();
                    Util.log(this,"result3 set closed");
                }
                if (stmt != null) {
                    stmt.close();
                    Util.log(this,"statement closed");
                }
                if (stmt2 != null) {
                    stmt2.close();
                    Util.log(this,"statement closed");
                }
                if (stmt3 != null) {
                    stmt3.close();
                    Util.log(this,"statement closed");
                }
                if (conn != null) {
                    conn.close();
                    Util.log(this,"connection closed");
                }

            } catch (java.sql.SQLException e) {
                Util.log(this, e);
                throw e;
            }
        }

}
private void qbicQuery2(String qbicQuery, int numberToRetrieve) throws
java.sql.SQLException {

        Util.log(this, "qbicQuery2 - numberToRetrieve: " + numberToRetrieve);

        // Retrieve an image to a client buffer or file, based on QBIC query using
QBScoreFromStr UDF
        //
        // CONTENT(
        //              DB2IMAGE - the column name where the image is stored
        // )

        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;

        try {
            conn = getDBConnection();
            Util.log(this, "connection opened");
```

```
            String sql = null;
            // try to set the function path, if it fails, it just means that it was already
set for that pooled connection
            try {
                  stmt = conn.createStatement();
                  sql = "SET CURRENT FUNCTION PATH = mmdbsys,
CURRENT FUNCTION PATH";
                  Util.log(this, "executing update: " + sql);
                  stmt.executeUpdate(sql);
            } catch (java.sql.SQLException se) {
                  //Util.log(this, "benign!!! " + se.toString());
            } catch (java.lang.UnknownError ue) {
                  //Util.log(this, "benign!!! " + ue.toString());
            }
            // execute the QBIC query

            sql = "SELECT "
                              + "Thumbnail(IMAGE_ID)"
                              + ",Comment(IMAGE_ID)"
                              + ",Filename(IMAGE_ID)"
                              + ",SCORE"
                              + " FROM TABLE"
                              + " (QbScoreTBFromStr (" + qbicQuery
                              +       ", \'" + tableName + "\'"
                              +       ", \'" + imageColumn + "\'"
                              + ", " + numberToRetrieve + ")"
                              + ") AS T1"
                              + " ORDER BY SCORE"
                              ;

            Util.log(this, "executing query: " + sql);
            rs = stmt.executeQuery(sql);

            displayScoredImages(rs);

      } catch (java.sql.SQLException e) {
            Util.log(this, e);
            throw e;
      } catch (Throwable e) {
            Util.log(this, e);
            throw new java.sql.SQLException(e.toString());
      } finally {
            try {
                  if (rs != null) {
                        rs.close();
                        Util.log(this,"result set closed");
                  }
```

```
                if (stmt != null) {
                        stmt.close();
                        Util.log(this,"statement closed");
                }

                if (conn != null) {
                        conn.close();
                        Util.log(this,"connection closed");
                }

        } catch (java.sql.SQLException e) {
                Util.log(this, e);
                throw e;
        }
    }

}

private void displayScoredImages(ResultSet rs) throws java.sql.SQLException {

    Util.log(this, "displayScoredImages");
    java.util.Vector filesV = new java.util.Vector();
    String[] files = null;

    try {
        int row = 0;
        while (rs.next()) {
            System.out.println("row[" + (++row) + "]");
        // clear and delete previous retrieved files
        if (row == 1) {
      processClearAction();
       }
            int column = 0;
            byte[] thumbnail = rs.getBytes(++column);
            String comment = rs.getString(++column);
            String fileName = rs.getString(++column);
            double score = rs.getDouble(++column);
            System.out.println(
                "\t thumbnail: "
                    + ((thumbnail == null) ? "null" : "byte array")
                    + "\t comment: "
                    + comment
                    + "\t fileName: "
                    + fileName
                    + "\t score: "
```

```
                                + score);


                        if (fileName != null)
                                fileName = parseFileName(fileName);

                        if (thumbnail != null) {
                                //** thumbnails are always returned as GIF's **/
                                String thumbnail_format = "GIF";
                                String useFileName =
                                        retrievedDir    + row + "thumb_" + ((fileName !=
null)
                                                                ? (fileName + "." +
thumbnail_format)
                                                                : (comment + "." +
thumbnail_format));
                                writeImageToFile(thumbnail, useFileName);
                                filesV.add(useFileName);
                        }
                }
        } catch (SQLException e) {
                throw e;
}

        //*********************************************************
        //
        //  Add the bestfit picture to the bottom panel
        //
        //*********************************************************

        if (filesV.size() > 0) {
                files = new String[filesV.size()];
                filesV.copyInto(files);
        }

        if (files != null) {
                retrievedFiles = files;
                for (int i = 0; i < files.length; ++i) {
                        icon_label = new JLabel(new ImageIcon(files[i]));
                        SearchResultsPanel.add(icon_label);
                }
                SearchResultsPanel.add(icon_label);
                SearchResultsPanel.repaint();
                SearchResultsPanel.invalidate();
                SearchResultsPanel.doLayout();

        }
```

```
}

private void writeImageToFile(byte[] image, String filename) {

        try {
                System.out.println("writing image to file: " + filename);
                java.io.FileOutputStream file = new FileOutputStream(filename);
                file.write(image);
                file.close();
                System.out.println("image written to file: " + filename);
        } catch (java.io.FileNotFoundException e) {
                System.out.println("writeImageToFile: " + e);
        } catch (java.io.IOException e) {
                System.out.println("writeImageToFile: " + e);
        }
}

private String parseFileName( String fileName ) {

        java.util.StringTokenizer tok = new java.util.StringTokenizer( fileName, "\\" );
        int count = tok.countTokens();
        String token = null;
        for (int i = 0; i < (count-1); ++i) token = tok.nextToken();
        fileName = tok.nextToken();
        return fileName;
}

private void processClearAction() {


        Connection conn = null;
        try {
                        // close connection
                        if (conn != null) {
                                conn.close();
                                }
                } catch (java.sql.SQLException e) {
                        Util.log(this, e);
                        //throw new ProcessPatientCommandException(e.toString());
                }


        // clear images from GUI
        textSearchPanel.removeAll();
        textSearchPanel.invalidate();
        PatientIDTextField.setText(null);
```

```
        NameField.setText(null);
        jButton1.removeAll();
        jButton1.invalidate();
        SearchResultsPanel.removeAll();
        SearchResultsPanel.invalidate();
        SearchResultsPanel.repaint();
        SearchResultsPanel.doLayout();

                // delete retrieved files from temporary file system storage
        if (retrievedFiles != null) {
                Util.log(this, "deleted previously retrieved files");
                for (int i = 0; i < retrievedFiles.length; ++i) {
                        File file = new File(retrievedFiles[i]);
                        file.delete();
                }
        }
}

private java.sql.Connection getDBConnection() throws java.sql.SQLException {

        Util.log(this, "getDBConnection");
        java.sql.Connection connection = null;
        try {
                connection = Util.getDBConnection();
        } catch (java.sql.SQLException e) {
                Util.log(this, e);
                throw e;
        }
        return connection;
}

        public static void main(String[] args){
        // read in parameters
        boolean useDataSource = false;
        String providerURL = null;
        String initialContextFactory = null;

        if (args.length > 0) {
                String param = args[0];
                if (param.equalsIgnoreCase("useDataSource"))
                        useDataSource = true;
        }

        if (useDataSource) {
                if (args.length > 1) {
                        providerURL = args[1];
                        System.out.println("providerURL set to: " + providerURL);
```

```
            }

            if (args.length > 2) {
                    initialContextFactory = args[2];
                    System.out.println("initialContextFactory set to: " +
initialContextFactory);
            }
        }

        // create QueryDB app

            JFrame.setDefaultLookAndFeelDecorated(true);
            JDialog.setDefaultLookAndFeelDecorated(true);
            try
            {

        UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLook
AndFeel");
            }
            catch (Exception ex)
            {
                    System.out.println("Failed loading L&F: ");
                    System.out.println(ex);
            }
            System.out.println("creating QueryDB instance");
            QueryDB q = new QueryDB();
            System.out.println("QueryDB instance created");
            //q.setLocation(60, 10);
            q.setVisible(true);

        }

}
```

Appendix B

```
db2ext => connect to medical

   Database Connection Information

Database server      = DB2/NT 8.1.3
Local database alias = MEDICAL

db2ext => enable database for db2image
DMB0027I: The current database is enabled for extender "db2image".
db2ext => enable table patient for db2image
DMB0028I: Table "patient" is enabled for extender "db2image".
db2ext => ENABLE COLUMN patient CoronalVIEW FOR DB2IMAGE
DMB0029I: Column "CoronalVIEW" in table "patient" is enabled for extender
"DB2IMAGE".
db2ext => ENABLE COLUMN patient sagittalVIEW FOR DB2IMAGE
DMB0029I: Column "sagittalVIEW" in table "patient" is enabled for extender
"DB2IMAGE".
db2ext => ENABLE COLUMN patient TransverseVIEW FOR DB2IMAGE
DMB0029I: Column "TransverseVIEW" in table "patient" is enabled for extender
"DB2IMAGE".
db2ext =>
db2ext => create qbic catalog patient CoronalVIEW on
DMB0055I: The "CREATE QBIC CATALOG" command completed successfully.
db2ext =>
db2ext => create qbic catalog patient sagittalVIEW on
DMB0055I: The "CREATE QBIC CATALOG" command completed successfully.
db2ext => create qbic catalog patient TransverseVIEW on
DMB0055I: The "CREATE QBIC CATALOG" command completed successfully.
db2ext => open qbic catalog patient CoronalVIEW
DMB0055I: The "OPEN QBIC CATALOG" command completed successfully.
db2ext => open qbic catalog patient sagittalVIEW
DMB0055I: The "OPEN QBIC CATALOG" command completed successfully.
db2ext => open qbic catalog patient TransverseVIEW
DMB0055I: The "OPEN QBIC CATALOG" command completed successfully.
db2ext => add qbic feature texture
DMB0055I: The "ADD QBIC FEATURE" command completed successfully.
db2ext => GET QBIC CATALOG INFO
In the QBIC catalog for column TRANSVERSEVIEW table DB2ADMIN.PATIENT,
auto-cataloging has been set to ON.
There are 1 features:
 QbTextureFeatureClass
DMB0055I: The "GET QBIC CATALOG INFO" command completed successfully.
db2ext => open qbic catalog patient sagittalVIEW
```

DMB0055I: The "OPEN QBIC CATALOG" command completed successfully.
db2ext => GET QBIC CATALOG INFO
In the QBIC catalog for column SAGITTALVIEW table DB2ADMIN.PATIENT, auto-cataloging has been set to ON.
There are 0 features:

DMB0055I: The "GET QBIC CATALOG INFO" command completed successfully.
db2ext => add qbic feature texture
DMB0055I: The "ADD QBIC FEATURE" command completed successfully.
db2ext => GET QBIC CATALOG INFO
In the QBIC catalog for column SAGITTALVIEW table DB2ADMIN.PATIENT, auto-cataloging has been set to ON.
There are 1 features:
QbTextureFeatureClass
DMB0055I: The "GET QBIC CATALOG INFO" command completed successfully.
db2ext => open qbic catalog patient CoronalVIEW
DMB0055I: The "OPEN QBIC CATALOG" command completed successfully.
db2ext => add qbic feature texture
DMB0055I: The "ADD QBIC FEATURE" command completed successfully.
db2ext => GET QBIC CATALOG INFO
In the QBIC catalog for column CORONALVIEW table DB2ADMIN.PATIENT, auto-cataloging has been set to ON.
There are 1 features:
QbTextureFeatureClass
DMB0055I: The "GET QBIC CATALOG INFO" command completed successfully.

Appendix C USER MANUAL

Preparation

Before you can begin using the CBIR system, there are a few preparations to make. First and foremost, you need to make sure that Java Software Development Kit (JDK) is installed in your operating system.

At the time of the development of this system, the version of Java 2 Standard Edition (J2SE) Development Kit that was used is jdk1.5.0_01. Newer versions of the development kit may have been developed and are usually backward compatible with older versions.

Make sure you download and use the correct JDK according to the operating system you are using. Otherwise, the development kit would not set up properly and yield installation errors.

This CBIR system was developed mainly under the JCreator Light Edition (LE) v2.50 IDE produced by Xinox Software. The Light Edition is a freeware available for downloads at http://www.jcreator.com . Besides that, this system also uses non-commercial IDEs such as NetBeans, Eclipse and JFrameBuilder to write coding of certain components.

The second things need to do is to create a user account for the window, which user name is "db2admin" with password "user". After that, use the account to install software DB2 UDB Personal Edition for Window XP or DB2 UDB Enterprise Edition for Window 2000 Server. Please refer the installation requirement for different platform.

After install the DB2 UDB, you can test the installation by type "db2start" in the command line processor. The successful installation is shown below.

```
(c) Copyright IBM Corporation 1993,2002
Command Line Processor for DB2 SDK 8.1.3

You can issue database manager commands and SQL statements from the command
prompt. For example:
   db2 => connect to sample
   db2 => bind sample.bnd

For general help, type: ?.
For command help, type: ? command, where command can be
the first few keywords of a database manager command. For example:
 ? CATALOG DATABASE for help on the CATALOG DATABASE command
 ? CATALOG         for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

db2 =>  db2start
SQL1026N  The database manager is already active.
db2 =>
```

After the installation of DB2, continue with the installation of DB2 AIV Extenders. After finish this installation, test the installation by type "dmbstart" in the DB2 AIV Extenders Command Line Processor.

Now you can start create the database Medical and table with name PATIENT by type the following code in the DB2 Command Line Processor.

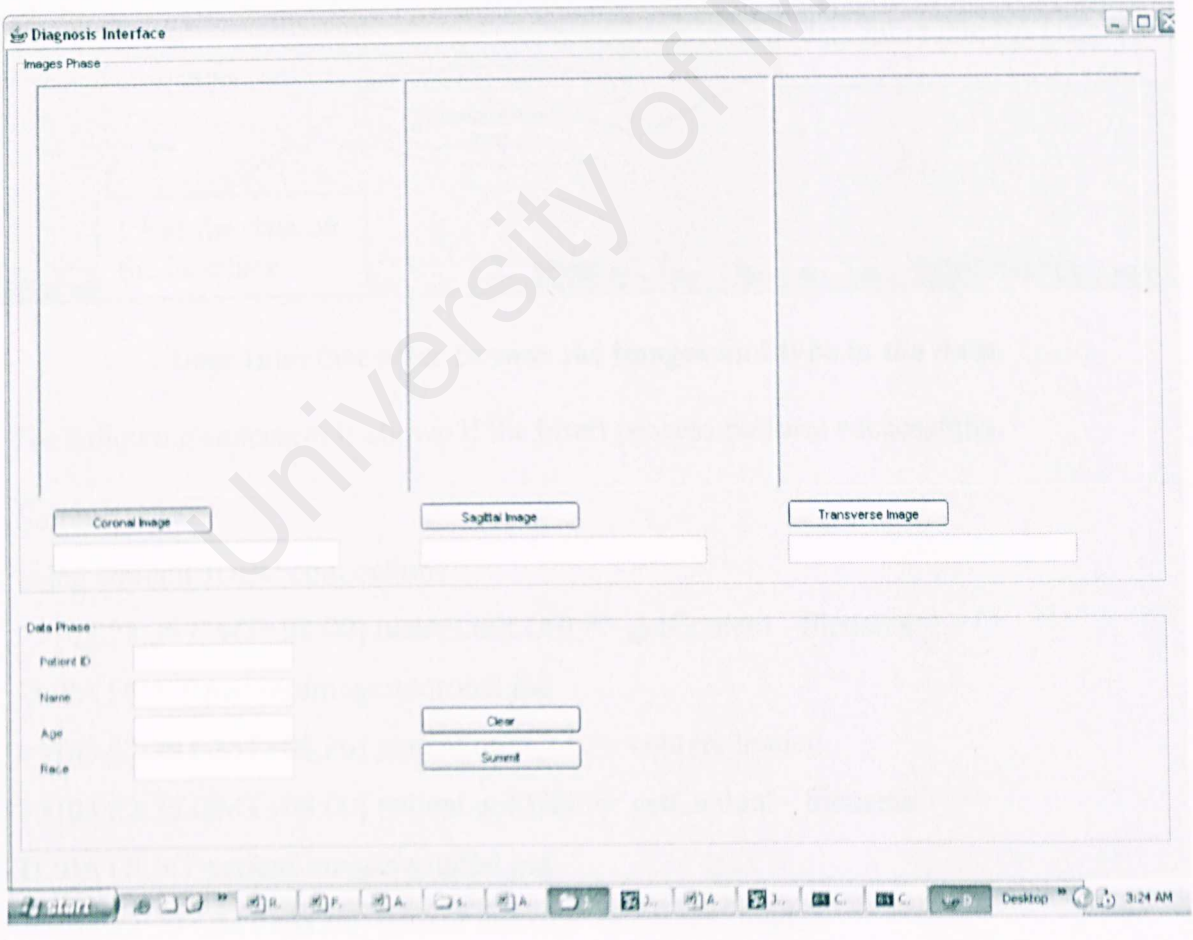
```
db2 => create database medical
db2 => The CREATE DATABASE command completed successfully.
db2 => create table patient(patientID char(10) not null primary key, Name long varchar,age
char(4), Race char(12), CoronalVIEW mmdbsys.db2image, SagittalVIEW
mmdbsys.db2image,TransverseVIEW mmdbsys.db2image)
```

After done these, refer the **Appendix B** to enable the database for db2image which is the feature of the DB2 AIV Extenders.

In order to view the model and 3D CAD, there need to install the ModelPressreader software.

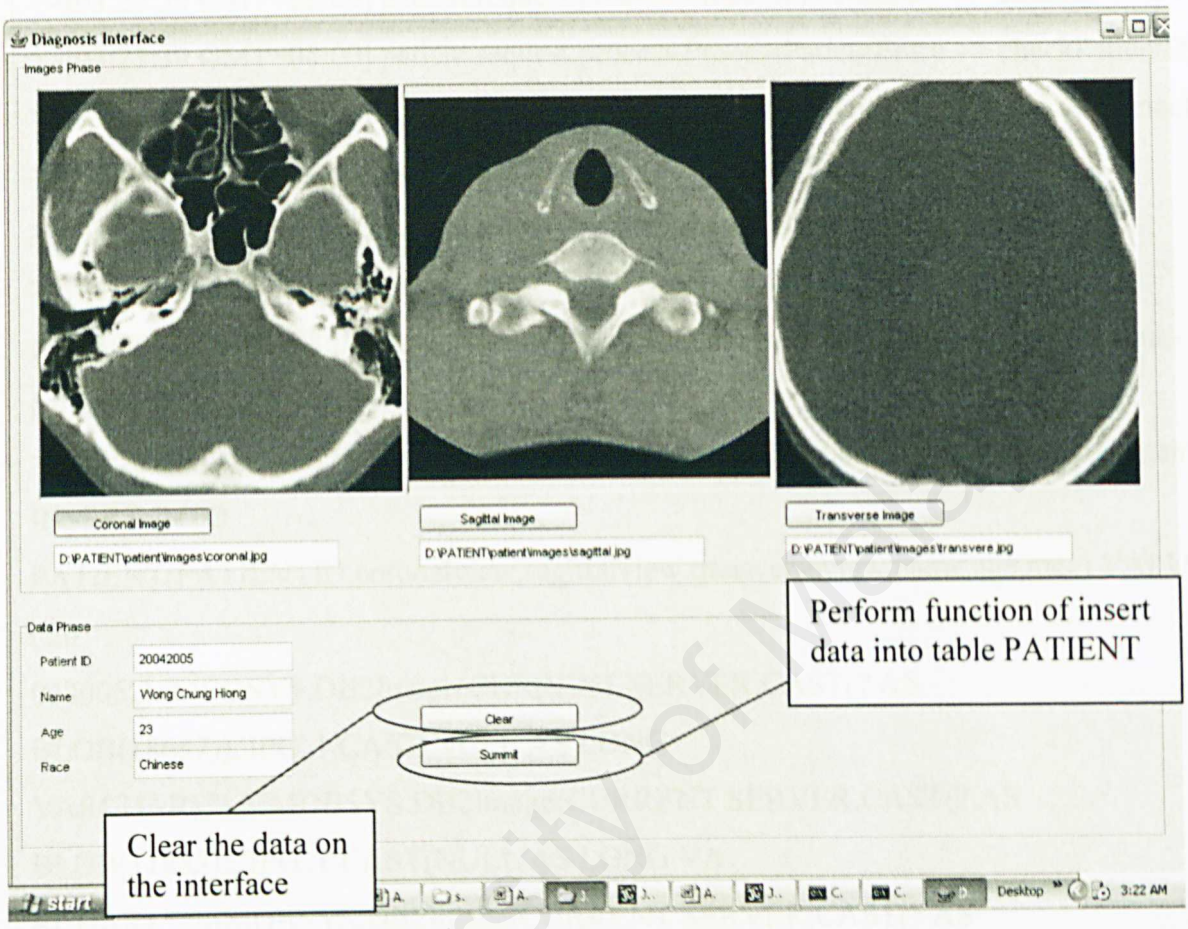
Demo for using the GUIs

After install the DB2 software and configure the database installation, now can open the PATIENT folder and search for DiagnosisInsertInterface. Open this java file, compile and run it. Accually the DiagnosisInsertInterface MS-DOS Batch File was created to open the java automatically, but still not function cause by the bug in the DiagnosisInsertInterface.java. Some procedure needs to perform with QueryDB.java After run the DiagnosisInsertInterface, the following interface will prompt out which used to insert data and images.



User Interface for Insert Data and Images

After insert the data and image, click the summit button. Please define and use the correct images. The sample image below is just for demo.



User Interface after browse the images and type in the data

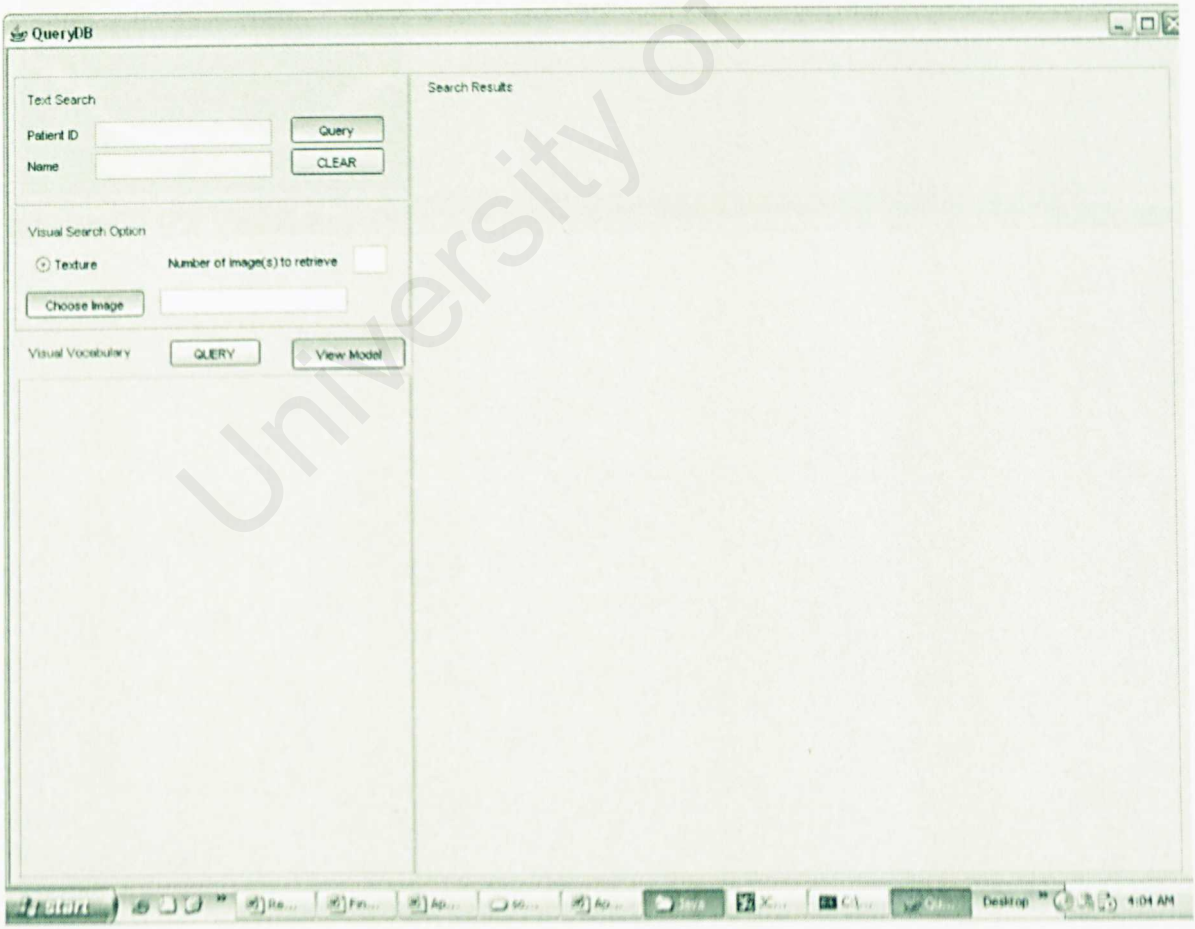
The following statement is shown if the insert process perform successfully.

```
Summit
using straight JDBC connection
>>[03:22:39 GMT+08:00] patient.util.Util >> getContent - filename:
D:\PATIENT\patient\images\coronal.jpg
>>[03:22:39 GMT+08:00] patient.util.Util >> content loaded
>>[03:22:39 GMT+08:00] patient.util.Util >> getContent - filename:
D:\PATIENT\patient\images\sagittal.jpg
>>[03:22:39 GMT+08:00] patient.util.Util >> content loaded
>>[03:22:39 GMT+08:00] patient.util.Util >> getContent - filename:
```

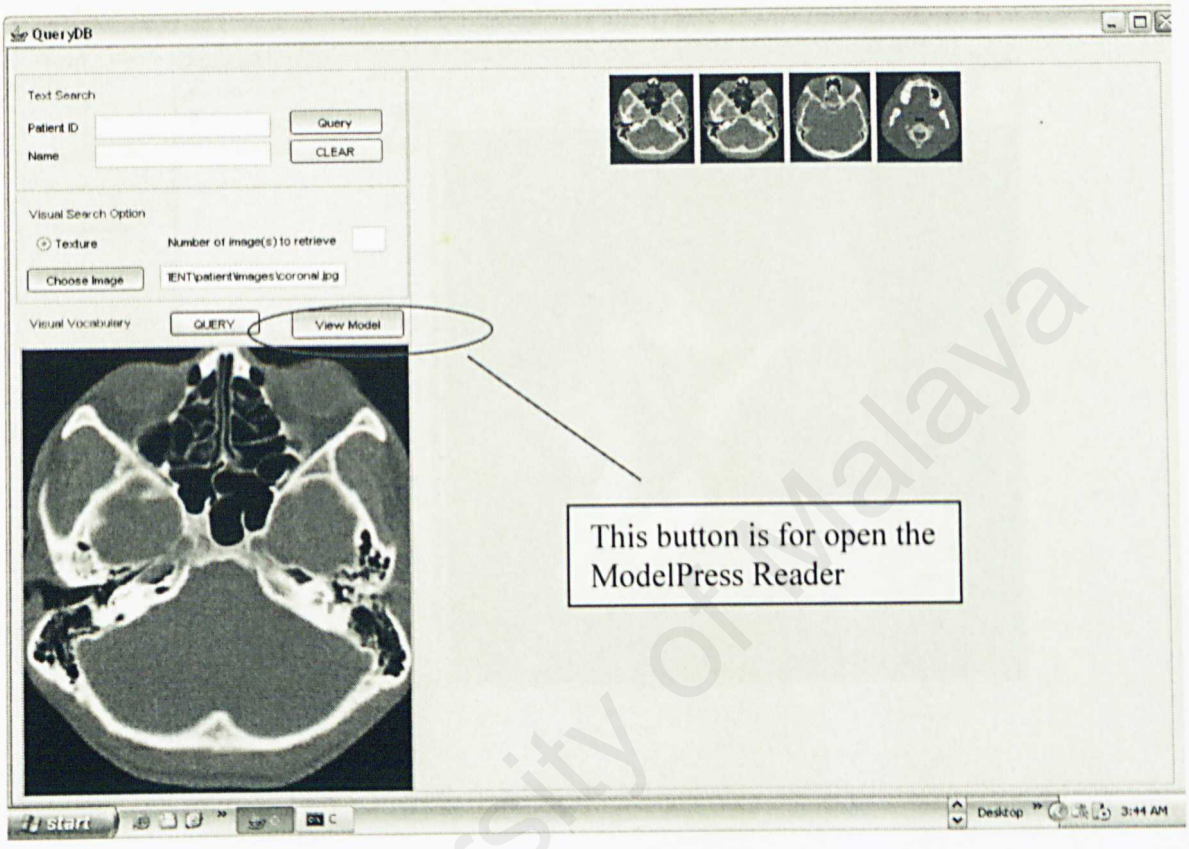
```
D:\PATIENT\patient\images\transvere.jpg
>>[03:22:39 GMT+08:00] patient.util.Util >> content loaded
>>[03:22:39 GMT+08:00] patient.request.process.ProcessPatientEntry >> execute
>>[03:22:39 GMT+08:00] patient.request.process.ProcessPatientEntry >> checkPatientData
>>[03:22:39 GMT+08:00] patient.request.process.ProcessPatientEntry >> getDBConnection
>>[03:22:39 GMT+08:00] patient.util.Util >> getDBConnection
>>[03:22:39 GMT+08:00] patient.util.Util >> using jdbcURL: jdbc:db2:medical
>>[03:22:40 GMT+08:00] patient.util.Util >> connection opened
>>[03:22:40 GMT+08:00] patient.request.process.ProcessPatientEntry >> insertPatient -
patientId: 20042005
>>[03:22:40 GMT+08:00] patient.request.process.ProcessPatientEntry >> prepare statement:
INSERT INTO
PATIENT(PATIENTID,coronalview,sagittalview,transverseview,name,age,race) VALUES
('20
042005',MMDBSYS.DB2Image(CURRENT SERVER,CAST(? AS
BLOB(18687)),'JPG',1,CAST(NULL AS LONG
VARCHAR),''),MMDBSYS.DB2Image(CURRENT SERVER,CAST(? AS
BLOB(11883)),'JPG',1,CAST(NULL AS LONG VA
RCHAR),''),MMDBSYS.DB2Image(CURRENT SERVER,CAST(? AS
BLOB(14295)),'JPG',1,CAST(NULL AS LONG VARCHAR),''),'Wong Chung
Hiong','23','Chinese')
>>[03:22:41 GMT+08:00] patient.request.process.ProcessPatientEntry >> statement prepared:
INSERT INTO
PATIENT(PATIENTID,coronalview,sagittalview,transverseview,name,age,race) VALUES ('2
0042005',MMDBSYS.DB2Image(CURRENT SERVER,CAST(? AS
BLOB(18687)),'JPG',1,CAST(NULL AS LONG
VARCHAR),''),MMDBSYS.DB2Image(CURRENT SERVER,CAST(? AS
BLOB(11883)),'JPG',1,CAST(NULL AS LONG V
ARCHAR),''),MMDBSYS.DB2Image(CURRENT SERVER,CAST(? AS
BLOB(14295)),'JPG',1,CAST(NULL AS LONG VARCHAR),''),'Wong Chung
Hiong','23','Chinese')
```

```
>>[03:22:41 GMT+08:00] patient.request.process.ProcessPatientEntry >> execute statement
>>[03:22:47 GMT+08:00] patient.request.process.ProcessPatientEntry >> statement executed
>>[03:22:47 GMT+08:00] patient.request.process.ProcessPatientEntry >> statement closed
>>[03:22:47 GMT+08:00] patient.request.process.ProcessPatientEntry >> Patient Info inserted
>>[03:22:47 GMT+08:00] patient.request.process.ProcessPatientEntry >> connection
committed
>>[03:22:47 GMT+08:00] patient.request.process.ProcessPatientEntry >> connection closed
>>[03:22:47 GMT+08:00] patient.request.process.ProcessPatientEntry >> Patient created
Patient table populated
```

In order to retrieve the image, run the QueryDB.java. the following interface will prompt out. This interface can use to retrieve images by text search and texture search. The results sometime are not accurate or correct.



Choose an image and click the QUERY button. This will perform the CBIR method based on texture. The results sometimes are not accurate.



After click the view model button, the following application will prompt out.