Project Report On

Content Base Image Retrieval (CBIR)

Akanksha Avinash (12300106)

Dharti Rathod (84213)

Karthikeyan Sivanandi (12200010)

Slesha Vemuganti (82938)

Sneha Gang (84114)

Trupti Kholiya (84829)

INTERNATIONAL TECHNOLOGICAL UNIVERSITY

SEN 942– Advance Software Engineering

Spring 2014

Acknowledgement

We would like to express our gratitude to our professor, Dr. Richard Riehle, Core Faculty, Computer Science, International Technological University, San Jose CA for his valuable guidance, inspiration and suggestions that helped us in the preparation of this project and without which the accomplishment of this task would have never been possible. We also thank him for giving us this opportunity to explore into the real world and understand different Advanced Software Engineering process and techniques. We thank our colleagues who helped in successful completion of the project.

Akanksha Avinash (12300106)
Dharti Rathod (84213)
Karthikeyan Sivanandi (12200010)
Slesha Vemuganti (82938)
Sneha Gang (84114)
Trupti Kholiya (84829)

INDEX

ABSTRACT
(CONTENT BASED IMAGE RETRIEVAL)

Content-based image retrieval uses the visual contents of an image such as color, shape, texture, and spatial layout to represent and index the image. In typical Content based image retrieval systems, the visual contents of the images in the database are extracted and described by multi-dimensional feature vectors. The feature vectors of the images in the database form a feature database. To retrieve images, users provide the retrieval system with example images or sketched figures. The system then changes these examples into its internal representation of feature vectors. The similarities /distances between the feature vectors of the query example or sketch and those of the images in the database are then calculated and retrieval is performed with the aid of an indexing scheme. The indexing scheme provides an efficient way to search for the image database. Recent retrieval systems have incorporated users relevance feedback to modify the retrieval process in order to generate perceptually and semantically more meaningful retrieval results.

1. PROJECT CONCEPT

Introduction to CBIR

As processors become increasingly powerful, and memories become increasingly cheaper, the deployment of large image databases for a variety of applications have now become realizable. Databases of art works, satellite and medical imagery have been attracting more and more users in various professional fields: for example, geography, medicine, architecture, advertising, design, fashion, and publishing. Effectively and efficiently accessing desired images from large and varied image databases is now a necessity.

Definition - CBIR or **Content Based Image Retrieval** is the retrieval of images based on visual features such as color, texture and shape. Reasons for its development are that in many large image databases, traditional methods of image indexing have proven to be insufficient, laborious, and extremely time consuming. These old methods of image indexing, ranging from storing an image in the database and associating it with a keyword or number, to associating it with a categorized description, have become obsolete. This is not **CBIR**. In CBIR, each image that is stored in the database has its features extracted and compared to the features of the query image. It involves two steps.

1. Feature Extraction: The first step in the process is extracting image features to a distinguishable extent.

2. Matching: The second step involves matching these features to yield a result that is visually similar.

Applications of CBIR

Examples of CBIR applications are:

1. **Crime prevention:** Automatic face recognition systems, used by police forces.
2. **Security Check:** Finger print or retina scanning for access privileges.
3. **Medical Diagnosis:** Using CBIR in a medical database of medical images to aid diagnosis by identifying similar past cases.
4. **Intellectual Property:** Trademark image registration, where a new candidate mark is compared with existing marks to ensure no risk of confusing property ownership.

CBIR Systems

Several CBIR systems currently exist, and are being constantly developed. Examples are:

1. **QBIC** or **Query By Image Content** was developed by IBM, Almaden Research Centre, to allow users to graphically pose and refine queries based on multiple visual properties such as color, texture and shape. It supports queries based on input images, user-constructed sketches, and selected color and texture patterns.
2. **VIR Image Engine** by Virage Inc., like QBIC, enables image retrieval based on primitive attributes such as color, texture and structure. It examines the pixels in the image and performs an analysis process, deriving image characterization features.
3. **VisualSEEK** and **WebSEEK** were developed by the Department of Electrical Engineering, Columbia University. Both these systems support color and spatial location matching as well as texture matching.
4. **The Department of Electrical and Computer Engineering, University of California developed NeTra**. It supports color, shape, spatial layout and texture matching, as well as image segmentation.
5. **MARS** or **Multimedia Analysis and Retrieval System** was developed by the Beckman Institute for Advanced Science and Technology, University of Illinois. It supports color, spatial layout, texture and shape matching.
6. **Viper** or **Visual Information Processing for Enhanced Retrieval** was developed at the Computer Vision Group, University of Geneva. It supports color and texture matching.

2. VISION DOCUMENT

Content-based image retrieval (CBIR), also known as query by image content (QBIC) and content-based visual information retrieval (CBVIR) is the application of computer vision techniques to solve image retrieval problem, that is, the problem of searching for digital imagesin large databases.

Purpose

The area of content-based image retrieval is a hybrid research area that requires knowledge of both computer vision and database systems. Users are exploiting the opportunities to access remotely stored images in all kinds of new and exciting ways. However, this has exacerbated the problem of locating a desired image in a large and varied collection. This led to the rise of new research and development field known as content based image retrieval. CBIR has been suggested for the purpose of mobile learning in academic profession where observational exercises are required, such as the observation and identification of fireflies. CBIR in the medical world has more or less unique characteristic that the important information is highly localized within the image. There are many possible applications in the field of medicine, such as the retrieval of X-rays, infrared images and various fields of research. With increasingly cheaper digital image acquisition and storage devices becoming available the size of image collections has grown rapidly. Many such collections are now organized online in image archives such as those provided by Web shots and Flickr. These databases currently only offer meta data based search options where captions, tagged key words and surround text are analyzed to produce results to a keyword based search.

Content based image retrieval is defined as the task of searching for images within a database, which are visually similar to a given example image. There has been a lot of research into methods and systems to achieve this; however it is a widely accepted fact and a reoccurring point made by many researchers into the area, that there is currently no universally accepted technique.

Through this project we intent to study in detail the retrieval of images on the basis of features automatically extracted from the images themselves. The project reviews details and discusses previous work into sketched based content based image retrieval and closely related topics drawing conclusions as to appropriate techniques and possible areas for focus and further work

Scope

Current image search systems generally rely upon Meta data relating to the images in the database. Our aim with this project is to tough base on below.

1. To develop an image retrieval method that utilizes the layout and the structure of the perceptually correct color within an image to measure the similarity of images.
2. To develop a system which allows query upon the visual content of the images.
3. To identify the limitations of current content based image retrieval systems so that these areas can be improved.
4. To research into the field of content based image retrieval and fast algorithm development.

2.1 Problem Statement

The problem involves entering an image as a query into a software application that is designed to employ CBIR techniques in extracting visual properties, and matching them. This is done to retrieve images in the database that are visually similar to the query image.

2.2 Problem Motivation

Image databases and collections can be enormous in size, containing hundreds, thousands or even millions of images. The conventional method of image retrieval is searching for a keyword that would match the descriptive keyword assigned to the image by a human categorizer. Currently under development, even though several systems exist, is the retrieval of images based on their content, called Content Based Image Retrieval, CBIR. While computationally expensive, the results are far more accurate than conventional image indexing. Hence, there exists a tradeoff between accuracy and computational cost. This tradeoff decreases as more efficient algorithms are utilized and increased computational power becomes inexpensive.

2.3 Proposed System:

1. Content-based image retrieval (CBIR), also known as content-based visual information retrieval (CBVIR) is the application of computer vision to the image retrieval problem, that is, the problem of searching for digital images in large databases.

2. "Content-based" means that the search will analyze the actual contents of the image. The term 'content' in this context might refer to colors, shapes, textures, or any other information that can be derived from the image itself. Without the ability to examine image content, searches must rely on metadata such as captions or keywords, which may be laborious or expensive to produce.

3. REQUIREMENTS OVERVIEW

The requirements analysis phase of this document primarily summarizes the functional and nonfunctional requirements along with the organizational and derived requirements.

3.1 Functional Requirements

Capability to Extract Image Data

The proposed software system should have the capability to extract data from images. This is the challenging part of the project and involves complex image processing techniques and algorithms which are the key components for getting the accurate results. This extracted data is used to make the index for the particular image and is later considered as a search key while searching the images.

Ability to create a data Structure

From the image data that was generated from the above step (Extracting data from Images), a data structure should be created. This data structure should be compact and should be created such that a very fast searching algorithm could invoke on that. Creation of this data structure is the key challenge for this project. The scope of the searching algorithm is based on these data structure.

Ability to search the input image

In this step, we make a search by providing the input image in the search space; the system should accept the image and based on the search key it should find the exact or the best matching images. If the exact matching images are not found, then the using the image processing methodologies and approximation capabilities it should be able to find the closest matching images based on the indexes.

Flexibility to change the search space

The proposed system should have the flexibility to update or change the search space as needed by the administrator. Admin should be able to add or remove the image contents from the search space as desired.

User Interface requirements

Initially the user is given an interface to input a keyword. Then the search space is searched according to that keyword (In order to do that each and every image of the search space must be tagged with a keyword) and is presented to the end user. Then the end user can select an image from that set to do a similarity search. So this selected image will be the input query to the search engine. An administrative interface is also available for administrative purposes like changing the search space.

3.2 Non-Functional Requirements

Performance Requirement

Performance is the important requirement in this system. For the search engine it is mandatory to have quick response time, this will be a major challenge during the implementation of software as extracting data from images involve a lot of image processing and it contains a lot of I/O overhead and resource utilization, the applications should be properly tuned to meet the expected response time.

Safety Requirements

The proposed system should implement the data security by restricting the access to the key data for only authorized users such as the administrator having the capability to remove or add images to search space but not the other end users.

Security Requirement

Data should be well secured. It is highly recommended that end user must not be able to do any changes to the system and obtain any valuable system data from the system.

Software Quality Attributes

The system should be user friendly and the response must be given in an average time. System should also be flexible to handle updates.

3.3 Organizational and Derived Requirements

Organizational requirements

These are the requirements specific to the organization. The application should be thoroughly quality tested. The proposed system should be load and stress tested and should be capable to handle bulk users simultaneously without degradation of response time. The change management policies of the organization should be followed. This proposed system should meet the legal policy requirement and ethical discipline of the organization.

Derived requirements

These are emerged in concert with the decomposition of the system. This contrasts with a functionally based approach, where the partitioning occurs without regard to any architectural decomposition, and refinement of the system-level requirements. The aim of this report is to review the current state of the art in content-based image retrieval (CBIR), a technique for retrieving images on the basis of automatically-derived features such as

Primitive features: color, texture and shape or spatial location of image elements.

Derived or logical features: involves degree of logical interference about identity of objects depicted in image. Ex.: Retrieval of objects of a given type and retrieval of individual objects or persons.

Abstract attributes: Involves significant amount of high level reasoning about the meaning and purpose of objects or scenes depicted. Ex: Retrieval of named events, retrieval of pictures with emotional significance.

3.4 Real world Requirements

Building real world systems include regular user feedback during development phase, as required in any other software development life cycle. Only few image retrieval systems are deployed for public usage, Example Google Images or Yahoo Images (which are based primarily on surrounding meta-data rather than content). There are, however, a number of propositions for real-world implementation. It is interesting to note that CBIR has been applied to many diversified fields like Botany, Astronomy, Mineralogy, and remote sensing. There is a good

chance that CBIR based real-world systems will diversify and expand further. Various fields where we can implement CBIR in real world are IRM-based publicly available similarity search tool on an online database of over 800,000 airline-related images. The integration of similarity search functionality to a large collection of art and cultural images. Based on our experience with implementing CBIR systems on real world data for public usage, we list here some of the issues that we found to be critical for real-world deployment.

Performance: The most critical issue is the quality of retrieval and how relevant it is to the domain- specific user community. Most of the current effort is concentrated on improving performance in terms of their precision and recall.

Semantic learning: To handle the problem of semantic gap faced by CBIR, learning image semantics from training data and developing retrieval mechanisms to efficiently leverage semantic estimation are important directions.

Volume of Data: Public image databases tend to grow into unwieldy proportions. The software system must be able to efficiently handle indexing and retrieval at such high scale.

Heterogeneity: If the images originate from diverse sources, parameters such as quality, resolution and color depth are likely to vary. This in turn causes variations in color and texture features extracted. The systems can be made more robust by suitably handling these variations.

Concurrent Usage: In on-line image retrieval systems, it is likely to have multiple concurrent users. While most systems have high resource requirements for feature extraction, indexing etc., they must be efficiently designed so as not to exhaust the host server resources. Alternatively, a large amount of resources must be allocated.

Multi-modal features: The presence of reliable meta- data such as audio or text captions associated with the images can help understand the image content better, and hence leverage the retrieval performance. On the other hand, ambiguous captions may actually add to the confusion, in which case the multi-modal features together may be able to resolve the ambiguity.

User-interface: As discussed before, a greater effort is needed to design intuitive interfaces for image retrieval such that people are actually able to use the tool to their benefit.

Operating Speed: Time is critical in on-line systems as the response time needs to be low for good interactivity. Implementation should ideally be done using efficient algorithms, especially for large databases. For computationally complex tasks, off-line processing and caching the results in parts is one possible way out.

System Evaluation: Like any other software system, image retrieval systems are also required to be evaluated to test the feasibility of investing in a new version or a different product. The design of a CBIR benchmark requires careful design in order to capture the inherent subjectivity in image retrieval.

4. FEASIBILITY STUDY

A feasibility study is carried out to select the best system that meets performance requirements. The main aim of the feasibility study activity is to determine whether it would be financially and technically feasible to develop the product. The feasibility study activity involves the analysis of the problem and collection of all relevant information relating to the product such as the different data items which would be input to the system, the processing required to be carried out on these data, the output data required to be produced by the system as well as various constraints on the behavior of the system.

4.1 Technical Feasibility

CBIR is currently aimed to be stand-alone server based image retrieval system. The communication of data is done through Java based API and image details are sent to the server where it is processed and matched with database images. Being able to communicate through client and server based model, CBIR is considered to be technically strong in terms of processing of data to distant locations. User can send image as query and retrieves results in fraction of seconds.

The facility to produce output in given time is considered as second most important factor for any software product. As CBIR is server based tool, it runs considerably fast when size of database is small. Once there are more images accumulating to the database, performance of the system decreases down as there are many images to be matched with. Response time under this condition also increases as increasing database size.

As with upcoming better and efficient approaches and implementing those in our system, we can assure better performance by processing certain high volume of image data at a higher speed.

4.2 Economic Feasibility

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. More commonly known as Cost / Benefit analysis, the procedure is to determine the benefits and savings that are expected from a proposed system and compare them with costs. If benefits outweigh costs, a decision is taken to design and implement the system. Otherwise, further justification or alternative in the proposed system will have to be made if it is to have a chance of being approved. This is an outgoing effort that improves in accuracy at each phase of the system life cycle.

As CBIR is server based database search system, the only cost associated with it is maintenance of server and memory space requirement. For now, stand-alone version of CBIR works perfectly with limited storage space of few GB of data. Further implementation like online version of system would require better performance and maintenance round the clock. By comparing cost, users get profit of saving time and money for manual search of images. Overall, this system is economic feasible for the development.

4.3 Operational Feasibility

This is mainly related to human organizational and political aspects. This feasibility study is carried out by a small group of people who are familiar with information system technique and are skilled in system analysis and design process. Proposed projects are beneficial only if they can be turned into information system that will meet the operating requirements of the organization. This test of feasibility asks if the system will work when it is developed and installed.

CBIR is used in varieties of functional and application areas when input data is image instead of words. Search engines like Google, Bing etc are considered as pioneer applications in the current era for text based searching. The limitation of these systems is it doesn't take image data for searching with other images. CBIR can be applied in such case when user wants to quickly search for similar images in huge amount of database. In this regard, it is useful for forensic applications in which police department wants to quickly find history of criminals with their faces.

Operational skills required for dealing with this system are very limited. A programmer and database administrator are enough for maintaining the database and assuring the performance and quality of the system.

5. PROJECT PLAN

5.1 Project management plan

Project management involves the planning, monitoring, and control of the people, process, and events that occur as our "Content based image search project" evolves from a preliminary concept to full operational deployment. Apart from team leader, everyone "manages" to some extent, but the scope of management activities varies among each team members involved in our project. Project plan is produced as management activities commence. The plan defines the process and tasks to be conducted, the people who will do the work, and the mechanisms for assessing risks, controlling change, and evaluating quality.

5.2 Coordination and Communication

There are many reasons that our software projects get into trouble. The scale of development efforts is large, leading to complexity, confusion, and significant difficulties in coordinating team members. Uncertainty is common, resulting in a continuing stream of changes that ratchets the project team. To deal with it effectively, we established effective methods for coordinating the people who do the work. To accomplish this, we each members of our software team share ideas on an ad hoc basis, ask for help as problem arise, and interact with one another on a daily basis via Google drive/doc, mobile chats, in person meeting at university as a group.

5.3Team Structure

Name	Student Id	Role
AkankshaAvinash	12300106	Team Member
DhartiRathod	84213	Team Member
KarthikeyanSivanandi	12200010	Team Member
Slesha	82938	Team leader
Sneha Gang	84114	Team Member
TruptiVala	84829	Team Member

5.4 Project Responsibilities

Team Leader

Leader's responsibility is to ensure that tasks assigned to every team member is performed well and on time, which is very smoothly been taken care by our Team leader other than this her work includes:

1. Planning and coordinating team activities through Weekly Meetings in university, mobile messenger, Emails.
2. Providing feedback about team progress to everyone.

Team Member

Each member's (listed in above table) responsibility is to ensure that a task assigned to him/her by the team lead is performed well and on time. The tasks needed to be done and the topics that are important to our project was discussed with every group member in the meeting held every week and then the task distribution was done. Each member of "content based image search project" was responsible for:

1. Assisting Team Leader by signaling problems in an early stage·
2. Executing plans made by the Team Leader.
3. Keeping track of time spent on various tasks.
4. Following procedures and plans.

5.5 Risk Associated

The risks associated while working in team for "Content Based Image Search Project" are:

Miscommunication

Team members might not clearly understand the scope & requirements of our project. Maintaining good communication at every point till the end of project is very important to every team member. It clarifies the issues in the project.

1. **Prevention:** To prevent this each group member decided that after every meeting, one team member creates a 'minutes of meeting' report. Every person in the team has got a copy of this report. Team members should not hesitate to ask and questions if things are unclear.
2. **Correction:** When it becomes clear that miscommunication is causing problems, the team members and the Team Leader are gathered in a meeting to clear things up.

Time shortage

After making good planning to our project, knowing how long and how much time each task in a project takes, at some point we have shortage of time.

1. **Prevention:** Care is taken to plan enough spare time through many numbers of meetings and being dedicated to give more effort on the task allotted.
2. **Correction:** When tasks was failed to be finished in time or when they are finished earlier than planned, the project planning is adjusted. If time shortage becomes severe, user requirements, which have low priority, are dropped after consultation with Team Leader and other team members.

Design Errors

1. **Prevention:** The design should be reviewed very critically. Each member should be consulted frequently on his opinion about the feasibility and the correctness of certain design decisions.
2. **Correction:** When errors in the design are noticed the member should be consulted to help correct the design errors as soon as possible. Also all the work that depends on the faulty design should be halted until the error is corrected.

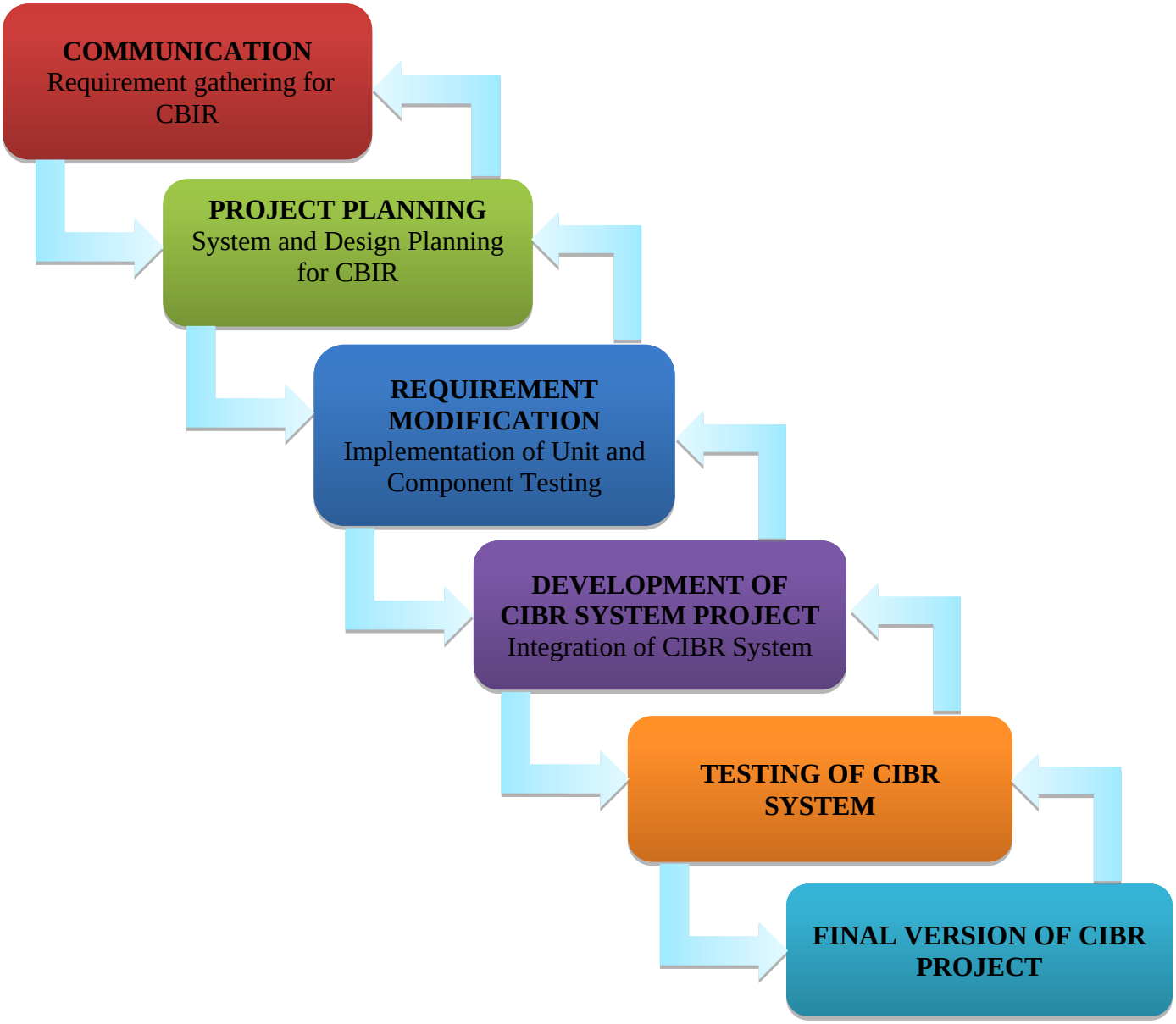
Illness or absence of team members

1. **Prevention:** Team members should inform the team leader timely before a planned period of absence.
2. **Correction:** By ensuring that knowledge is shared between team members, work can be taken over quickly by someone else if a person gets ill. When work needs to be taken over by someone else, a re-division is made on his/her other tasks so that the workload does not get too high.

6. PROCESS MODEL

Waterfall Model

This waterfall approach ensures that the big picture of site functionality and for this project we would be using Waterfall methodology, but in real/virtual world we would prefer agile methodology.



Agile Process Model

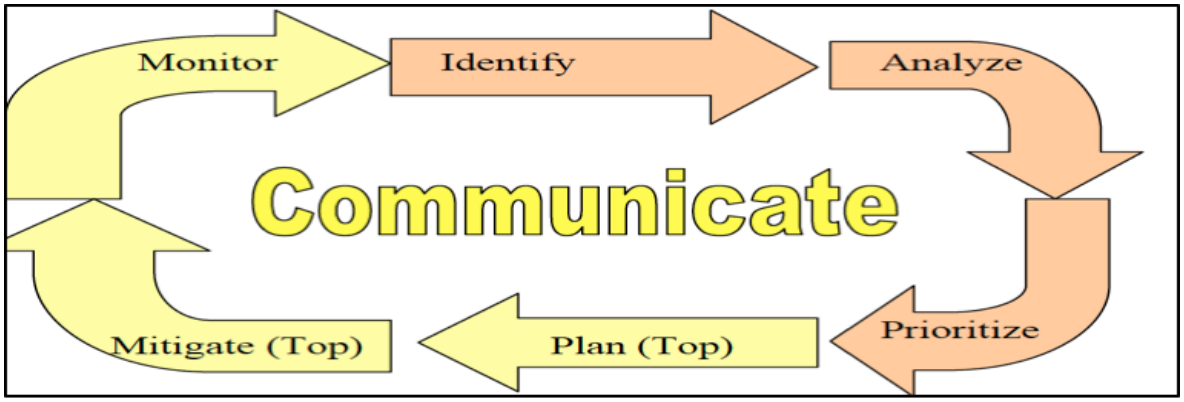
It's the business importance that makes CBIR the right fit for Agile development. Our CBIR sites need to respond in support of the search of Images based on Image recognition system. The industrial scale of our CBIR will Practice's agile development methods, which will help organization, create environments that are flexible, responsive and ready for the future.

Agile Development offers three important differentiators.

1. High Velocity and availability,
2. High Quality- 24/7 response time is essential for image search,
3. Needs of people searching.

7. RISK MANAGEMENT

Risk management is the identification, assessment, prioritization of risks (potential future harm that may arise due to some present action) and eliminates risk items before they either become a threat to the successful operation of the software or requires expensive and time-consuming rework. It can be broken down into different phases as mentioned in the diagram below.



7.1 Phases of Risk Management

Essentially these phases can be grouped together into risk assessment (identify, analyze and prioritize) and risk control (monitor, plan and mitigate).

Risk Identification:

There are risks involved at every step of the project. The sooner they're identified the easier it gets to mitigate them. Some risks are generic in nature and apply to all the systems equally. For instance, risks like people, size, technology tools, customer, sales, estimation etc. Other risks can be more product-related. For instance, user's account getting hacked into, database size, providing irrelevant data to the users etc. If a product is made for an enterprise, business risks are also something to be taken care of. Risks like, what is the viability of software, problems in maintaining software when an upgrade is required etc.

Analyze:

After risks have been identified and enumerated, the next step is risk analysis. Through risk analysis, we transform the risks that were identified into decision-making information. In turn, each risk is considered and a judgment made about the probability and the seriousness of the risk.

Prioritize:

After identifying and analyzing all the risks, the team prioritizes the risks by ranking them. It is too costly and perhaps even unnecessary to take action on every identified risk. Some of them have a very low impact or a very low probability of occurring – or both. Through the prioritization process, the team determines which risks it will take action on.

Very Low	< 10 %
Low	10 – 25 %
Moderate	25 – 50 %
High	50 – 75 %
Very High	> 75 %

The above table is one way of prioritizing risks on a very high level. If a possibility of a risk happening is less than 10%, it qualifies as a very low priority risk and if it's more than 75%, it qualifies as very high priority risk.

Planning:

This phase basically plans for 2 things. Firstly, how can we prevent the risk from becoming a greater threat or materializing it? Secondly, come up with a contingency plan. That is, what needs to be done if a certain risk materializes?

Mitigate:

Related to risk planning, through risk mitigation, the team develops strategies to reduce the possibility or the loss impact of a risk. Risk mitigation produces a situation in which the risk items are eliminated or otherwise resolved.

Monitor:

After risks are identified, analyzed, and prioritized, and actions are established, it is essential that the team regularly monitor the progress of the product and the resolution of the risk items, taking corrective action when necessary. This monitoring can be done as part of the team project management activities or via explicit risk management activities.

7.2 Risks Involved & Mitigation Strategies

Based on the above approach we have identified following to be the risks for content based image retrieval system along with the strategies to mitigate that risk. These can further be subdivided into technical and non-technical risks.

Non-Technical risks involved:

1. Understanding user needs – For this system, a user is a consumers of the system. Software should be made keeping the user requirements in mind and it should be very easy to understand and use. If the system does not do what the user expects or is showing irrelevant results or is too complex to use; it can become obsolete in no time.

2. Illness - It is impossible to totally ensure that I do not get ill. Ensuring that I lead a healthy lifestyle so that the chances of getting ill are reduced and any contracted illnesses are reduced in severity. Ensure work is started with plenty of time to spare should reduce the effect of time lost due to illness.

3. Other work reduces time available for the project - Ensuring that work on this project is done with priority and started as early as possible should ensure that possible delays from other work are kept to a minimum.

4. Budget for the system – Allocating appropriate budget in each department (requirements, development, testing etc.) is also a very important. Overestimating or underestimating the budget for any department can have dire consequences and can affect the release of the product. Thorough research needs to be done before finalizing on the budget and underestimating should be avoided at all times. There should always be some surplus amount allocated for unforeseen circumstances.

Technical risks involved

1. Failure to implement a functional algorithm - The incremental approach to development of the algorithm and software for this project should ensure that at a minimum the most basic of retrieval algorithms is implemented successfully.

2. Failure to produce algorithm with a suitably low level of complexity - With each increment the complexity with relation to the speed of execution of the search algorithm can be assessed and improved thereby ensuring that the algorithm runs in the minimum possible time.

3. Failure to develop an algorithm with a suitable degree of accuracy - Experimentation with each increment of the implementation process should provide more understanding and experience leading to an increase in retrieval accuracy with each iteration of the design.

4. Hardware computational power unable to process image data at a required rate - Keeping the computational complexity of the algorithm to a minimum will ensure that the required computational power is kept to a minimum; thereby reducing the chance that this becomes an issue.

5. Time required to develop algorithms at each iteration is underestimated - Experimentation and development can be time consuming and is the main focus of this project. Therefore this task will take priority over others to ensure its completion despite underestimates in time required. In the case of disaster the number of iterations of development can be reduced with a functional if not optimal algorithm still produced.

6. Time required to test and evaluate the performance of each algorithm is underestimated - Testing the algorithms with user input is essential to further development at each iteration, with the results produced providing the information to make changes and draw conclusions. It should therefore take high priority along with algorithm development to ensure that it gets completed ahead of other not so essential tasks.

7. Time to develop user interface is underestimated - The importance of the user interface is secondary to algorithm development, but essential to allow easy testing and further development to the project. It should therefore be prioritized once initial core components are implemented to ensure that it is completed.

8. Image database grows too large - Initially the image database used will not be so large that full implementation of a database rational will be required. If the database grows too large then full database implementation will have to be undertaken.

9. Computer Hardware Failure - Ensure that the tools used for the development of the project are available on a number of machines to maintain access to the development tools in the case that one machine should suffer failure.

10. Inherent variability in image findings - While computer-driven image descriptors and similarity analyses can help reduce the variability, it can be conjectured that image-only approaches have limited resolving power.

11. Tools for data collection - Fortunately, computer hardware and are no longer barriers to creation of large repositories of data. However, data must be entered systematically and efficiently to achieve collections with millions of images. Automation of segmentation (if needed) or interactive tools requiring minimal user input will be critical. This can be decisive in retrieving relevant results from the database.

12. Threat from hackers – Proper encryption technologies should be kept in mind at all times while developing the system so that we are sure of building a secure system. If the system is not secure enough some hacker can get access to the database and can alter or delete the data.

13. DoS attack – There's always a risk of denial of service attack where a huge number of automated fake request hit the server causing it go slow in response and eventually crashing the server. This is a very high-risk because it can cause the website to not respond. Steps should be taken to identify stop these fake requests to even reach the server. Lately, there has been a captcha image approach used by a majority of websites to make sure request is coming from a human source and it is legitimate.

Although not a risk, but one caveat here is that the storage, retrieval and use of images in some of these application areas (such as surveillance video monitoring) can have significant privacy and civil liberties implications. Such implications need to be explored thoroughly before any decision on adoption is taken. It is also unlikely that general-purpose image retrieval software will meet the needs of these user communities without a significant degree of customization. Each of these application areas has their own range of special needs and constraints. Software solutions that fail to address these needs are unlikely to perform well enough to convince users that they are worth adopting.

8. SYSTEM DESIGN OVERVIEW

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering.

Object-oriented analysis and design methods are becoming the most widely used methods for computer systems design. The UML has become the standard language in object-oriented analysis and design. It is widely used for modeling software systems and is increasingly used for high designing non-software systems and organizations.

Logical design

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modeling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems design are included. Logical design includes ER Diagrams i.e. Entity Relationship Diagrams.

Physical design

The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified/authenticated, how it is processed, and how it is displayed as In Physical design, the following requirements about the system are decided.

- Input requirement
- Output requirements
- Storage requirements
- Processing Requirements
- System control and backup or recovery.

Put another way, the physical portion of systems design can generally be broken down into three sub-tasks:

- User Interface Design
- Data Design

- Process Design

User Interface Design is concerned with how users add information to the system and with how the system presents information back to them. Data Design is concerned with how the data is represented and stored within the system. Finally, Process Design is concerned with how data moves through the system, and with how and where it is validated, secured and/or transformed as it flows into, through and out of the system. At the end of the systems design phase, documentation describing the three sub-tasks is produced and made available for use in the next phase.

Physical design, in this context, does not refer to the tangible physical design of an information system. To use an analogy, a personal computer's physical design involves input via a keyboard, processing within the CPU, and output via a monitor, printer, etc. It would not concern the actual layout of the tangible hardware, which for a PC would be a monitor, CPU, motherboard, hard drive, modems, video/graphics cards, USB slots, etc. It involves a detailed design of a user and a product database structure processor and a control processor. The H/S personal specification is developed for the proposed system.

It is crucial to understand the scope and nature of image data in order to determine the complexity of image search system design. The design is also largely influenced by factors such as the diversity of user-base and expected user traffic for a search system. Along this dimension, search data can be classified into the following categories:

1. Archives - usually contain large volumes of structured or semi-structured homogeneous data pertaining to specific topics.

2. Domain-Specific Collection - this is a homogeneous collection providing access to controlled users with very specific objectives. Examples of such a collection are biomedical and satellite image databases.

3. Enterprise Collection - a heterogeneous collection of images that is accessible to users within an organization’s intranet. Pictures may be stored in many different locations.

4. Personal Collection - usually consists of a largely homogeneous collection and is generally small in size, accessible primarily to its owner, and usually stored on a local storage media.

5. Web - World Wide Web images are accessible to everyone with an Internet connection. These image collections are semi-structured, non-homogeneous and massive in volume, and are usually stored in large disk arrays.

In this Approach we convert image feature vectors to weighted-term vectors and extract the frequently occurred feature vectors by implementing relevance feedback technique in content-based image retrieval to demonstrate the efficiency of this conversion. According to obtain effectiveness we extract color and texture features because of using a single feature extraction does not give high accuracy and effectiveness. High Dimensional feature reduce the query efficiency and low level feature reduce the query accuracy. So we have to integrate those above two visual features in our approach to obtain high effectiveness

8.1 Software Architecture

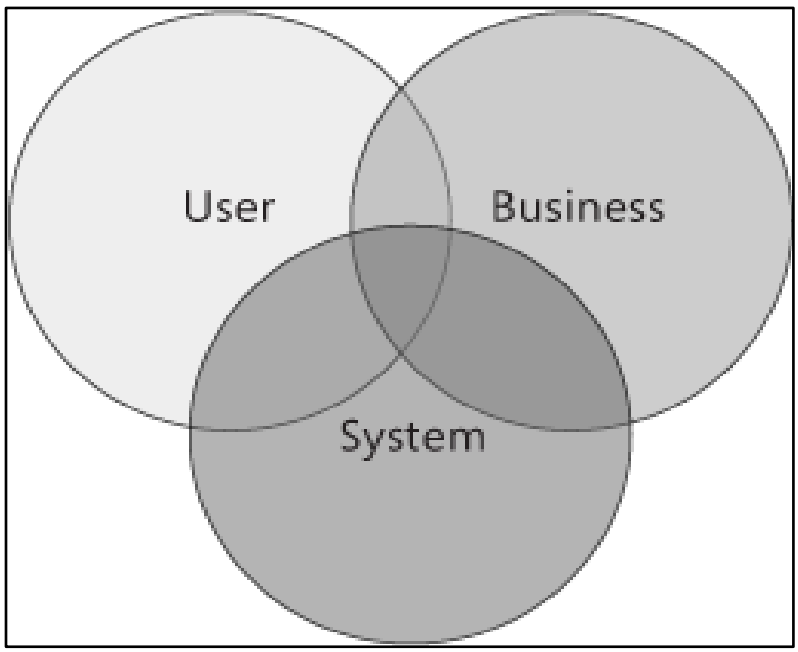
Software application architecture is the process of defining a structured solution that meets all of the technical and operational requirements, while optimizing common quality attributes such as performance, security, and manageability. It involves a series of decisions based on a wide range of factors, and each of these decisions can have considerable impact on the quality, performance, maintainability, and overall success of the application.

Software architecture encompasses the set of significant decisions about the organization of a software system including the selection of the structural elements and their interfaces by which the system is composed; behavior as specified in collaboration among those elements; composition of these structural and behavioral elements into larger subsystems; and an architectural style that guides this organization. Software architecture also involves functionality, usability, resilience, performance, reuse, comprehensibility, economic and technology constraints, tradeoffs and aesthetic concerns.

Like any other complex structure, software must be built on a solid foundation. Failing to consider key scenarios, failing to design for common problems, or failing to appreciate the long term consequences of key decisions can put the application at risk. Modern tools and platforms help to simplify the task of building applications, but they do not replace the need to design the application carefully, based on the specific scenarios and requirements. The risks exposed by

poor architecture include software that is unstable, is unable to support existing or future business requirements, or is difficult to deploy or manage in a production environment.

Systems should be designed with consideration for the user, the system (the IT infrastructure), and the business goals. For each of these areas, we should outline key scenarios and identify important quality attributes (for example, reliability or scalability) and key areas of satisfaction and dissatisfaction. Where possible, develop and consider metrics that measure success in each of these areas.



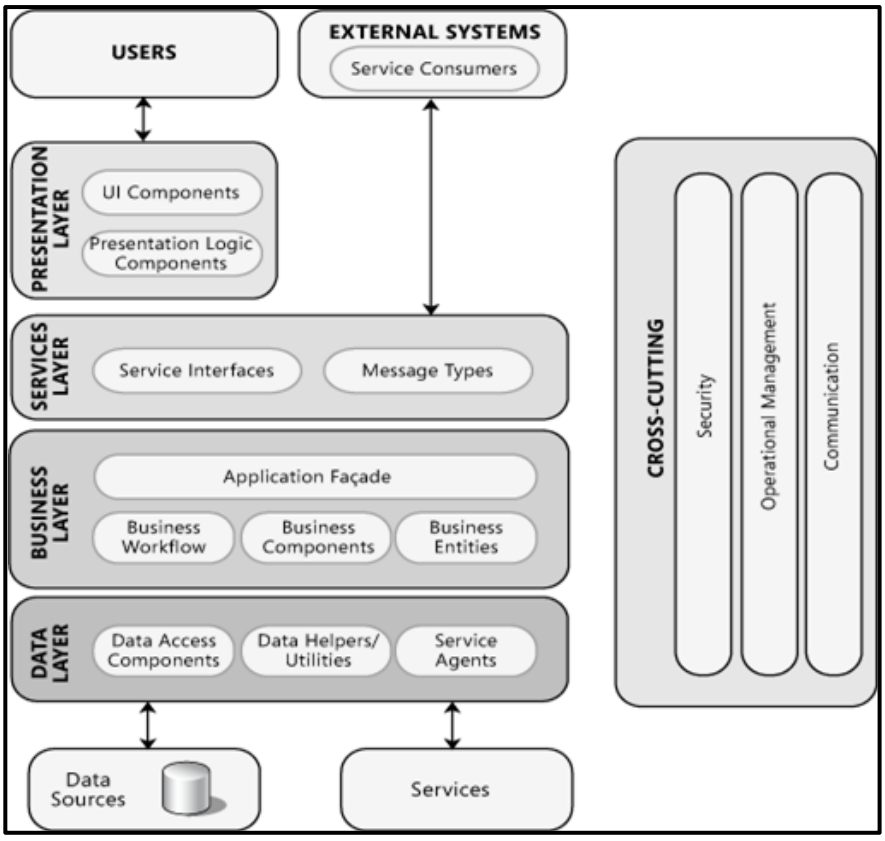
The Goals of Architecture

Application architecture seeks to build a bridge between business requirements and technical requirements by understanding use cases, and then finding ways to implement those use cases in the software. The goal of architecture is to identify the requirements that affect the structure of the application. Good architecture reduces the business risks associated with building a technical solution. A good design is sufficiently flexible to be able to handle the natural drift that will occur over time in hardware and software technology, as well as in user scenarios and requirements. An architect must consider the overall effect of design decisions, the inherent tradeoffs between quality attributes (such as performance and security), and the tradeoffs required to address user, system, and business requirements.

Keep in mind that the architecture should:

- Expose the structure of the system but hide the implementation details.
- Realize all of the use cases and scenarios.
- Try to address the requirements of various stakeholders.
- Handle both functional and quality requirements.

Software architecture is often described as the organization or structure of a system, where the system represents a collection of components that accomplish a specific function or set of functions. In other words, architecture is focused on organizing components to support specific functionality. This organization of functionality is often referred to as grouping components into "areas of concern." Figure 1 illustrates common application architecture with components grouped by different areas of concern.



Key Architecture Principles

Considering the following key principles when designing architecture:

1. **Build to change instead of building to last.** Consider how the application may need to change over time to address new requirements and challenges, and build in the flexibility to support this.

2. **Model to analyze and reduce risk.** Use design tools, modeling systems such as Unified Modeling Language (UML), and visualizations where appropriate to helped capturing requirements and architectural and design decisions, and to analyze their impact. However, do not formalize the model to the extent that it suppresses the capability to iterate and adapt the design easily.

3. **Use models and visualizations as a communication and collaboration tool**. Efficient communication of the design, the decisions you make, and ongoing changes to the design, is critical to good architecture. Use models, views, and other visualizations of the architecture to communicate and share your design efficiently with all the stakeholders, and to enable rapid communication of changes to the design.

4. **Identify key engineering decisions.** Use the information in this guide to understand the key engineering decisions and the areas where mistakes are most often made. Invest in getting these key decisions right the first time so that the design is more flexible and less likely to be broken by changes.

Key Design Considerations

This guide describes the major decisions that you must make, and which help to ensure that you consider all of the important factors as you begin and then iteratively develop your architecture design. The major decisions, briefly described in the following sections, are:

1. Determine the Application Type
2. Determine the Deployment Strategy
3. Determine the Appropriate Technologies
4. Determine the Quality Attributes
5. Determine the Crosscutting Concerns

8.2 Security Component in Design

Content-Based Image Retrieval (CBIR) has been recently used as a filtering mechanism against the piracy of multimedia contents. Many publications in the last few years have proposed very robust schemes where pirated contents are detected despite severe modifications. As none of these systems have addressed the piracy problem from a security perspective, it is time to check whether they are secure

Security Components of any software will be the measures or the techniques one is using to protect the software from some kind of virus attack or miss use of the software due to some loop holes of the product. Now question here arises that

What causes software security problems?

So the answer is all security vulnerabilities in software are the result of security bugs or defects, within the software. In most cases, these defects are created by two primary causes:

1. Non-conformance, or a failure to satisfy requirements; and

2. An error or omission in the software requirements.

Non-conformance, or a failure to satisfy requirements

A non-conformance may be simple–the most common is a coding error or defect–or more complex (i.e., a subtle timing error or input validation error). The important point about non-conformance is that verification and validation techniques are designed to detect them and security assurance techniques are designed to prevent them. Improvements in these methods, through a software security assurance program, can improve the security of software.

Errors or omissions in software requirements

The most serious security problems with software-based systems are those that develop when the software requirements are incorrect, inappropriate, or incomplete for the system situation. Unfortunately, errors or omissions in requirements are more difficult to identify. For example, the software may perform exactly as required under normal use, but the requirements may not correctly deal with some system state. When the system enters this problem state, unexpected and undesirable behavior may result. This type of problem cannot be handled within the software discipline; it results from a failure of the system and software engineering processes which developed and allocated the system requirements to the software.

Software Security Assurance (SSA) is the process of ensuring that software is designed to operate at a level of security that is consistent with the potential harm that could result from the loss, inaccuracy, alteration, unavailability, or misuse of the data and resources that it uses, controls, and protects.

The software security assurance process begins by identifying and categorizing the information that is to be contained in, or used by, the software. The information should be categorized according to its sensitivity. For example, in the lowest category, the impact of a security violation is minimal

There are two basic types of Software Security Assurance activities.

1. Some focus on ensuring that information processed by an information system is assigned a proper sensitivity category, and that the appropriate protection requirements have been developed and met in the system.
2. Others focus on ensuring the control and protection of the software, as well as that of the software support tools and data.

At a minimum, a software security assurance program should ensure that:

1. A security evaluation has been performed for the software.
2. Security requirements have been established for the software.
3. Security requirements have been established for the software development and/or operations and maintenance (O&M) processes.
4. Each software review, or audit, includes an evaluation of the security requirements.
5. A configuration management and corrective action process is in place to provide security for the existing software and to ensure that any proposed changes do not inadvertently create security violations or vulnerabilities.
6. Physical security for the software is adequate.

Building in Security

Improving the software development process and building better software are ways to improve software security, by producing software with fewer defects and vulnerabilities. A first-order approach is to identify the critical software components that control security-related functions and

pay special attention to them throughout the development and testing process. This approach helps to focus scarce security resources on the most critical areas.

Tools and techniques

There are many commercial off-the-shelf**(COTS)** software packages that are available to support software security assurance activities. However, before they are used, these tools must be carefully evaluated and their effectiveness must be assured.

Common weaknesses enumeration

One way to improve software security is to gain a better understanding of the most common weaknesses that can affect software security. With that in mind, there is a current community-based program called the Common Weaknesses Enumeration project, which is sponsored by the Mitre Corporation to identify and describe such weaknesses. The list, which is currently in a very preliminary form, contains descriptions of common software weaknesses, faults, and flaws.

Security architecture/design analysis

Security architecture design analysis verifies that the software design correctly implements security requirements. Generally speaking, there are four basic techniques that are used for security architecture/design analysis.

Logic analysis

Logic analysis evaluates the equation, algorithms, and control logic of the software design.

Data analysis

Data analysis evaluates the description and intended usage of each data item used in design of the software component. The use of interrupts and their effect on data should receive special attention to ensure interrupt handling routines do not alter critical data used by other routines.

Interface analysis

Interface analysis verifies the proper design of a software component's interfaces with other components of the system, including computer hardware, software, and end- users.

Constraint analysis

Constraint analysis evaluates the design of a software component against restrictions imposed by requirements and real-world limitations. The design must be responsive to all known or anticipated restrictions on the software component. These restrictions may include timing, sizing, and throughput constraints, input and output data limitations, equation and algorithm limitations, and other design limitations.

Secure code reviews, inspections, and walkthroughs

Code analysis verifies that the software source code is written correctly, implements the desired design, and does not violate any security requirements. Generally speaking, the techniques used in the performance of code analysis mirror those used in design analysis.

Secure code reviews are conducted during and at the end of the development phase to determine whether established security requirements, security design concepts, and security-related specifications have been satisfied. These reviews typically consist of the presentation of material to a review group. Secure code reviews are most effective when conducted by personnel who have not been directly involved in the development of the software being reviewed.

Informal reviews

Informal secure code reviews can be conducted on an as-needed basis. To conduct an informal review, the developer simply selects one or more reviewer(s) and provides and/or presents the material to be reviewed. The material may be as informal as pseudo-code or hand-written documentation.

Formal review

Formal secure code reviews are conducted at the end of the development phase for each software component. The client of the software appoints the formal review group, who may make or affect a "go/no-go" decision to proceed to the next step of the software development life cycle.

Inspections and walkthroughs

A secure code inspection or walkthrough is a detailed examination of a product on a step-by-step or line-by-line (of source code) basis. The purpose of conducting secure code inspections or walkthroughs is to find errors. Typically, the group that does an inspection or walkthrough is composed of peers from development, security engineering and quality assurance.

Security testing

Software security testing, which includes penetration testing, confirms the results of design and code analysis, investigates software behavior, and verifies that the software complies with security requirements. Special security testing, conducted in accordance with a security test plan and procedures, establishes the compliance of the software with the security requirements. Security testing focuses on locating software weaknesses and identifying extreme or unexpected situations that could cause the software to fail in ways that would cause a violation of security requirements. Security testing efforts are often limited to the software requirements that are classified as "critical" security items.

8.3 Hardware and Software Interface

Real-world applications often demand real-time response. One way to make the increasingly complex image retrieval algorithms practical is to use domain-specific hardware acceleration. Unfortunately, very little has been explored in this direction. The notable few include an FPGA implementation of a color histogram based image retrieval system, an FPGA implementation for sub-image retrieval within an image database, and a method for efficient retrieval in a network of imaging devices. While the focus has generally been on retrieval and annotation performance, presentation of results has often taken a back-seat.

Subjectivity in the needs as well as interpretation of results is an issue. One way around it is to allow for greater flexibility in querying/visualization. Some recent innovations in querying include sketch-based retrieval of color images, querying using 3-D models motivated by the fact that 2-D image queries are unable to capture the spatial arrangement of objects within the image, and a multi-modal system involving hand-gestures and speech for querying and RF. For image annotation systems, one way to conveniently create sufficiently representative manually annotated training databases is by building interactive, public domain games. For designing querying/visualization for image retrieval systems, it helps to understand factors like how people manage their digital photographs or frame their queries for visual art images.

In, user studies on various ways of arranging images for browsing purposes are conducted, and the observation is that both visual feature based arrangement and concept-based arrangement have their own merits and demerits. Thinking beyond the typical grid-based arrangement of top

matching images, spiral and concentric visualization of retrieval results have been explored in. Efficient ways of browsing large images interactively, e.g., those encountered in pathology or remote sensing, using small displays over a communication channel are discussed in. Speaking of small displays, user log based approaches to smarter ways of image browsing on mobile devices have been proposed in. For personal images, innovative arrangements of query results based on visual content, time-stamps, and efficient use of screen space add new dimensions to the browsing experience.

9. CBIS PROTOTYPE

The Software Prototyping refers to building software application prototypes which display the functionality of the product under development but may not actually hold the exact logic of the original software. Software prototyping is becoming very popular as a software development model, as it enables to understand customer requirements at an early stage of development. It helps get valuable feedback from the customer and helps software designers and developers understand about what exactly is expected from the product under development.

Software Prototyping

1. Prototype is a working model of software with some limited functionality.
2. The prototype does not always hold the exact logic used in the actual software application and is an extra effort to be considered under effort estimation.
3. Prototyping is used to allow the users evaluate developer proposals and try them out before implementation.
4. It also helps understand the requirements which are user specific and may not have been considered by the developer during product design.

Following is the stepwise approach to design a software prototype:

1. **Basic Requirement Identification:** This step involves understanding the very basics product requirements especially in terms of user interface. The more intricate details of the internal design and external aspects like performance and security can be ignored at this stage.

2. **Developing the initial Prototype:** The initial Prototype is developed in this stage, where the very basic requirements are showcased and user interfaces are provided. These features may not exactly work in the same manner internally in the actual software developed and the workarounds are used to give the same look and feel to the customer in the prototype developed.

3. **Review of the Prototype:** The prototype developed is then presented to the customer and the other important stakeholders in the project. The feedback is collected in an organized manner and used for further enhancements in the product under development.

4. **Revise and enhance the Prototype:** The feedback and the review comments are discussed during this stage and some negotiations happen with the customer based on factors like, time and budget constraints and technical feasibility of actual implementation. The changes accepted are again incorporated in the new Prototype developed and the cycle repeats until customer expectations are met.

Prototypes can have *horizontal* or *vertical* dimensions. Horizontal prototype displays the user interface for the product and gives a broader view of the entire system, without concentrating on internal functions. A vertical prototype on the other side is a detailed elaboration of a specific function or a sub system in the product.

The purpose of both horizontal and vertical prototype is different. Horizontal prototypes are used to get more information on the user interface level and the business requirements. It can even be presented in the sales demos to get business in the market. Vertical prototypes are technical in nature and are used to get details of the exact functioning of the sub systems. For example, database requirements, interaction and data processing loads in a given sub system.

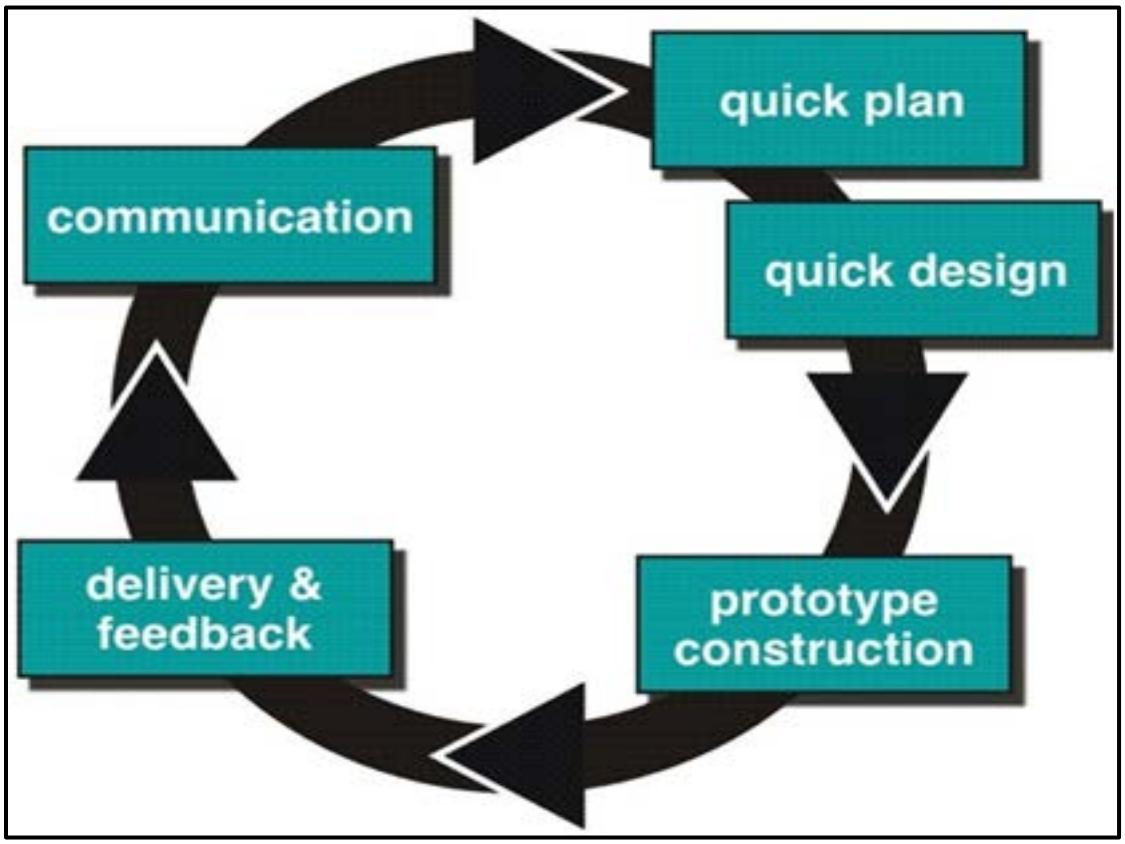


Figure: Software Prototyping

Software Prototyping Types

There are different types of software prototypes used in the industry. Following are the major software prototyping types used widely:

1. Throwaway/Rapid Prototyping: Throwaway prototyping is also called as rapid or close ended prototyping. This type of prototyping uses very little efforts with minimum requirement analysis to build a prototype. Once the actual requirements are understood, the prototype is discarded and the actual system is developed with a much clear understanding of user requirements.

2. Evolutionary Prototyping: Evolutionary prototyping also called as breadboard prototyping is based on building actual functional prototypes with minimal functionality in the beginning. The prototype developed forms the heart of the future prototypes on top of which the entire system is built. Using evolutionary prototyping only well understood requirements are included in the prototype and the requirements are added as and when they are understood.

3. Incremental Prototyping: Incremental prototyping refers to building multiple functional prototypes of the various sub systems and then integrating all the available prototypes to form a complete system.

4. Extreme Prototyping: Extreme prototyping is used in the web development domain. It consists of three sequential phases. First, a basic prototype with all the existing pages is presented in the html format. Then the data processing is simulated using a prototype services layer. Finally the services are implemented and integrated to the final prototype. This process is called Extreme Prototyping used to draw attention to the second phase of the process, where a fully functional UI is developed with very little regard to the actual services.

Software Prototyping Application

Software Prototyping is most useful in development of systems having high level of user interactions such as online systems. Systems which need users to fill out forms or go through various screens before data is processed can use prototyping very effectively to give the exact look and feel even before the actual software is developed.

Software that involves too much of data processing and most of the functionality is internal with very little user interface does not usually benefit from prototyping. Prototype development could be an extra overhead in such projects and may need lot of extra efforts.

Snapshots of CBIR Working Prototype:

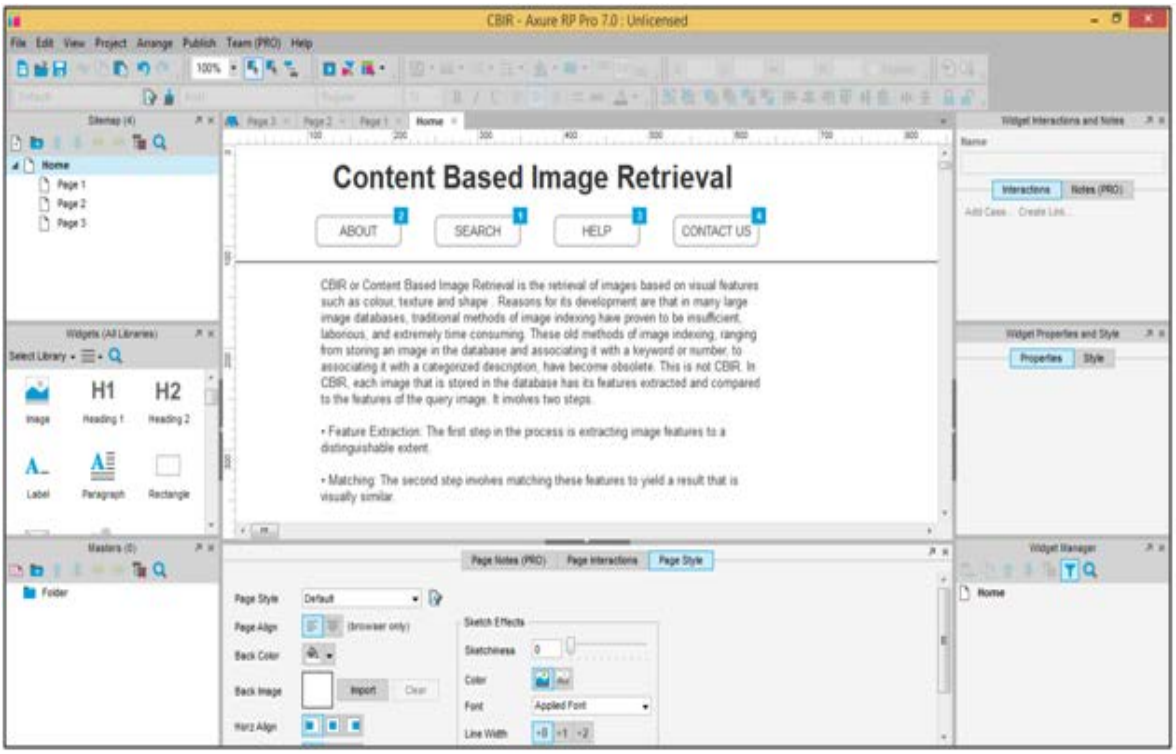


Figure: Development of CBIR prototype using Axure Pro 7.0

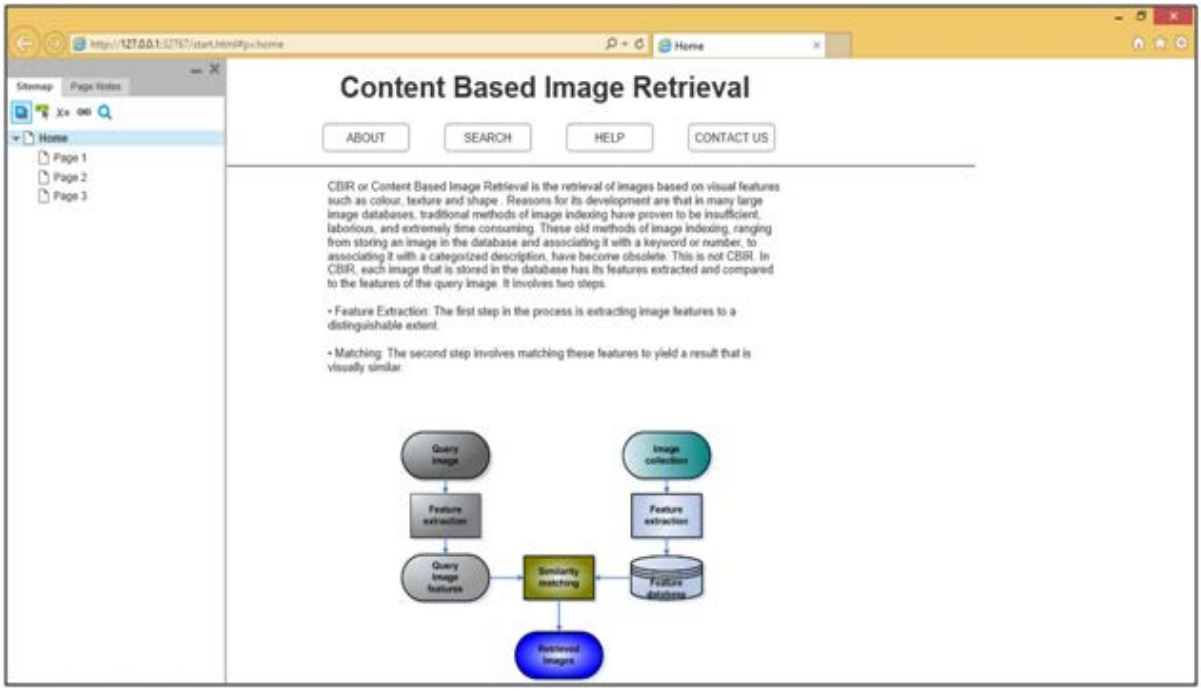


Figure: CBIR Home Page

Figure: CBIR Search Page

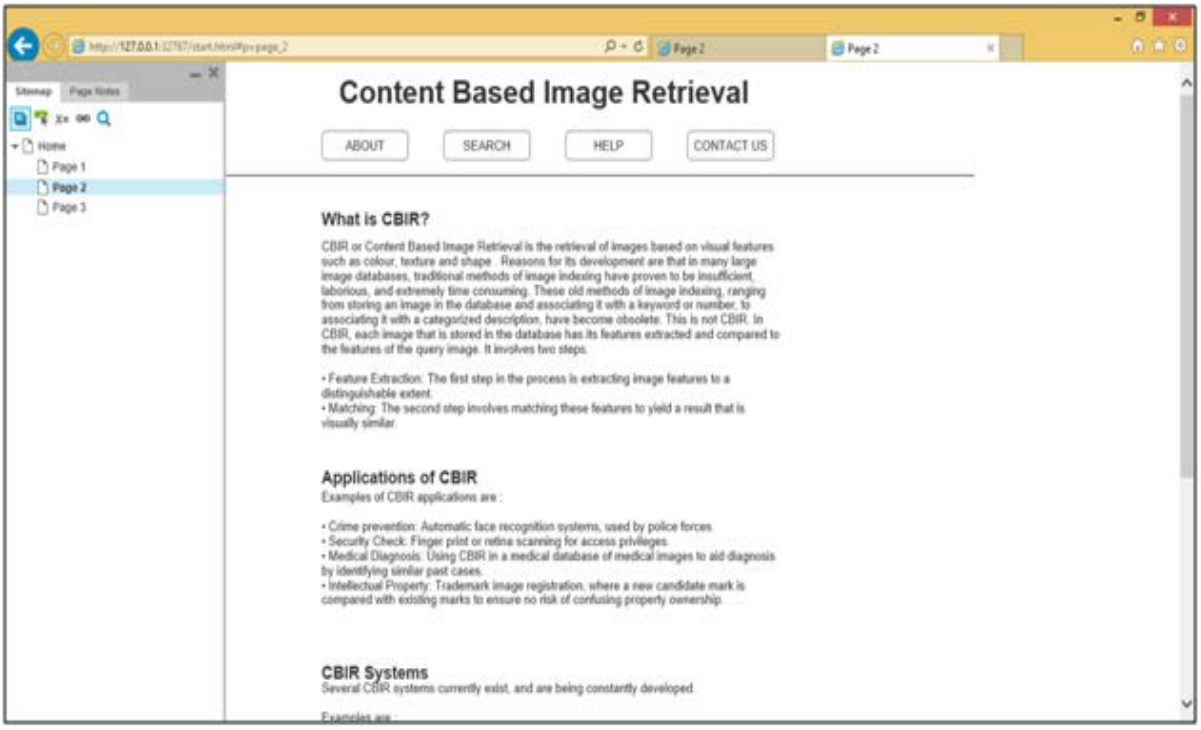
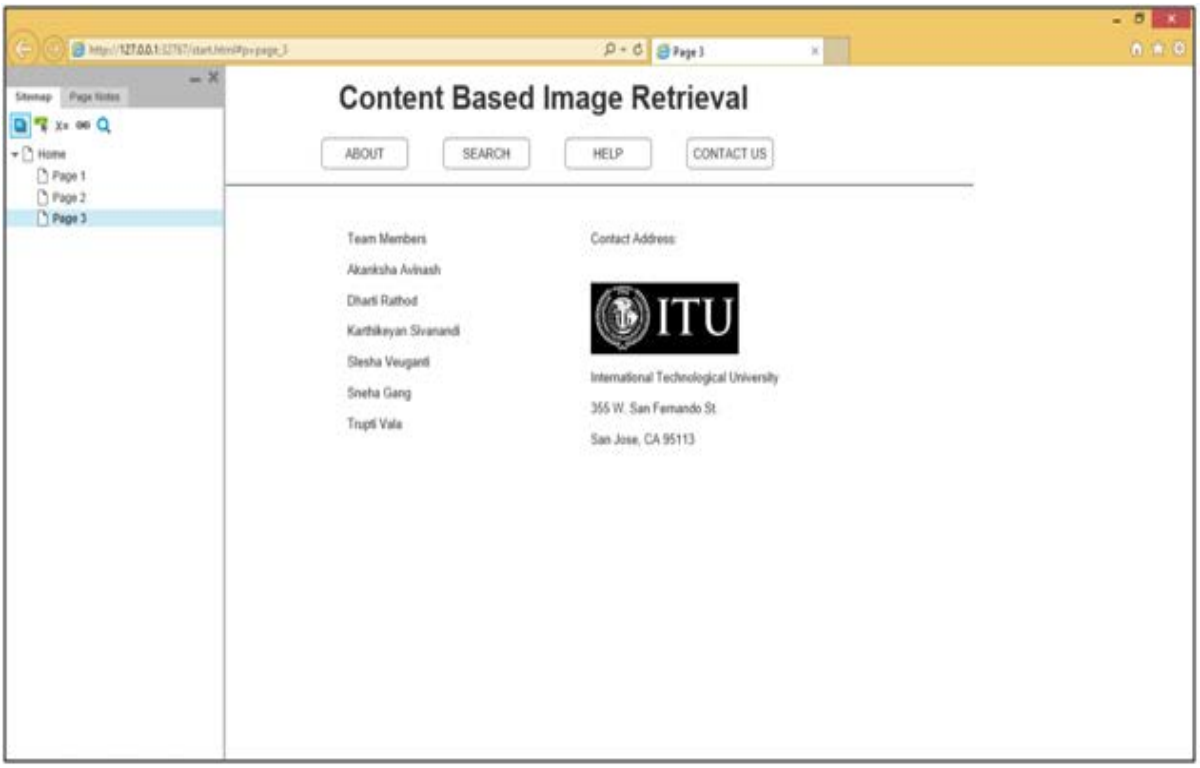


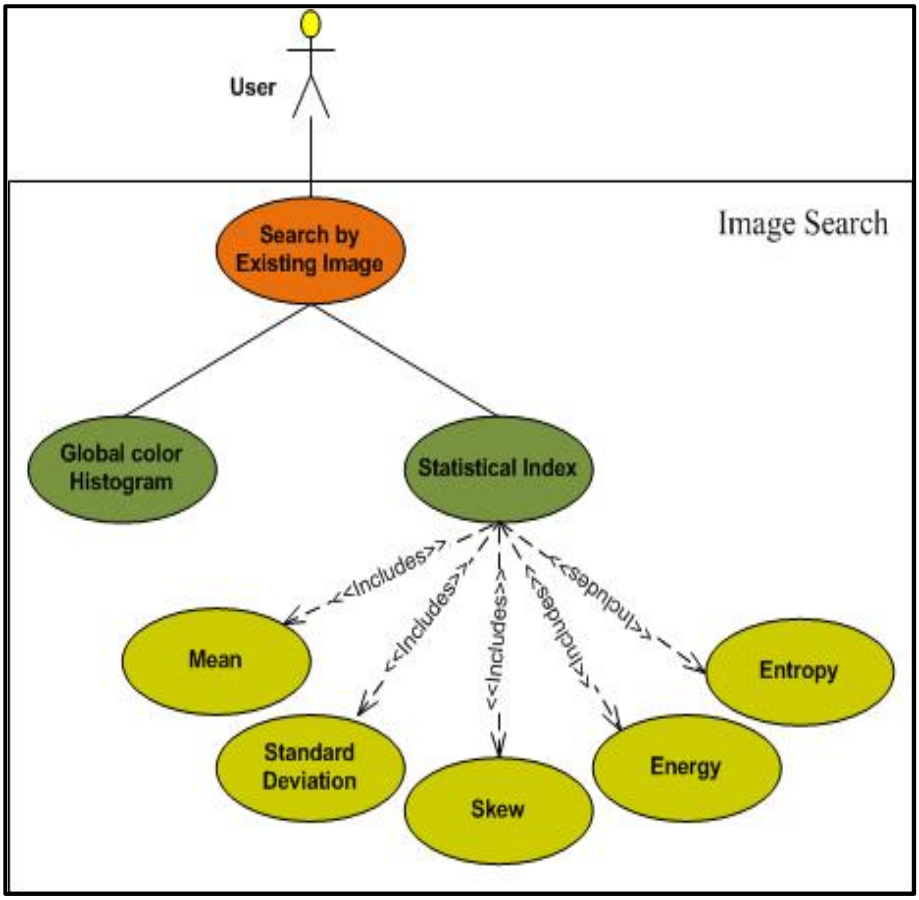
Figure: CBIR Help Page



10. UML DIAGRAMS

10.1 Use case Diagrams and Details

Use case Diagram 1: Image Search (User)



Detail Use Case UC1: Image Search (User)

Scope: Content Based Image Retrieval (CBIR) application

Level: user goal

Primary Actor: User

Triggering Event: User uploads image to be searched in the database and click on Search button.

Stakeholders and Interests:

1. User: wants to search database of existing images using fast and accurate retrieval tool. Wants user-friendly retrieval system, which can take image and returns all similar images in no time.
2. Administrator: Wants to keep the database updated with images and makes sure that CBIR works perfectly all the time. Wants to make sure that it is accessible all the time.

Preconditions:

CBIR is in working condition. User must have to upload image for searching.

Success Guarantee (or Post conditions):

Similar images matching with input image is returned and displayed in user friendly manner. Similarity score is calculated and reported on the result page.

Main Success Scenario (or Basic Flow):

1. User visits CBIR homepage.
2. User selects image search option given on the homepage to search similar images in database.
3. User uploads image using upload button.
4. CBIR system takes the input image given by user and calculates Global Color Histogram.
5. CBIR calculates Statistical Index like Mean, Standard Deviation, Skew, Energy and Entropy of input and database images.
6. CBIR selects top matching images having a score greater than threshold.
7. CBIR displays selected matching images as output in a user friendly tabular format.
8. User gets the output and analyzes the result.

Extension (or Alternative Flows):

a. At any time System fails:

User needs to restart the system again and perform the search.

b. Out of connection:

User must be connected with internet in order to use CBIR as it is online system.

1a. CBIR server is down and homepage is not working.

- User needs to refresh the system in order to repair minor network conflict.
- If it is still not working, user needs to report to the administrator.

2-3a. User fails to upload image.

- If image upload button is not working, user must report bug to the administrator
- If image is not in specified format, user must need to generate image in image specific format like jpeg, png, etc
- If image search option on homepage doesn't work, user must file bug to the system.
- If user selects invalid file, the system must report appropriate error and display it on screen.

4a. System fails to calculate Global Color Histogram.

- If system cannot calculate accurate GCH, the administrator must fix the bug.
- If system fails to download uploaded image, the administrator must look for the possible error like path error or format error or error in code and fix it.

5a. Error in Statistical Index Calculation

- If system fails to calculate mean, standard deviation, skew, energy and entropy of database images and input image, the administrator must look for the possible error in calculation and code in order to fix it.

6-7a. Error in selection of highly similar images

- If there is no similar images found due to inefficient threshold, system must report no matching found.
- If there is error in parameter calculation, administrator must fix the problem so that user gets all possible matching images.
- If there is problem in reporting similar images, administrator must look at the code and find error in printing output.

Exceptions:

2-3a. The system should upload appropriate image selected by user. If not, the system should display error message on screen.

4a. If system cannot calculate Global Color Histogram because of internal error, system should provide valid message. It may also happen that system calculates it wrong, appropriate message should be displayed.

5a. If system wrongly calculates statistical index values like mean, standard deviation, skew, energy and entropy, it should provide valid message to user.

6a. If system fails to detect similar image, or image database is not found, appropriate message should be displayed on screen.

Special Requirements:

1. Attractive and user friendly home page
2. Easy to analyze output format with matching image picture and parameter values
3. Order of matching – highest matching image first and less similar images late.
4. Acceptance of varieties of image format like jpeg, png, bitmap, etc.
5. Fast calculation and rendering output in within 30 second 75% of the time

Frequency of Occurrence: Could be nearly continuous

Use case Diagram 2: Image Search (DBA)

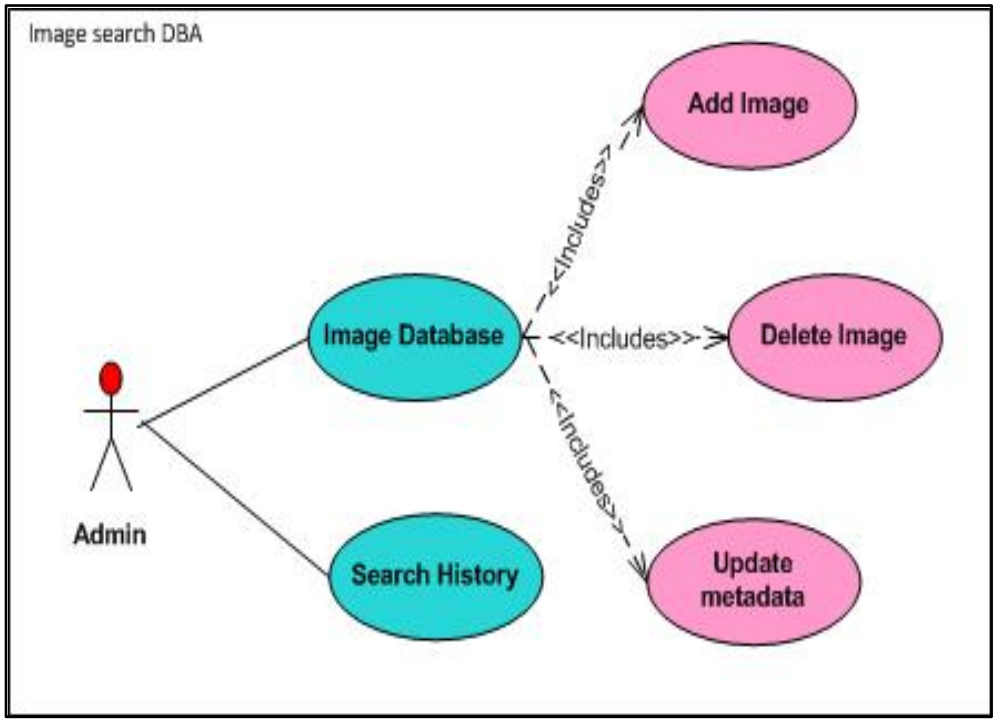


Diagram Version 1.1

Detail Use Case UC2: Image Search (DBA)

Scope: Content Based Image Retrieval (CBIR) application maintenance

Level: Admin goal

Primary Actor: Database Administrator (DBA)

Triggering Event: DBA performs login operation and look for updating/maintenance.

Stakeholders and Interests:

1. User: wants to search database of existing images using fast and accurate retrieval tool. Wants user-friendly retrieval system which can take image and returns all similar images in no time.
2. Administrator: Wants to keep the database updated with images and makes sure that CBIR works perfectly all the time. Wants to make sure that it is accessible all the time.

Preconditions:

CBIR is in working condition.

The administrator must logon to access admin privileges.

Success Guarantee (or Post conditions):

Upon giving valid administrator username and password by admin, the system should allow admin to access privileges.

Administrator performs additions, deletion and updating of image metadata.

Main Success Scenario (or Basic Flow):

1. Administrator visits CBIR homepage and logon using admin link provided.
2. Administrator looks for database search history.
3. Administrator updates images by adding images to database.
4. Administrator deletes images which are not of use.
5. Administrator updates metadata.
6. Administrator must need to logout after performing necessary operations.

Extension (or Alternative Flows):

a. At any time System fails:

Admin needs to restart the system again and perform updates again.

b. Out of connection:

Admin must be connected with internet in order to use CBIR as it is online system.

1a. CBIR server is down and homepage is not working.

- Admin needs to refresh the system in order to repair minor network conflict.
- If it is still not working, the administrators need to look for possible error and fix it.

1b. Admin fails to logon using admin link provided on website

- Admin must need to work on possible solution/alternative method for logon
- Admin needs to look for exact error caused to fail

1c. Admin gives wrong username/password.

- System must provide three chances for logon and should exit automatically if admin tries to logon fourth time.

2a. If search history link is not working, admin should take necessary steps to fix it

3a. If admin fails to add images, he should look for bugs causing error in adding images.

4a. If admin fails to delete image from specific path and image remains there, admin should look for fixing the bug.

5a. If admin fails to update metadata, admin should would need to find cause of error and fix it.

6a. If system fails to logout, admin should quickly find error and fix it.

Exceptions:

2a. If there is no search history, system should give appropriate message such as “No search history found”.

3a. If added images are not copied to specific folder, admin should double check path where images are being stored.

4a. If wrong image is being deleted by the system, admin should perform testing on each step and look to fix the error.

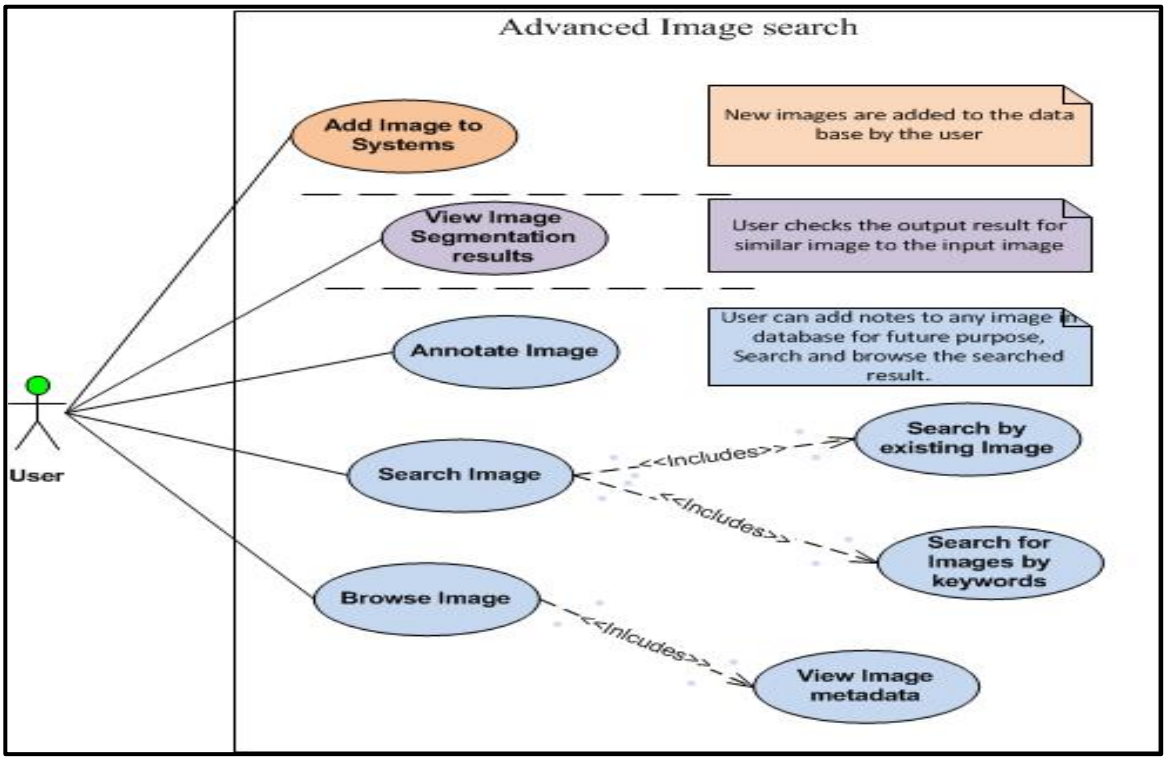
6a. If the system is hacked by using admin password, admin should keep track of records every time whenever he logon and perform necessary check.

Special Requirements:

1. Easy to use maintenance site
2. Log history for each and every admin operations performed.

Frequency of Occurrence: Could be nearly continuous

Use Case 3 Image Search (Advanced)



Detail Use Case UC3: Advanced Image Search (User)

Scope: Content Based Image Retrieval (CBIR) application

Level: user goal

Primary Actor: User

Triggering Event: User uploads image or image's keyword to be searched in the database and click on Search button.

Stakeholders and Interests:

1. User: wants to search database of existing images or its keywords using fast and accurate retrieval tool. Wants user-friendly retrieval system which can take image or its associated keywords to search and extract all similar images in no time.

2. Administrator: Wants to keep the database updated with images, its keywords and its metadata to make sure that CBIR works perfectly all the time and also to make sure that it is accessible all the time and is secured.

Preconditions:

CBIR is in working condition. User must have to upload image or its keyword for searching.

Success Guarantee (or Post conditions):

Similar images matching with input image or its keyword is returned and displayed in user friendly manner. Similarity score is calculated and reported on the result page.

Main Success Scenario (or Basic Flow):

1. User visits CBIR homepage.
2. User selects image search option or search by keyword option given on the homepage to search similar images using keywords in database.
3. User uploads image using upload button or enters a keyword for the image.
4. CBIR system takes the image as input from user to compute Global Color Histogram and also retrieves images based on the search keys that are annotated for each image by the admin.
5. CBIR performs complex calculations such as statistical indexing, compute mean, Standard Deviation, Skew, Energy and Entropy, This system maintains the repository of the keywords for the stored images and also its metadata.
6. CBIR selects top matching images having score greater than threshold.
7. CBIR displays selected matching images as output in a user friendly tabular format.
8. User gets the output and analyzes the result.

Extension (or Alternative Flows):

a. At any time System fails:

User needs to restart the system again and perform the search.

b. Out of connection:

User must be connected with internet in order to use CBIR as it is online system.

1a. CBIR server is down and homepage is not working.

- User needs to refresh the system in order to repair minor network conflict.
- If it is still not working, user needs to report to the administrator.

2-3a. User fails to upload image or its keyword.

- If image upload or the keyword button is not working, user must report bug to the administrator.
- If image is not in specified format, user must need to generate image in image specific format like jpeg, png, etc
- If image search option on homepage doesn't work, user must file bug to the system.
- If user selects invalid file, the system must report appropriate error and display it on screen.

4a. System fails to calculate Global Color Histogram.

- If system cannot calculate accurate GCH, the administrator must fix the bug.
- If the GCH fails, user can still retrieve the image based on search key, if the user search key matches the keyword in database, the corresponding image is displayed.
- If the keyword search fails to work, it should trigger an alert to the admin to address it.
- If system fails to download uploaded image, the administrator must look for the possible error like path error or format error or error in code and fix it.

5a. Error in Statistical Index Calculation or metadata computations.

- If system fails to calculate mean, standard deviation, skew, energy and entropy of database images and input image, the administrator must look for the possible error in calculation and code in order to fix it.
- If metadata is wrongly computed it should raise an alert to the admin to fix it.

6-7a. Error in selection of highly similar images

- If there is no similar images found due to inefficient threshold, system must report no matching found.
- If there is error in parameter calculation, administrator must fix the problem so that user gets all possible matching images.
- If there is problem in reporting similar images, administrator must look at the code and find error in printing output.

Exceptions:

2-3a. The system should upload appropriate image selected by user. If not, the system should display error message on screen.

4a. If system cannot calculate Global Color Histogram because of internal error , system should provide valid message. It may also happen that system calculates it wrong, appropriate message should be displayed.

4b. If the GCH fails, user can still retrieve the image based on search key, if the user search key matches the keyword in database, the corresponding image is displayed.

if the keyword search fails to work, it should trigger an alert to the admin to address it.

5a. If system wrongly calculates statistical index values like mean, standard deviation, skew, energy entropy and metadata, it should provide valid message to user.

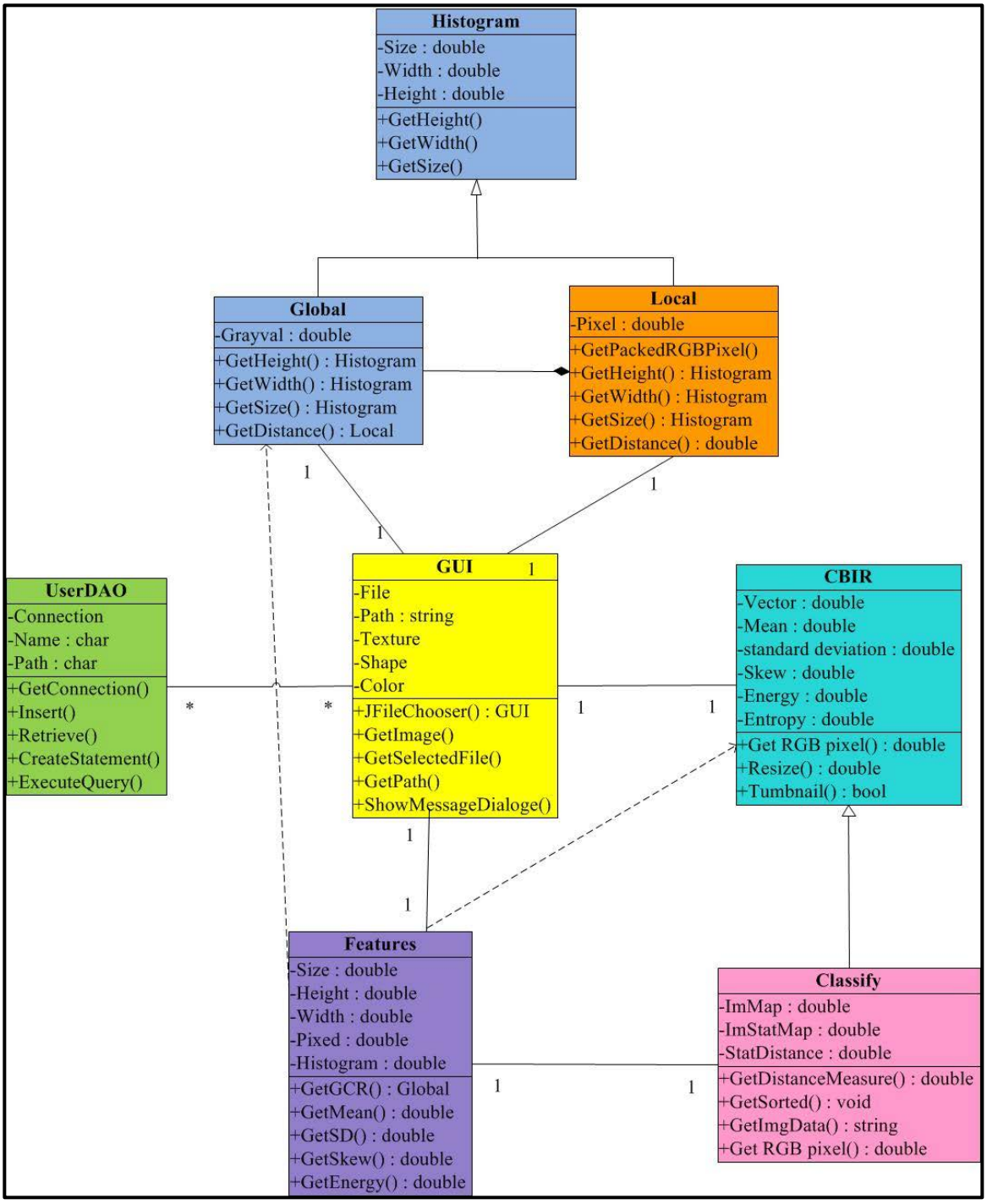
6a. If system fails to detect similar image, or image database is not found, appropriate message should be displayed on screen.

Special Requirements:

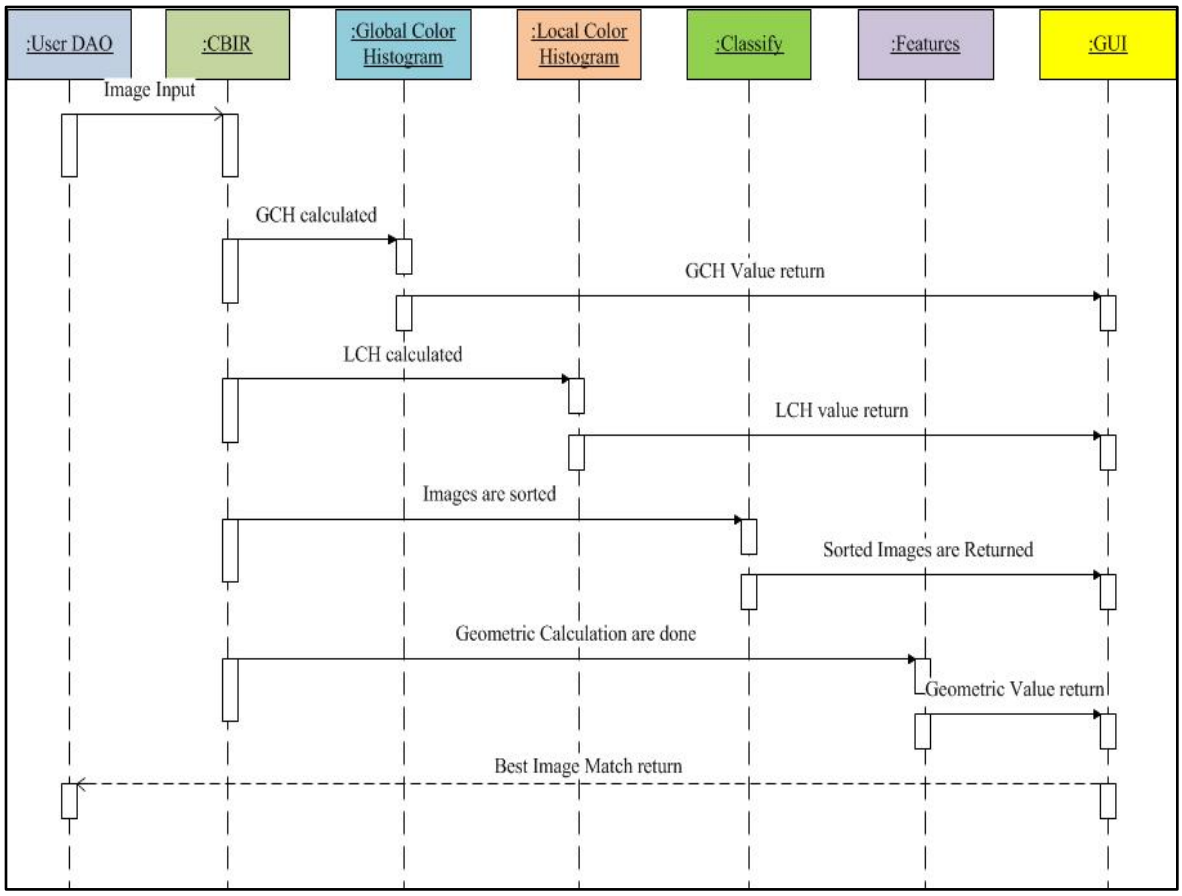
1. Attractive and user friendly home page
2. Easy to analyze output format with matching image picture and parameter values
3. Order of matching – highest matching image first and less similar images later
4. Acceptance of varieties of image format like jpeg, png, bitmap, etc
5. Fast calculation and rendering output in within 30 second 75% of the time

Frequency of Occurrence: Could be nearly continuous

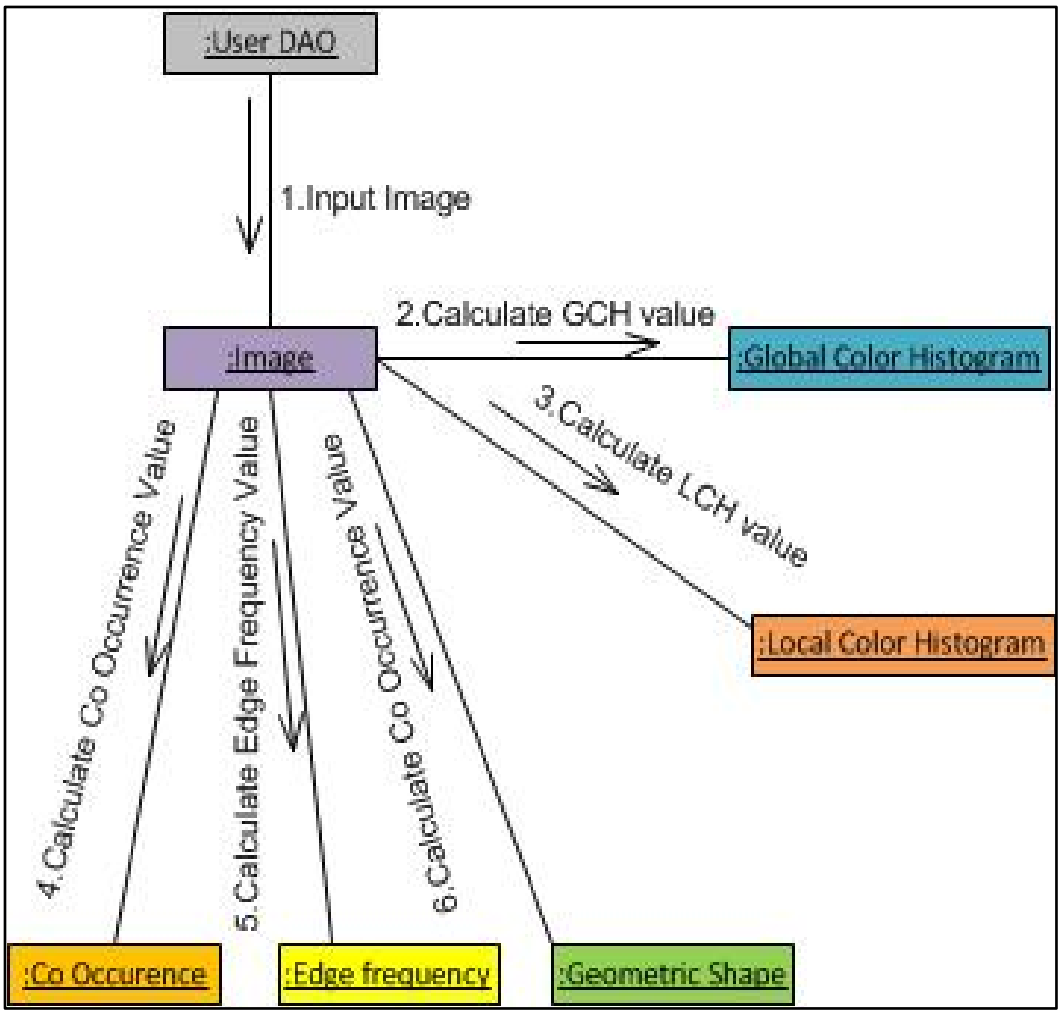
10.2 CLASS DIAGRAM



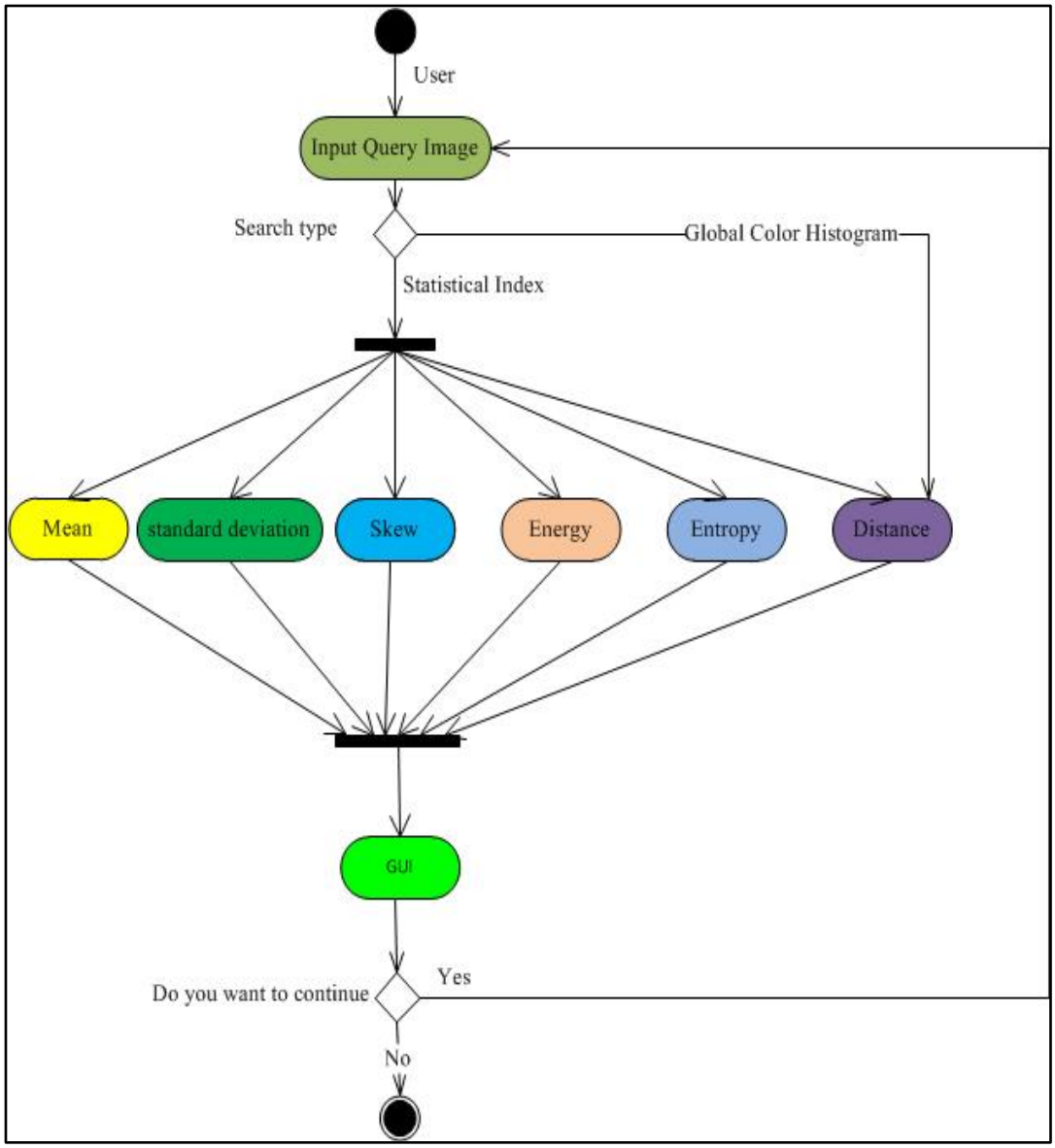
10.3 SEQUENCE DIAGRAM



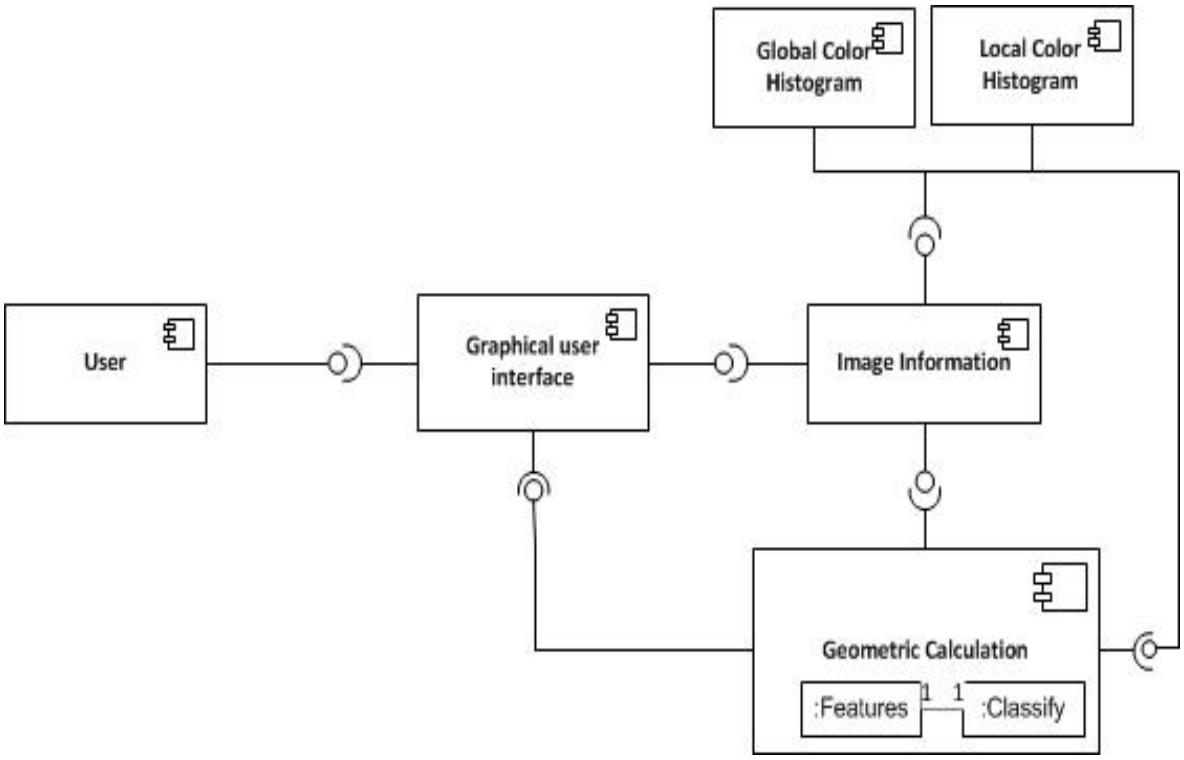
10.4 COLLABORATION DIAGRAM



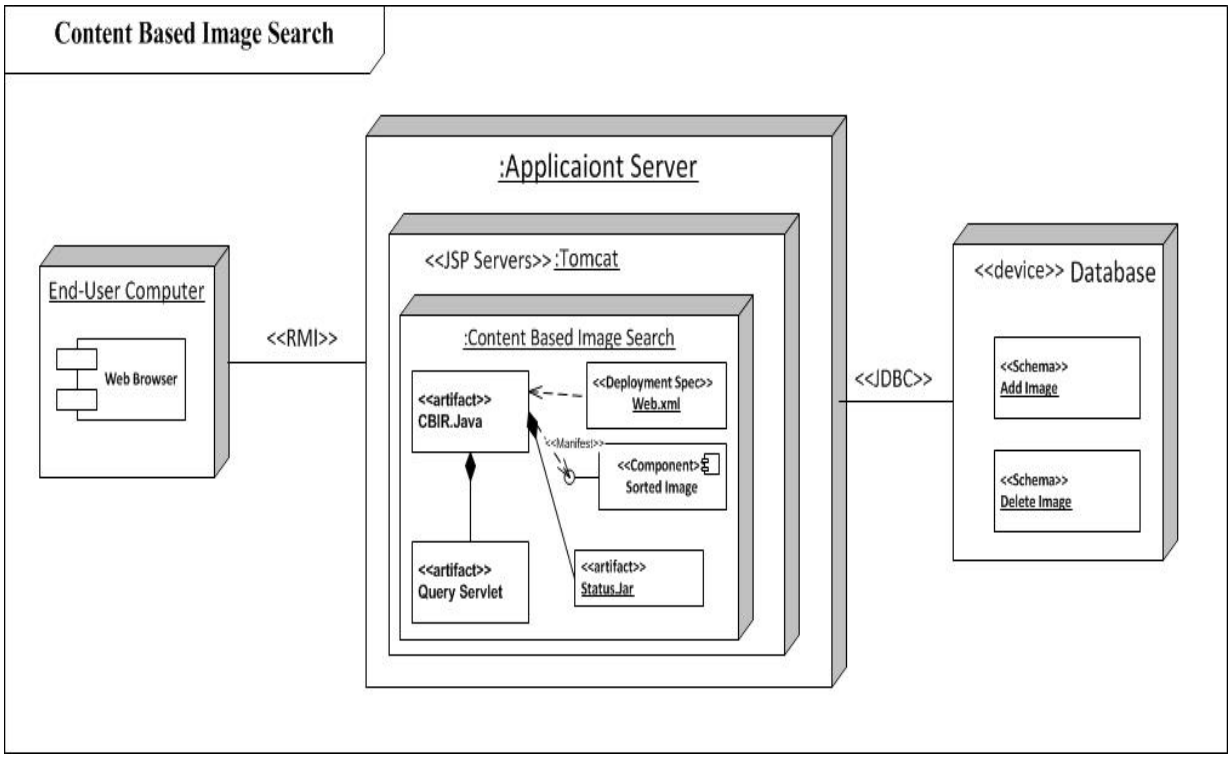
10.5 ACTIVITY DIAGRAM



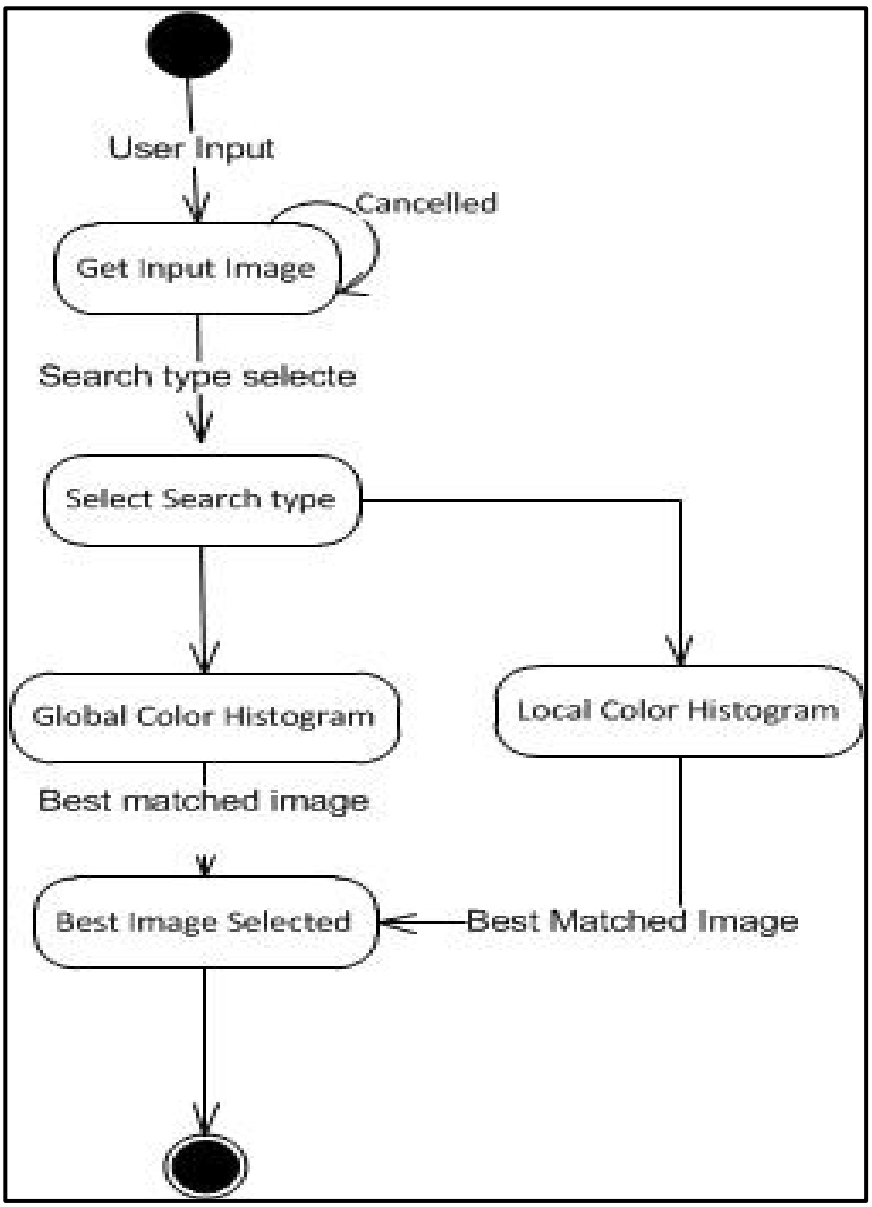
10.6 COMPONENT DIAGRAM



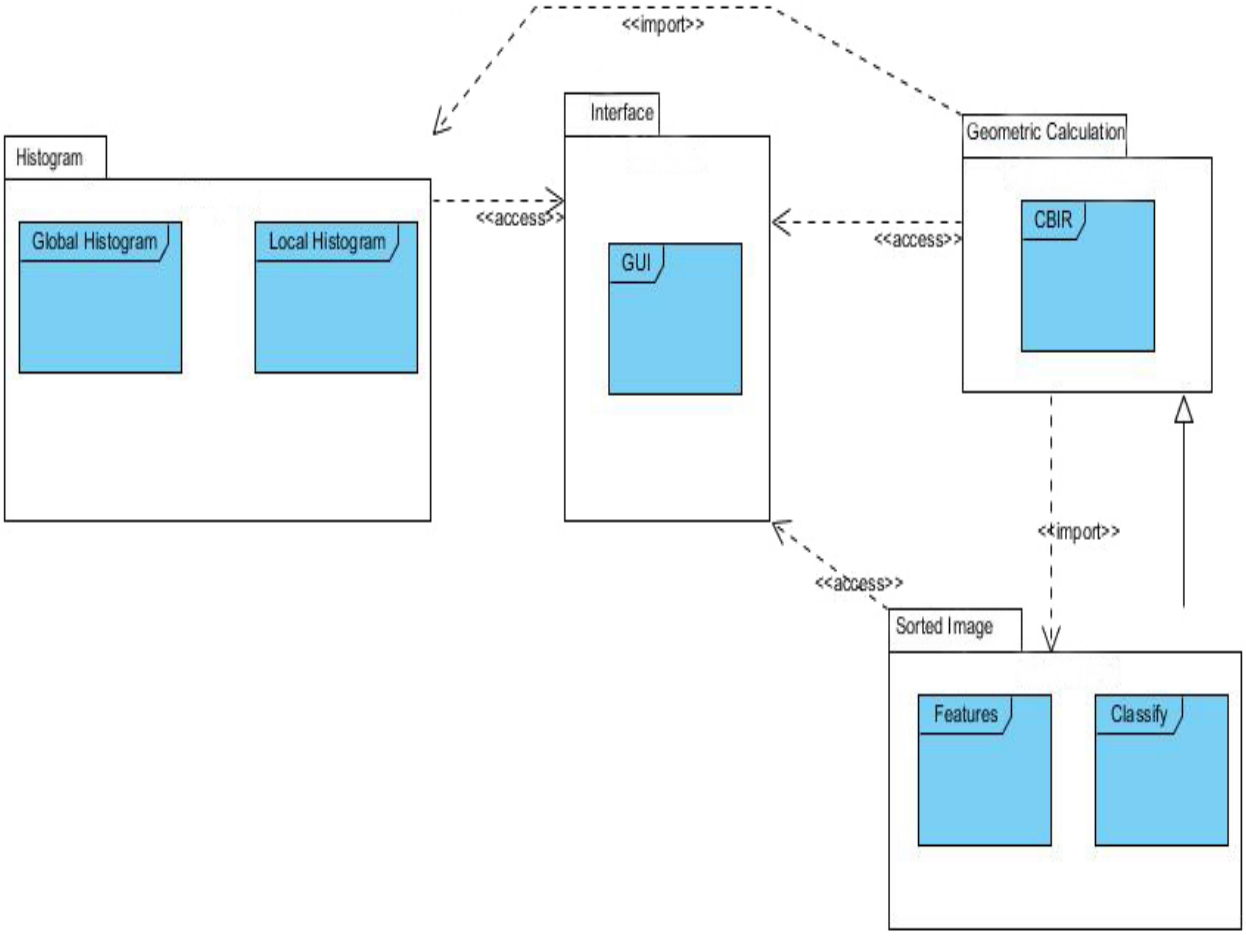
10.7 DEPLOYMENT DIAGRAM



10.8 STATE CHART



10.9 PACKAGE DIAGRAM



11. Training Plan

Scope of the Training Plan

1. The training plan will explicitly describe the CBIS software organization.
2. This will describe entire company's each division and organization.
3. The users, software engineers, developers, project manager, quality assurance department's each personnel will be covered under the training program.
4. Distinctive circumstances like including personnel outside the scope of training of training program should be explicitly stated.
5. Our plan will include CBIS organization's training analysis activities and funding activities.
6. CBIS software and hardware interfaces, GUI performances would be addressed by training program.

Responsibility for the Plan

1. CBIS's managerial owner is ultimately responsible for effective training program.
2. CBIS's organizational unit is responsible for creating and maintaining the training plan document. Organizational training plan is revised and develop according to documented procedure. Plan must clearly designate the responsibility for creating, updating, reviewing, approving, managing and controlling it.
3. The stakeholders will be responsible for providing suggestion and support the implementation.

Training Objectives

1. Goals and benefits to be achieved
2. It will include performing training needs, analysis and communicating the results across the CBIS organization, designing curricula and gaining consensus of priorities the curricula represents.
3. Training objective will address communication channels amongst each division, culture of organization, training policy and organizational goals the training program supports.
4. Training policy and organizational goals will imply constraints as well as success measurement of training program.

5. CBIS training policy will require personnel to receive 40hrs of minimum training per year.
6. Cost of training will also be taken into consideration.

Technical Strengths and Weaknesses of the Software Organization

1. Software risk and assessments.
2. Training based on CMM (CMM and level 3 KPAs).
3. Standards – CMM, ISO 9000, various CBIS's companies standard
4. Formal specification and highlight training needs.

Software Engineering Curriculum

1. Content description for the courses.
2. Intended audience for the courses.
3. Funding sources for the course.
4. Length of course, delivery medium, course development and retirement schedule.

Table 1

Courses	Coverage	Duration	Curriculum architecture
Design & code inspection	Senior Engineers	16	Technical / Professional skills Document templates
Software testing	Quality assurance Engineers	30	
Software project management	Development management	25	Interpersonal / Managerial skills. Resource / Reference. Related group roles / internal companies' policies.
Human resource management		7	Software team management
Software engineers and practitioners		10	Software estimations and inspection training.

Table 2:

TRAINING COURSE	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
Software Related Personnel:																		
CBIS Executive Management	m							r	r							r	r	
CBIS Middle Management	m							r	r							r	r	
CBIS Software Management:																		
CBIS Project Leader	m	r	r	r	m			m	m	r				r		m	m	
CBIS Chief (1st Line SW Manager)	m	r	m	r	r	m		m	m					r		m	m	
SEPG	m	m	m	r	r		r	m	m	r		r			m	m	m	
SQA	m	r	r	m			m	m	m	m						m	m	
SCM (Company Level)	m	r	r	r			r	m	m	r						m	m	
Software Planner/Estimator	m	r	r		r							m						
Training Facilitator	m										m							
CBIS Software Development:																		
Software Product Manager	m	m	m	m	r		r	m	m	m		r	m	m		m	m	

Software Product Acquisition Manager	m	r	r		m						r	m	m				m
Review Leaders	m	r	r	m			m	r	r	m			r			m	m	
Review Participants	m	r		m			m	r	r	r			r			m		
SCM (Project Level)	m							r	m	m			m			m		
System Engineer	m	r						m	m	m			r			m	m	
Software Engineer:																		
Design	m	r		r				m	m	m			m	r		m	m	
Code	m	r		r				m	m	m			m	r		m	m	
Test	m	r		r				m	m	m			m	r		m	m	
System Test Engineer	m	r		r				m	m	m			r	r		m	m	

A: Overview of the Software Development Process
B: Process Fundamentals
Reports C: Managing Software Development
D: Software Product Evaluations
E: Project/Program Management
F: Chief Training
G: Inspections
H: Software Quality Assurance (SQA)
I: Software Configuration Management (SCM)
R: Subcontractor Management
J: Software Corrective Action System/ Software Problem/Anomaly
K: Train the Trainer
L: Software Estimating
M: Software Requirements Management
N: Risk Management
O: Software Engineering Process Group
P: Working as a Team
Q: Software Engineering Disciplines

Table 3: CBIS Software Engineering Core Curriculum

CBIS course Curriculum
Fundamental
Developing Quality Software
Personally Shaping the Software Solution
Languages
Java
Java Script
Server
Tom castle
SEI Capability Maturity Model and Quality System Review
Introduction to the CMM
CMM Advanced Concepts and KPA level 3 and level 4
Assessor Training for CBIS Software and Quality as well as its System Review
Systems Engineering
CBIS system Engineering Process
CBIS Systems Requirements and specifications Engineering
Process and Technology
Structured Methods for Database Applications
Software Reviews
Software Technology Planning for Practitioners
Implementing Continuous Improvement
Implementing Software Configuration Management
Implementing Software Sizing and Estimation

CBIS course curriculum
Process and Technology (continued)
Gathering Requirements
UML diagrams
Object-Oriented Analysis Using Object-Modeling Technique
Object-Oriented Design Using Object-Modeling Technique
Implementing Software Continuous Improvement
Software Metrics for Process Improvement
Software Reviews for Information Systems
Software Unit and Integration Testing
Software System Testing
Management
Managing the Software Development Process
Software Management Seminar
Understanding Software Sizing , continuous improvement and Estimation for Managers
Understanding Software metrics
Executive Workshop on Software Competency

Estimated Training Costs

There are three types of training costs that need to be considered when creating a training plan.

1. Direct costs. These are all of the costs directly associated with the development, acquisition, and delivery of the training like student and consumable materials, cost of facilities, faculty and staff, administrative personnel.
2. Indirect costs. These are the costs incurred by having the trainees inactive on their projects during the training.

3. Negative costs. These are the costs that would be incurred if necessary training were not delivered. Negative costs are also a form of indirect costs but need to be categorized separately.

12. Quality and Test Plan

The objective of our test plan is to find and report as many bugs as possible to improve the integrity of our program. Although exhaustive testing is not possible, we will exercise a broad range of tests to achieve our goal. We introduce a test collection for the evaluation of CBIR algorithms. In the test collection, the performance testing is based on photograph similarity perceived by end-users in the context of realistic illustration tasks and environment. The building process and the characteristics of the resulting test collection are outlined, including a typology of similarity criteria expressed by the subjects judging the similarity of photographs. A small-scale study on the consistency of similarity assessments is presented. A case evaluation of two CBIR algorithms is reported.

A test plan documents the strategy that will be used to verify and ensure that a product or system meets its design specifications and other requirements. A test plan is usually prepared by or with significant input from test engineers.

Depending on the product and the responsibility of the organization to which the test plan applies, a test plan may include a strategy for one or more of the following:

- *Design Verification or Compliance test* - to be performed during the development or approval stages of the product, typically on a small sample of units.
- *Manufacturing or Production test* - to be performed during preparation or assembly of the product in an ongoing manner for purposes of performance verification and quality control.
- *Acceptance or Commissioning test* - to be performed at the time of delivery or installation of the product.
- *Service and Repair test* - to be performed as required over the service life of the product.
- *Regression test* - to be performed on an existing operational product, to verify that existing functionality didn't get broken when other aspects of the environment are changed (e.g., upgrading the platform on which an existing application runs).

A complex system may have a high level test plan to address the overall requirements and supporting test plans to address the design details of subsystems and components.

Test plan document formats can be as varied as the products and organizations to which they apply. There are three major elements that should be described in the test plan: Test Coverage, Test Methods, and Test Responsibilities. These are also used in a formal test strategy.

Test coverage

Test coverage in the test plan states what requirements will be verified during what stages of the product life. Test Coverage is derived from design specifications and other requirements, such as safety standards or regulatory codes, where each requirement or specification of the design ideally will have one or more corresponding means of verification. Test coverage for different product life stages may overlap, but will not necessarily be exactly the same for all stages. For example, some requirements may be verified during Design Verification test, but not repeated during Acceptance test. Test coverage also feeds back into the design process, since the product may have to be designed to allow test access (see Design for test).

Test methods

Test methods in the test plan state how test coverage will be implemented. Test methods may be determined by standards, regulatory agencies, or contractual agreement, or may have to be created new. Test methods also specify test equipment to be used in the performance of the tests and establish pass/fail criteria. Test methods used to verify hardware design requirements can range from very simple steps, such as visual inspection, to elaborate test procedures that are documented separately.

Test responsibilities

Test responsibilities include what organizations will perform the test methods and at each stage of the product life. This allows test organizations to plan, acquire or develop test equipment and other resources necessary to implement the test methods for which they are responsible. Test responsibilities also includes, what data will be collected, and how that data will be stored and reported (often referred to as "deliverables"). One outcome of a successful test plan should be a record or report of the verification of all design specifications and requirements as agreed upon by all parties.

Color histogram is an efficient feature, which has been widely used by CBMIR systems. Among the color spaces, HSV is commonly considered for histogram due to its close relation with the human visual perception. YUV is also commonly used due to its practical benefits. Additionally, most JPEG codec's work in the YUV color space. These facts have led to employing HSV and YUV color histogram features for color-based image retrieval experiments.

Table 1 presents the feature extraction parameters.

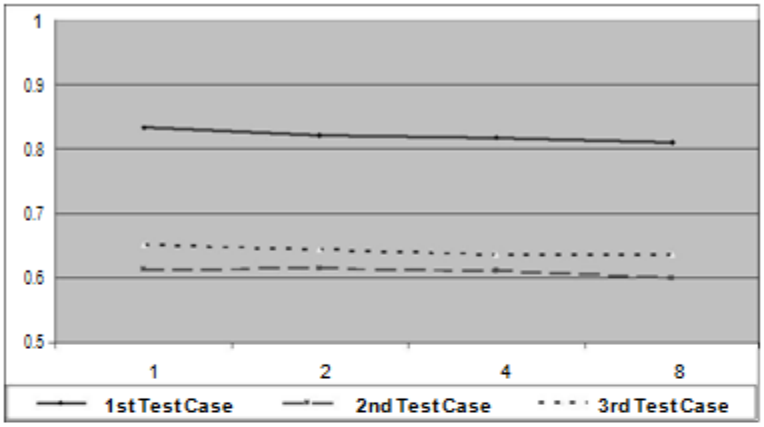


Figure 1: Overall color-based retrieval performance results

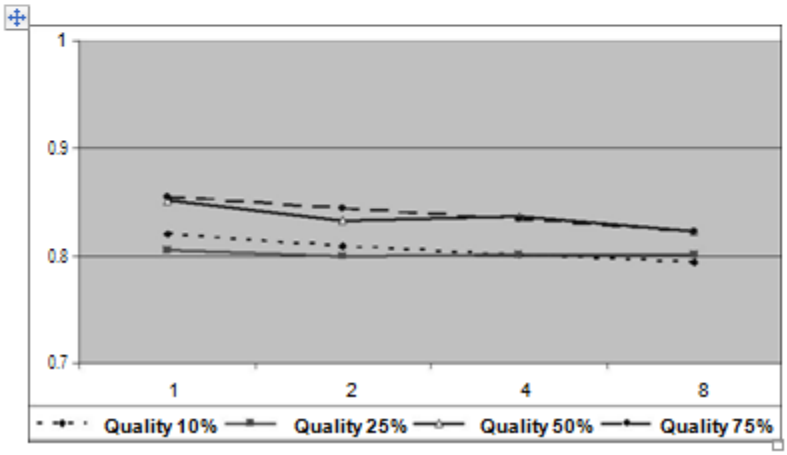


Figure 2: Color -based retrieval performance results for the first test case

In both figures, the horizontal axis represents the scaling factor while the vertical axis represents the normalized OQP values. The OQP values in the figures are between 80% and 90%, and they are stable in the horizontal direction for certain compression ratio. This means that scaling does not affect the performance significantly. The reason of this performance robustness to scaling can be explained with insignificant color information loss on the image pixels after DCT-based

downscaling. It can also be seen from Figure 2 that OQP values are slightly lower for higher compression ratio at certain scaling factors.

These differences for each scaling factor are also stable. The effects of compression are also studied in. In the second test case 16 query sets are queried on uncompressed and unscaled database to simulate the second use case. In general, the performance in this test case is lower than the previous case. This degradation in performance is caused by using different compression ratio for query image and image database. The quantization step in the loss compression may shift the color values for each pixel in the image; consequently the comparison based on color information yields worse performance. Even if the same image is used in the query and image database, retrieval performance does not need to be high. However, the horizontal stability of the OQP values in the figure shows that performance is still robust to scaling. Finally in the third case, the 16 query sets are queried on 75% compressed and un scaled database. Figure 1 plots the concerning average OQP values calculated for the queries. Third case reveals similar results to the second case. Different compression ratios in query image and image database cause low performance, although they are slightly better than in the second case. Scaling does not affect the performance significantly as in the previous test cases.

13. CAPABILITY MATURITY MODEL – CMM AND KEY PROCESS AREA (KPA – LEVEL 3, 4)

Building real-world systems for content based image search, involves regular user feedback during the development process, as required in any other software development life cycle. Over motivation to organize things is inherent. Thus training is one of the most important aspects of improving a software organization. Both Capability Maturity Model (CMM) and the People CMM (PCMM), developed by the Software Engineering Institute (SEI), contain key process areas (KPAs) addressing training. In both models, these training efforts become focused upon the entire organization at maturity level 3. Without an adequate training plan, funds may be wasted or improperly spent. There are many things to be considered when creating a training plan. The training process for Content based image search's software organization follows ETVX paradigm.

CMM Level	Course	A	B	C	D	E	F	G	H
2	Introduction to Software Process Improvement	X	X	X	X	X	X	X	X
2	Project Management and Tracking	X	X	X			X	X	X
2	Requirements Management	X		X	X	X	X	X	X
2	Software Quality Assurance	X		X	X		X	X	X
2	Software Configuration Management	X		X	X	X	X	X	X
2	Sizing and Estimating of Software and Computing Resources	X	X	X	X			X	X
2	Project-Specific Orientation	X	X	X	X	X	X	X	X
2	Quality Improvement Awareness	X	X	X	X	X	X	X	X
2	Managing Quality Improvement	X		X				X	X
2	Team Leader	X		X				X	X
2	Process Management	X	X	X	X	X	X	X	X
3	Engineering of Software	X		X	X	X	X	X	X
3	Peer Review	X		X	X		X	X	X
3	Introduction to Software Process Assessments	X		X				X	X
3	Software Engineering Process Group							X	X

3	Instructional Techniques								X
3	Principles of Team Building	X		X				X	X
4	Quantitative Process Management	X	X	X	X	X	X	X	X
4	Software Quality Management	X	X	X	X	X	X	X	X
5	Defect Prevention	X		X	X		X	X	X
5	Technology Change Management	X						X	X

A: Project Manager

B: Project Management Specialist

C: Software Team Leader

D: Software Practitioner

E: CM Specialist

F: QA Specialist

G: SEPG Member

H: Training Group Manager

E (Entry criteria) – Entry criteria are those condition that must be present prior to beginning the training process.

T (Task) – The tasks are the essential activities of the process. The designed guidelines focuses on the creating a training plan.

V (Validation tasks or criteria) – The validation training process ensures that the output of the tasks meet required standards.

X (Exit criteria) – Exit criteria revises and checks the output .

ENTRY CRITERIA	TASKS	VALIDATION CRITERIA	EXIT CRITERIA
Management support	Conduct training and perform training analysis	Training plan approvals	Needed training delivered
Training policy and objectives	Create training plan	Course development and delivery processes	Training objectives met.
Resources	Design curriculum	Quality standards met	
Organizational context	Pilot and deliver training	Training result analysis and generating report	
	Evaluate training		

13.1 KPA – KEY PROCESS AREA (LEVEL 3 - DEFINED)

The key process area for CBIS's capability maturity model, become focused upon entire organization at the maturity level 3. Our book, "Software engineering – A practitioner approach book says that KPA is used at each level of maturity. Then why is the training program KPA at level 3 in CBIS? The key is that at level 3 process improvement efforts are focused, defined and integrated. At level 3 improvement efforts are coordinated and focused at the organizational level, and are no longer a loose collection of bottom-up improvement efforts. Thus KPA focuses of

- Training program
- Focus on CBIS organizational software process.
- Focus on CBIS organizational process definition.
- CBIS integrated management.
- CBIS software product engineering.
- CBIS intergroup coordination and peer reviews.
- Organizational focus on training at level 3.

CBIS project driven training

The CBIS training program will be project-driven, thus training needs will emerge in a bottom-up fashion, focusing and identifying all training needs. Project-driven training is more efficient when the needs are common amongst the different team of same project.

Training plan based on KPA at level 3

Training plan would be focused on educating and training community, software engineer's team. Through this collaborative and joint venture training we are planning to develop the experience, well trained and expert engineers.

- Provide visibility to major skill/job shifts and emerging markets.
- Leveraging educational assets – human, technical, physical and sharing resources.
- *Improving learning efficiency, eliminating redundancy, and reducingtraining costs.*
- *Reducing the cycle time of training needs assessment and course development,*
- *Transferring knowledge through common training platforms.*

Training plan KPA common feature

A) Commitment to perform.

- CBIS software organization follows a written policy for meeting its training needs.

B) Ability to perform.

- Adequate resource, funding, fulfillment of training needs, skills and knowledge of training.

C) 4 inputs to training plan.

1 ***Management policy and support*** – CMM training will require on written policy which will meet CBIS training needs.

2 ***Resources.***

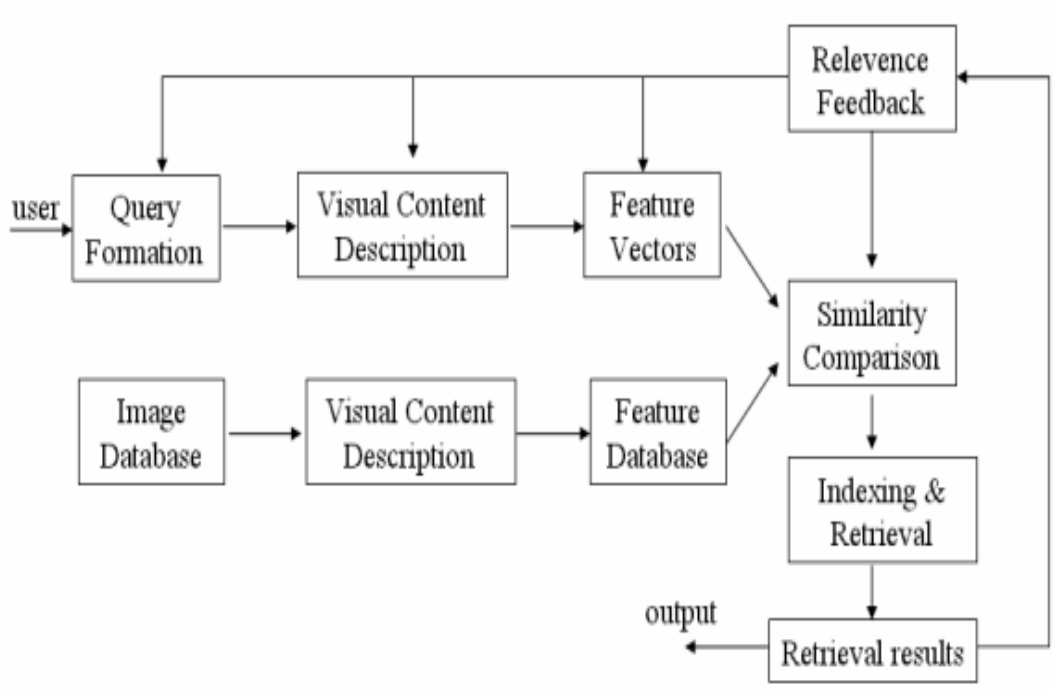
3 ***Project training plans.***

4 ***A result of an organizational training needs analysis.***

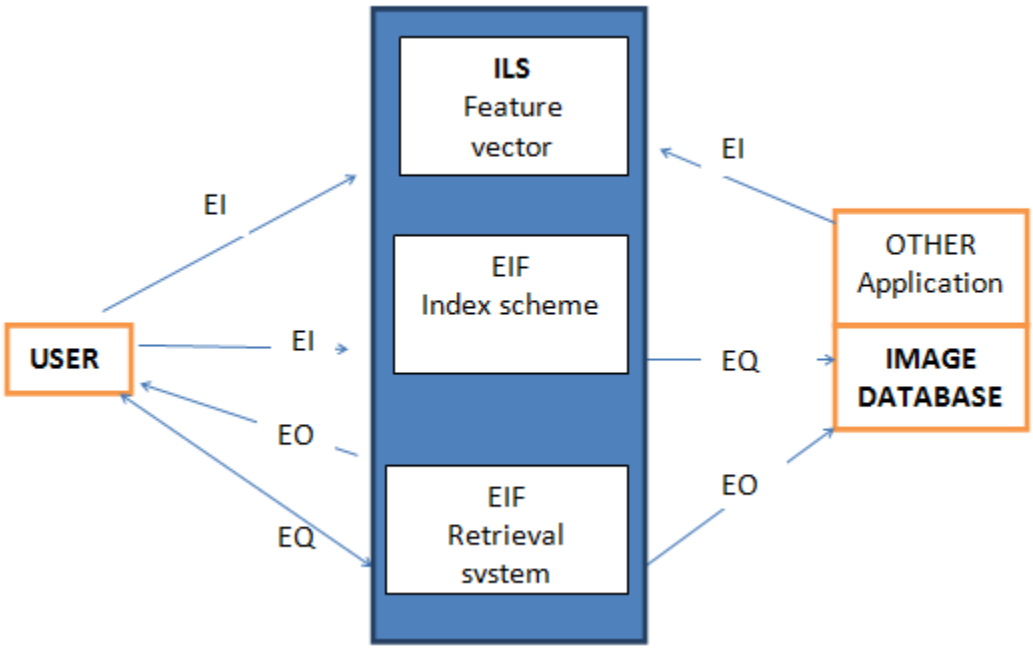
1. CBIS documented training policy	Notebook to record active written policies.
Identifying CBIS training requirement • CBIS team leader • Software project manager • Quality assurance specialist • CBIS project manager	CBIS software organizations are made aware of training policy/plan/ explain their role in training program. KPA at level 4 indicates that software manager throughout the organization are made aware of training policy and plans.
Requisite skill and knowledge	Provide needed training in acquiring skills and knowledge to individuals.
Planning training • Job related training • Individual-project specific needs.	
2. Resources training • Facilities	KPA level 1, 2, 3 refers to resources. Strategic and tactical planning by involving expertise in

• Funding • Staffing	organizational behavior. CBIS budgeting. Written and oral communication particularly with staff member.
3. Project training plans • Setting up needed skills • Project planning • Identifying training needs	
4. Results of organizational training and analysis. • Collect the data. • Analyzing the data	

13.2 Block Diagram



14. Functional point Analysis.



1. Determine Unadjusted Function Point

Measurement Parameter	Count	Simple	Average	Complex	
External Inputs	3 x	● 3	● 4	● 6	= 12
External Outputs	2 x	● 4	● 5	● 7	= 10
External Inquiries	3 x	● 3	● 4	● 6	= 18
Internal Logical Files	1 x	● 7	● 10	● 15	= 15
External Interface Files	1 x	● 5	● 7	● 10	= 10

Count = Total ---------- 65

2. Determine Value Adjustment Factor

Rate Each Factor: **(0 - No Influence, 1 - Incidental, 2 - Moderate, 3 - Average, 4 - Significant, 5 - Essential)**

1. How many data communication facilities are there? ---------- 5
2. How are distributed data and processing functions handled? ---------- 4
3. Was response time or throughput required by the user? ---------- 4
4. How heavily used is the current hardware platform? ---------- 5
5. How frequently are transactions executed? ---------- 5
6. What percentage of the information is entered online? ---------- 5
7. Was the application designed for end-user efficiency? ---------- 5
8. How many internal logical files are updated by on-line transaction? ---------- 5
9. Does the application have extensive logical or math processing? ---------- 5
10. Was the application developed to meet one or many user needs? ---------- 5
11. How difficult is conversion and installation? ---------- 3
12. How effective/automated are startup, backup, and recovery?---------- 3
13. Was the application designed for multiple sites/organizations?---------- 2
14. Was the application designed to facilitate change?----------4

Value Adjustment Factor---------- **60**

3. Determine Function Points (Calculation)

Unadjusted Function Points x (0.65 + 0.01 x Value Adjustment Factor) ——————→81 FP

65 * (0.65 + 0.01 * 60)

= 81 FUNCTION

15. Performance Metrics (Precedence Network)

The **precedence diagram method** is a tool for scheduling activities in a project plan. It is a method of constructing a project schedule network diagram that uses boxes, referred to as nodes, to represent activities and connects them with arrows that show the dependencies.

Rules for developing activity networks

1. Some determination of activity precedence ordering must be done prior to creating the network.
2. Network diagrams usually flow from left to right.
3. An activity cannot begin until all preceding connected activities have been completed.
4. Arrows on networks indicate precedence and logical flow. Arrows can cross over each other, although it is helpful for clarity's sake to limit this effect when possible.
5. Each activity should have a unique identifier associated with it (number, letter, code, etc.). Looping, or recycling through activities, is not permitted.
6. Although not required, it is common to start a project from a single beginning node. A single node point also is typically used as a project end indicator.

Rules when using the forward pass

1. Add all activity times along each path as we move through the network (ES + Dur = EF).
2. Carry the EF time to the activity nodes immediately succeeding the recently completed node. That EF becomes the ES of the next node, unless the succeeding node is a merge point.
3. At a merge point, the largest preceding EF becomes the ES for that node.

Rules for using the backward pass

1. Subtract activity times along each path as you move through the network (LF – Dur = LS).
2. Carry back the LS time to the activity nodes immediately preceding the successor node. Those LS becomes the LF of the next node, unless the preceding node is a burst point.

3. In the case of a burst point, the smallest succeeding LS become the LF for that node.

Steps to reduce the critical path

1. Eliminate tasks on the critical path.
2. Re plan serial paths to be parallel.
3. Overlap sequential tasks.
4. Shorten the duration on critical path activities.
5. Shorten early tasks.
6. Shorten longest tasks.
7. Shorten easiest tasks.
8. Shorten tasks that cost the least to speed up.

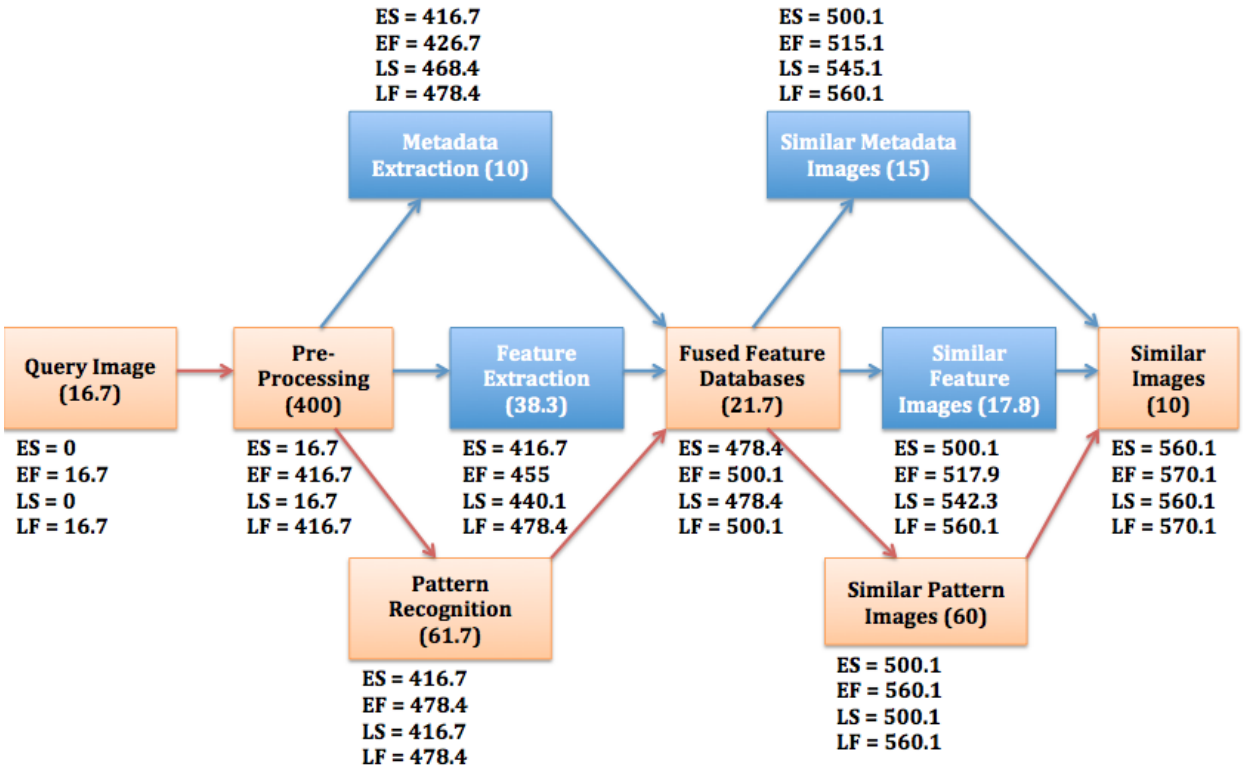
Activity	Description	Optimistic Time	Most Likely Time	Pessimistic Time	Estimated Time
A	Query Image	10	15	30	16.6
B	Pre-processing	200	400	600	400
C	Feature Extraction	20	40	50	38.33
D	Metadata Extraction	5	10	15	10
E	Pattern Recognition	40	60	90	61.66
F	Fused Feature Database	10	20	40	21.66
G	Similar Feature Images	10	18	25	17.83

H	Similar Metadata Images	10	15	20	15
I	Similar Pattern Images	30	60	90	60
J	Similar Images	10	10	10	10

All times are in milli seconds.

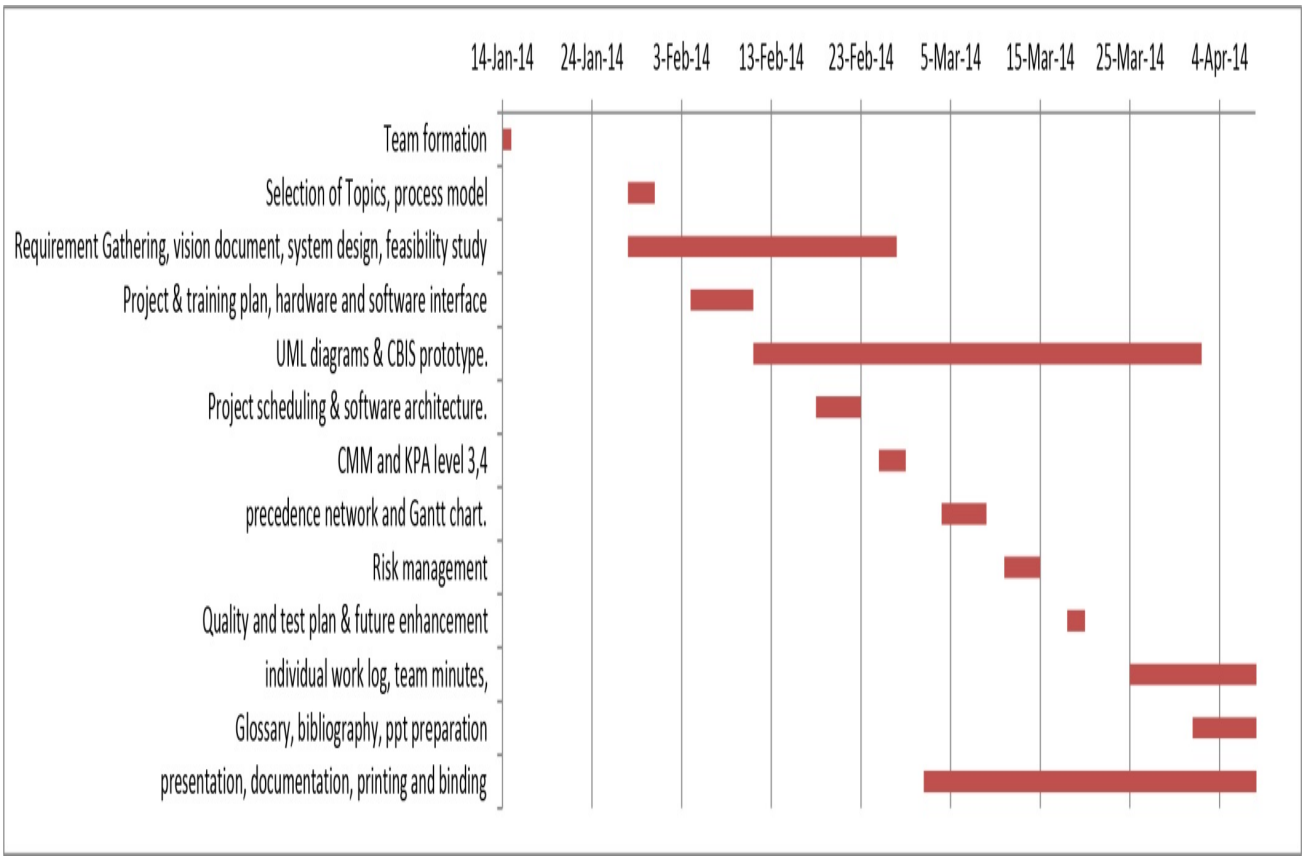
Estimated time is calculated using below formula.

Exp time = (Optimistic + 4 (Most likely) + pessimistic) / 6.



Precedence Diagram for Estimated time to answer one query.

16. Project Schedule (Gantt Chart)



17. Meeting Minutes.

Following are the basic weekly milestones for the progress as discussed during project startup:

Week 1 (January 14): Class start, formation of project team.

Week 2 (January 21): Brainstorming which topic to do for the project. Finalize project topic "Content Based Image Search". Electing team leader for our project.

Week 3 (January 28): Allotted different task to each team members.

Week 4 (February 4): Worked on UML diagrams and project documentation. Project milestone check.

Week 5 (February 11): Allotted topics for documentation and UML diagrams work to team mates following project capstones. Project milestone check.

Week 6 (February 18): 1st Midterm. Continued working on project documentation. Project milestone check.

Week 7 (February 5): Continued working on project documentation. Showed our UML diagrams to professor for proof read.

Week 8 (March 4): Project Milestone Check.

Week 9 (March 11): 2nd Midterm. Dissemination of tasks and project milestone check

Week 10 (March 19): Discussion UML diagram, work allocation and resolving team members query on issues they faced while documentation.

Week 11 (March 25): Project milestone check. Dissemination of task.Aggregation of team member's work by team lead.

Week 12 (April 1): Finalizing project document and UML diagrams.

Week 13 (April 8): Proof reading project document. Work on class presentation PPT.

Week 14 (April 15): Project Presentation. Error corrections. Printing and binding the project document.

Week 15 (April 22): Final exam

18. Revision History

Topic	Version	Reason for changes	Date	Name
Project Concept	1.0	Initial revision	01/21/2014	Karthik
	1.1	Review	01/22/2014	Slesha
	2.0	Final version	01/23/2014	Karthik
Requirements	1.0	Initial revision	01/21/2014	Slesha
	1.1	Review	01/31/2014	Dharti
	2.0	Final version	02/18/2014	Slesha
Vision Document	1.0	Initial revision	02/25/2014	Sneha
	1.1	Review	02/27/2014	Dharti
	2.0	Final version	03/04/2014	Sneha
Feasibility Report	1.0	Initial version	01/28/2014	Trupti
	1.1	Review	01/31/2014	Akanksha
	2.0	Final version	02/04/2014	Trupti
Project Plan	1.0	Initial revision	02/01/2014	DhartiRathod
	1.1	Review	02/04/2014	Akanksha
	2.0	Final version	02/12/2014	Dharti
Risk management	1.0	Initial revision	01/28/2014	Sneha
	1.1	Review	02/05/2014	Slesha

	2.0	Final version	02/11/2014	Sneha
System Design Overview	1.0	Initial revision	01/28/2014	Akanksha
	1.1	Review	02/08/2014	Dharti
	2.0	Final version	02/11/2014	AKanksha
CBIS Prototype	1.0	Initial version	02/11/2014	Trupti
	1.1	Review	02/13/2014	Dharti
	2.0	Final version	02/15/2014	Trupti
Process Model	1.0	Initial revision	03/15/2014	Akanksha
	1.1	Review	03/30/2014	Sneha
	2.0	Final version	04/01/2014	Akanksha
Use case diagrams	1.0	Initial revision	01/28/2014	Karthik
	1.1	Review	02/25/2014	Slesha
	2.0	Final version	04/01/2014	Karthik
Use case Details	1.0	Initial revision	02/25/2014	Trupti and Slesha
	1.1	Review	02/31/2014	Dharti
	2.0	Final version	03/05/2014	Trupti and Slesha
Class Diagram	1.0	Initial revision	02/25/2014	Karthik
	1.1	Review	02/27/2014	Akanksha

	2.0	Final version	03/01/2014	Karthik
Sequence Diagram	1.0	Initial version	01/18/2014	Trupti
	1.1	Review	01/20/2014	Karthik
	2.0	Final version	03/25/2014	Trupti
Collaboration Diagram	1.0	Initial revision	03/11/2014	Karthik
	1.1	Review	03/25/2014	Dharti
	2.0	Final version	03/28/2014	Karthik
Activity Diagram	1.0	Initial revision	02/18/2014	Trupti
	1.1	Review	03/26/2014	Karthik
	2.0	Final version	03/25/2014	Trupti
Component Diagram	1.0	Initial revision	03/04/2014	Karthik
	1.1	Review	03/15/2014	Slesha
	2.0	Final version	03/31/2014	Karthik
Deployment Diagram	1.0	Initial version	03/18/2014	Karthik
	1.1	Review	03/25/2014	Sneha
	2.0	Final version	03/31/2014	Karthik
State Charts	1.0	Initial revision	03/25/2014	Karthik
	1.1	Review	03/30/2014	Akanksha
	2.0	Final version	04/01/2014	Karthik

UML Packages	1.0	Initial revision	02/25/2014	Karthik
	1.1	Review	03/02/2014	Slesha
	2.0	Final version	03/31/2014	Karthik
Training Plan	1.0	Initial revision	03/05/2014	Dharti
	1.1	Review	03/07/2014	Sneha
	2.0	Final version	03/14/2014	Dharti
Quality and test plan	1.0	Initial version	03/25/2014	Akanksha
	1.1	Review	03/31/2014	Dharti
	2.0	Final version	04/01/2014	Akanksha
CMM and KPA level 3,4. Function point Analysis.	1.0	Initial revision	02/26/2014	DhartiRathod
	1.1	Review	03/01/2014	Slesha
	2.0	Final version	03/10/2014	DhartiRathod
Function point estimation model	1.0	Initial version	03/18/2014	Dharti&Slesha
	1.1	Review	03/18/2014	Trupti
	2.0	Final version	03/19/2014	Dharti
Precedence Network	1.0	Initial revision	02/18/2014	Sneha
	1.1	Review	02/26/2014	Slesha
	2.0	Final version	03/18/2014	Sneha

Project Schedule	1.0	Initial version	03/18/2014	Karthik
	1.1	Review	03/31/2014	Dharti
	2.0	Final version	04/01/2014	Karthik
Future Enhancements	1.0	Initial version	03/18/2014	Trupti
	1.1	Review	03/25/2014	Slesha
	2.0	Final version	04/01/2014	Sneha

19. Individual Work log

Dharti

Team Member Name:	*Dharti*		Team leader's signature	*Slesha.V*
Instructor	*Dr. Richard Riehle*			
Student Id of Member:	*84213*		Course	Advance Software Engineering (SEN 942)
Project Start Date:	1/14/2014			
Project End Date:	4/1/2014			
Date	Task Assigned	Project Detail Completed /In Progress	Duration	Medium of discussion
Jan 14, 2014	Finalizing project topic	Completed	2 days	Group meeting.
Jan 21, 2014	Worked on first draft	Completed	1 week	Group meeting.

	of Project plan management			
Jan 28, 2014	Documenting Project plan.	Completed	2 days	Google doc.
Feb 04, 2014	Documenting training Plan	Completed	1 week	Group meeting
Feb 11, 2014	Function point estimation model	Completed	1 days	Group meeting
Feb 18, 2014	Project Scheduling	On going	3 ½ months	Group meeting and Google doc.
Feb 25, 2014	CMM and KPA level 3,4	Completed	1 week	Group meeting.
Mar 04, 2014	Worked on Meeting minutes	Completed	1 week	Group meeting.
Mar 11, 2014	Worked on individual work log and lessons learnt.	Completed	3 days	Google doc.
Mar 18, 2014	Worked on Revision history and Future enhancements.	Completed	5 days	Google doc and in class meeting.
Mar 25, 2014	Worked on Glossary and references. Final documentation.	Completed	4 days	Internet and Google doc.
April 1, 2014	Worked on Power point presentation	Completed	1 week	Word Power Point

Akanksha

Team Member Name:	*Akanksha*		Team leader's signature	*Slesha.V*
Instructor	*Dr. Richard Riehle*			
Student Id of Member:	*12300106*		Course	Advance Software Engineering (SEN 942)
Project Start Date:	1/14/2014			
Project End Date:	4/1/2014			
Date	Task Assigned	Project Detail Completed /In Progress	Duration	Medium of discussion
Jan 14, 2014	Discussion of project topic.	Completed	2 days	Group meeting, online chatting.
Jan 21, 2014	Research and Finalizing the Project	Completed	2 weeks	Group meeting, internet chat,

	topic			phone calls
Jan 28, 2014	System Design Overview	Completed	1 day	Via email
Feb 04, 2014	Software Architecture	Completed	1 day	Via email
Feb 11, 2014	Security Component in Design	Completed	2 days	Google doc
Feb 18, 2014	Process model	Completed	1 day	In class meeting.
Feb 25, 2014	Hardware and software interfaces	Completed	1 week	Internet and Software Engg Book
Mar 04, 2014	Quality and Test Plan	Completed	3 days	Discussed with Professor and Search Internet
Mar 11, 2014	Worked on individual worklog and lessons learnt.	Completed	3 days	Google doc.
Mar 18, 2014	Worked on Revision history and Future enhancements.	Completed	5 days	Google doc and in class meeting.
Mar 25, 2014	Worked on Glossary and references. Final documentation.	Completed	4 days	Internet and Google doc.
April 1,	Worked on Power	Completed	1 week	Word Power

2014	point presentation			Point

Slesha

Team Member Name:	*SleshaVemuganti*		Team leader's signature	*Slesha.V*
Instructor	*Dr. Richard Riehle*			
Student Id of Member:	*82938*		Course	Advance Software Engineering (SEN 942)
Project Start Date:	1/14/2014			
Project End Date:	4/1/2014			
Date	Task Assigned	Project Detail Completed /In Progress	Duration	Medium of discussion
Jan 14, 2014	Finalizing project title	Completed	2 days	Group meeting.
Jan 21,	Worked on 1st draft of	Completed	1 week	Google doc.

2014	functional requirements			
Jan 28, 2014	Worked on non functional requirements	Completed	3 days	Via email and Google doc.
Feb 04, 2014	Worked on Organizational and derived requirements	Completed	3 days.	Via email and Google doc.
Feb 11, 2014	Reviewed and finalized Requirement document.	Completed	1 day	In Class group meeting.
Feb 18, 2014	Worked on Real world requirements	Completed	1 week	Internet and Google doc.
Feb 25, 2014	Worked on 1st draft of detail use case theory	Completed	1 week	Internet and Software Engg Book
Mar 04, 2014	Worked on detailed use case - Advanced image search.	Completed	1 week	Discussed with Professor
Mar 11, 2014	Worked on individual worklog and lessons learnt.	Completed	3 days	Google doc.
Mar 18, 2014	Worked on Revision history and Future enhancements.	Completed	5 days	Google doc and in class meeting.
Mar 25, 2014	Worked on Glossary and references. Final	Completed	4 days	Internet and Google doc.

	documentation.			
April 1, 2014	Worked on Power point presentation	Completed	1 week	Word Power Point

Trupti

Team Member Name:	*Trupti Vala*		Team leader's signature	*Slesha.V*
Instructor	*Dr. Richard Riehle*			
Student Id of Member:	*84829*		Course	Advanced Software Engineering (SEN 942)
Project Start Date:	1/14/2014			
Project End Date:	4/1/2014			
Date	Task Assigned	Project Detail Completed /In Progress	Duration	Medium of discussion
Jan 14, 2014	Discussion of project topic.	Completed	2 days	Group meeting
Jan 21, 2014	Research and	Completed	2 weeks	Group meeting

	Finalizing the Project topic			
Jan 28, 2014	Worked on feasibility study of CBIR	Completed	1 day	Via email, Google doc
Feb 04, 2014	Worked on activity diagram	Completed	1 day	Via email, Google doc
Feb 11, 2014	Study and development of Prototype of CBIR	Completed	2 days	Internet, Reference book
Feb 18, 2014	Worked on usecase diagram - Basic search	Completed	1 day	Reference Book
Feb 25, 2014	Worked on use case diagram - Image search (User)	Completed	1 day	Reference Book
Mar 04, 2014	Worked on use case diagram - Image Search (Admin)	Completed	1 day	Reference book
Mar 11, 2014	Worked on individual worklog and lessons learnt.	Completed	3 days	Google doc.
Mar 18, 2014	Worked on Revision history and Future enhancements.	Completed	5 days	Google doc and in class meeting.
Mar 25, 2014	Worked on Glossary and references. Final	Completed	4 days	Internet and Google doc.

	documentation.			
April 1, 2014	Worked on Power point presentation	Completed	1 week	Word Power Point

Sneha

Team Member Name:	*Sneha*		Team leader's signature	*Slesha.V*
Instructor	*Dr. Richard Riehle*			
Student Id of Member:	84114		Course	Advance Software Engineering (SEN 942)
Project Start Date:	1/14/2014			
Project End Date:	4/1/2014			
Date	Task Assigned	Project Detail Completed /In Progress	Duration	Medium of discussion
Jan 14, 2014	Finalizing project topic	Completed	2 days	Group meeting.

Jan 21, 2014	Research and Finalizing the Project topic	Completed	1 Week	Group Meeting
Jan 28, 2014	Risk Management	Completed	2 Weeks	Via email
Feb 04, 2014	Technical Risk	Completed	2 Weeks	Via email
Feb 11, 2014	Non-Technical Risk	Completed	2 days	Via email
Feb 18, 2014	Precedence Network	Completed	1 Week	Group Meeting
Feb 25, 2014	Vision Statement	Completed	1 week	Group Meeting
Mar 04, 2014	Scope of Vision Document	Completed	1 Week	Via email
Mar 11, 2014	Worked on individual worklog and lessons learnt.	Completed	3 days	Google doc
Mar 18, 2014	Worked on Revision history and Future enhancements.	Completed	5 days	Google doc and in class meeting.
Mar 25, 2014	Worked on Glossary and references. Final documentation.	Completed	4 days	Internet and Google doc.
April 1, 2014	Worked on Power point presentation	Completed	1 week	Word Power Point

Karthik

Team Member Name:	*KarthikeyanSivanandi*		Team leader's signature	*Slesha.V*
Instructor	*Dr. Richard Riehle*			
Student Id of Member:	*12200010*		Course	Advanced Software Engineering (SEN 942)
Project Start Date:	1/14/2014			
Project End Date:	4/1/2014			
Date	Task Assigned	Project Detail Completed /In Progress	Duration	Medium of discussion
Jan 14, 2014	Finalizing project title	Completed	2 days	Group meeting.

Jan 21, 2014	Worked on Project concept.	Completed	1 week	Google doc.
Jan 28, 2014	Worked on Use case diagram version 1	Completed	3 days	Via email and Google doc.
Feb 04, 2014	Worked on UML Activity diagram	Completed	3 days.	Via email and Google doc.
Feb 11, 2014	Worked on Use case diagram version 1.1	Completed	1 day	In Class group meeting.
Feb 18, 2014	Worked on UML Sequence Diagrams and Activity Diagram.	Completed	1 week	Internet and Google doc.
Feb 25, 2014	Worked on UML Class Diagrams and Packages.	Completed	1 week	Internet and Software Engg Book
Mar 04, 2014	Worked on UML Component Diagrams.	Completed	1 week	Discussed with Professor and
Mar 11, 2014	Worked on UML Collaboration Diagrams.	Completed	3 days	Google doc.
Mar 18, 2014	Worked on UML Deployment diagrams.	Completed	5 days	Google doc and in class meeting.
Mar 25, 2014	Worked on UML State charts.	Completed	4 days	Internet and Google doc.
April 1, 2014	Worked on Power point presentation	Completed	1 week	Word Power Point

20. Lessons Learned

Name and ID	Lessons Learnt
Akanksha 123000106	• Learnt and understood UML diagrams. • Worked on all UML diagrams by using various tools to draw it. • Detailed research in Software requirement gathering and gaining better and detailed knowledge regarding CBIR project. • Gained better experience of working with team. • Helped team and always remained an active project mate.
Dharti Rathod 84213	• Detailed research in Software requirement gathering and estimation metrics. Gained better knowledge of CBIS project. • Learned to generate project estimation model. • Experience working with team, learning new things and collaborating with each team mates. • Make decision more competitive to improve the product. • Understood and learned details of Agile process model, project scheduling, planning the training sessions for CBIS team using CMM and KPA level 3,4 model. • Implemented various topics taught by professor into the project. I thank to my teammates and respected professor who gave me enough space to come up with my creativity.
Karthikeyan Sivanandi	• Detailed research on Project Scope. • Learned Class Diagrams, discussed with team mates and professor.

12200010	• Learned Sequence Diagram and Activity Diagrams. • Learned Collaboration Diagram, Component Diagram and Deployment Diagram. • Learned State Charts and Packages.
Slesha Vemuganti 82938	• Identified the skill set of each team member and assigning the task respective to their skill set. • Detailed research on requirements and gained better understanding about the scope of the system and its expected functionalities. • Following up with the team was a challenge due to individuals availability, so we had to schedule frequent meetings to have the status updates from the respective team members. • As a team, we had a continuous knowledge sharing and information gathering session which helped us to study and research on various challenging areas during the Project execution. • As part of team meetings, we had a brainstorming discussion where lot of enhancements to the system was proposed and executed. • Learned Use case diagrams, and explored a lot on this topic with team mates.
Sneha 84114	• Learned the importance of Vision Document in a project and how to document the same with high-level scope and purpose of a program. • It is important to understand team dynamics, strengths and weakness and psychology of each team member to ensure that all team members work together to achieve a common goal • Learned to draw precedence network diagram and how to calculate its Critical Path**.** • Learned the significance of Risk Management in defining potential risk and ways to mitigate them for the success of the project.
Trupti 84829	• Detail research on Content based Image Retrieval Systems – methods, algorithms, other currently available CBIR systems and comparison with

	one which is proposed here • Implemented CBIR activity diagram • Study of feasibility report of software system and developed feasibility report for the proposed system • Study of prototyping of the software system and learned how to develop prototype for the proposed system using web based tools • Wrote of detail use case and learned how to write use cases effectively • Developed report on future enhancement of the proposed system • Learned to work in team and manage tasks, effective communication between team members

21. FUTURE ENHANCEMENTS

There is good reason to believe that the saliency of images should play a major part in automated image retrieval. The work has indicated that laying emphasis upon areas of images that attract high visual attention can improve retrieval performance.

This CBIR application has some drawbacks and weaknesses that could be improved. Some of these drawbacks that could be improved in the future are the following:

User Interface improvements:

The current version of CBIR is quite simple and user friendly. As future enhancement, a more attractive UI with additional features can be proposed. Current system doesn't provide image suggestions upon selecting image. This is also import area of research for future improvements.

Image Matching Algorithm:

One of the main and core improvements is the Image Matching Algorithm. The system gets all images and sorts them by how close they are to the source image. However, the

sorting algorithm (image matching algorithm) could be further improved to obtain better precision.

Image Scaling:

It was found better to scale down images from 384x256 to 160x240. However images with sizes smaller than 160x240 would be scaled up and so would not obtain similar results. In addition, very large images would lose lots of visual information when scaled down to 160x240. Therefore it would be better to preprocess images in a certain way so that it will provide the same effect as scaling images down.

Human in Loop:

A fundamental difference between a Computer vision pattern recognition system and Image Retrieval System is that human is an indispensable part in the latter system. We need to explore the synergy of human and computer. For example, MARS team formally proposes Relevance Feedback architecture in Image Retrieval where human and computer can interact with each other to improve the retrieval performance.

High level concepts and Low level visual features:

Human tend to use high-level concepts in everyday life. However, what current computer vision techniques can automatically extract from images are mostly low level features. In constrained application, such as human face and fingerprints, it is possible to link low level features to high level concepts. This would require implementing supervised learning algorithms like neural nets, genetic algorithm and unsupervised clustering techniques.

Adding graphical view for Local and Global Color Histogram values in UI

The results of local and global color histograms are shown in numerical values. In future we are going to add graphical view of mean, standard deviation etc. to have greater understanding on the differences in global and local color histograms.

22. CODE

```
public abstract class  Classifier
{
 public static Vector classify(ImageData input, Vector dataSet)
 {
        Vector<Vector> result = new Vector<Vector>();
               LinkedHashMap<String, Double> imMap = new  LinkedHashMap<String, Double>();
               LinkedHashMap<String, Double> imStatMap = new  LinkedHashMap<String, Double>();
        for (int x=0;x< dataSet.size();x++ )
        {
          double distance = getDistanceMeasure(input.getImageFeature(),((ImageData)dataSet.get(x)).getImageFeature());
                 imMap.put(((ImageData)dataSet.get(x)).getImageName(),new Double(distance));
                 double statDistance = getDistanceMeasure(input.getImageStatFeature(),((ImageData)dataSet.get(x)).getImageStatFeature());
                 imStatMap.put(((ImageData)dataSet.get(x)).getImageName(),new Double(statDistance));

        }
       LinkedHashMap imSortedMap = getSorted(imMap);
               LinkedHashMap imStatSortedMap = getSorted(imStatMap);
               Vector r1 = getImageData(imSortedMap, dataSet);
               Vector r2 = getImageData(imStatSortedMap, dataSet);
        result.add(r1);
        result.add(r2);
        return result;
 }
public static double getDistanceMeasure(Vector a, Vector b)
    {
      int noOfDimension = a.size();
      if(noOfDimension != b.size())
      return 99999.8999999999994D;
      double total = 0.0D;
      for(int i = 0; i < noOfDimension; i++)
       {
          double doubleA = ((Double)a.elementAt(i)).doubleValue();
          double doubleB = ((Double)b.elementAt(i)).doubleValue();
          total += (doubleA - doubleB) * (doubleA - doubleB);
       }
       return Math.sqrt(total / (double)noOfDimension);
   }
       public static LinkedHashMap getSorted(LinkedHashMap imMap) {
        LinkedHashMap<String, Double> imSortedMap = new  LinkedHashMap<String, Double>();
        java.util.List imMapKeys = new ArrayList(imMap.keySet());
        java.util.List imMapValues = new ArrayList(imMap.values());
        TreeSet sortedSet = new TreeSet(imMapValues);
        Object[] sortedArray = sortedSet.toArray();
        int size = sortedArray.length;
    for (int i=0; i<size; i++) {
        imSortedMap.put((String)imMapKeys.get(imMapValues.indexOf(sortedArray[i])),((Double)sortedArray[i]).doubleValue());
    }
       return imSortedMap;
       }
```

```
public class  cbir
{
Vector<ImageData> dataSet;

public cbir(String path){

        try
        {
                dataSet = new Vector<ImageData>();

                BufferedReader br = new BufferedReader(new FileReader( new File(path, "data.txt")));

                while(true)
                {

                ImageData imData = new ImageData();
                String line = br.readLine();
                if (line == null) break;
                if (line.startsWith("#")) continue;
                StringTokenizer st = new StringTokenizer(line);
                if (st.countTokens() < 2) continue;
                imData.setImageName(st.nextToken());
                imData.setImageClass(st.nextToken());
                dataSet.add(imData);
                }
                br.close();
                for (int i=0;i<dataSet.size();i++ )
                {
                ImageData imData = (ImageData)dataSet.get(i);
                BufferedImage bi = ImageIO.read(new File(path,imData.getImageName()));
                CBIRImage cbirImage = new CBIRImage(bi);
                Vector histogram = Features.getGlobalColorHistogram(cbirImage);
                Vector<Double> statFeature = new Vector<Double>();
                double mean = Features.getMean(histogram);
                double sd = Features.getSD(histogram, mean);
                double skew = Features.getSkew(histogram, mean, sd);
                double energy = Features.getEnergy(histogram);
                double entropy = Features.getEntropy(histogram);
                statFeature.add(new Double(mean));
                statFeature.add(new Double(sd));
                statFeature.add(new Double(skew));
                statFeature.add(new Double(energy));
                statFeature.add(new Double(entropy));
                imData.setImageFeature(histogram);
                imData.setImageStatFeature(statFeature);
                imData.setImageMean(mean);
                imData.setImageSD(sd);
                imData.setImageSkew(skew);
                imData.setImageEnergy(energy);
                imData.setImageEntropy(entropy);
                }

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

}
```

```
public abstract class Features
{

    public static Vector getGlobalColorHistogram(CBIRImage image)
    {
        Vector globalColorHistogram = new Vector(64);
        int width = image.getWidth();
        int height = image.getHeight();
        int size = width * height;
        int pixels[] = image.getPackedRGBPixel();
        double histogram[] = new double[64];
        for(int i = 0; i < 64; i++)
            histogram[i] = 0.0D;
        for(int i = 0; i < size; i++)
        {
            int k = pixels[i] >> 18 & 0x30 | pixels[i] >> 12 & 0xc | pixels[i] >> 6 & 3;
            histogram[k]++;
        }
        for(int i = 0; i < 64; i++)
            globalColorHistogram.add(new Double(histogram[i] / (double)size));
        return globalColorHistogram;
    }
        public static double getMean(Vector histogram){
        double mean = 0.0D;
        double sum = 0.0D;
        for (int i=0;i<histogram.size() ;i++ )
        {
            sum += i*((Double)histogram.get(i)).doubleValue();
        }
          return (double)sum;
        }
        public static double getSD(Vector histogram, double mean){
       double sum = 0.0D;
           for (int i=0;i<histogram.size() ;i++ )
        {
            sum += Math.pow((i- mean),2)*((Double)histogram.get(i)).doubleValue();
        }
       return Math.sqrt((double)sum);
        }
        public static double getSkew(Vector histogram, double mean, double sd) {
           double sum = 0.0D;
           for (int i=0;i<histogram.size() ;i++ )
        {
            sum += Math.pow((i- mean),3)*((Double)histogram.get(i)).doubleValue();
        }
           return (double)sum/Math.pow(sd,3);
        }

        public static double getEnergy(Vector histogram) {
         double energy = 0.0D;
          for (int i=0;i<histogram.size() ;i++ )
        {
            energy += Math.pow(((Double)histogram.get(i)).doubleValue(),2);
        }
        return energy;
        }
```

23. PROJECT GLOSSARY.

CBIR : (Content Based Image Retrieval)- The process of retrieving images based on visual features such as texture and color.

Color distances: The degree of similarity between two color histograms represented by numerical distance. The closer the distance is to zero the more similar two images are in terms of color. The further away the distance is from zero, they are less similar.

Color histogram: A histogram that represents the color information of an image. The x-axis represents all the different colors found in the image. Each specific color is referred to as a bin. The y-axis represents the number of pixels in each bin.

Color space: The three dimensional space in which color is defined.

Data conditioning: The use of **data** management and optimization techniques which result in the intelligent routing, optimization and protection of **data** for storage or **data** movement in a computer system.

Encryption: Is the process of encoding messages or information in such a way that only authorized parties can read it.

Enumerate: Mention (a number of things) one by one.

Euclidean distance: Equation used to compare average intensities of pixels.

Feature extraction: In pattern recognition and in image processing, **feature extraction** is a special form of dimensionality reduction.

Global color histogram: The color histogram of the whole image.

HSV: Hue Saturation Value color space.

Iteration: The repetition of a process or utterance.

Local color histogram: The color histogram of a subsection of an image.

Quadratic metric: Equation used to calculate the color distance. It consists of three terms, color histogram difference, transpose of color histogram difference and similarity matrix.

Quantization: Reducing the number of color bins in a color map by taking colors that are very similar to each other and putting them in the same bin.

QBIC: CBIR system developed by IBM.

RGB: Red Green Blue color space**.**

Similarity matrix: Matrix that denotes the similarity between two sets of data. This is identified by the diagonal of the matrix. The closer the sets of data, the closer the diagonal is to one. The farther the sets of data, the farther the diagonal is from one.

24. RESOURCE REFERENCES.

1. Barbeau Jerome, Vignes-Lebbe Regine, and Stamon Georges, "A Signature based on Delaunay Graph and Co-occurrence Matrix," Laboratoire Informatique et Systematique, Universiyt of Paris, Paris, France, July 2002, Found at: http://www.math-info.univ-paris5.fr/sip-lab/barbeau/barbeau.pdf
2. Sharmin Siddique, "A Wavelet Based Technique for Analysis and Classification of Texture Images," Carleton University, Ottawa, Canada, Proj. Rep. 70.593, April 2002.
3. Thomas Seidl and Hans-Peter Kriegel, "Efficient User-Adaptable Similarity Search in Large Multimedia Databases," in Proceedings of the 23rd International Conference on Very Large Data Bases VLDB'97, Athens, Greece, August 1997, Found at: http://www.vldb.org/conf/1997/P506.PDF
4. FOLDOC, *Free On-Line Dictionary Of Computing*, "cooccurrence matrix," May 1995, [Online Document], Available at: http://foldoc.doc.ic.ac.uk/foldoc/foldoc.cgi?cooccurrence+matrix
5. Colin C. Venteres and Dr. Matthew Cooper, "A Review of Content-Based Image Retrieval Systems", [Online Document], Available at: http://www.jtap.ac.uk/reports/htm/jtap-054.html
6. Linda G. Shapiro, and George C. Stockman, *Computer Vision*, Prentice Hall, 2001.
7. Shengjiu Wang, "A Robust CBIR Approach Using Local Color Histograms," Department of Computer Science, University of Alberta, Edmonton, Alberta, Canada, Tech. Rep. TR 01-13, October 2001, Found at: http://citeseer.nj.nec.com/wang01robust.html
8. R. Jain, R. Kasturi, and B. G. Schunck, *Machine Vision*, McGraw Hill International Editions, 1995.
9. FOLDOC, *Free On-Line Dictionary Of Computing*, "texture," May 1995, [Online Document], Available at: http://foldoc.doc.ic.ac.uk/foldoc/foldoc.cgi?query=texture
10. "Texture," class notes for *Computerized Image Analysis MN2*, Centre for

Image Analysis, Uppsala, Sweden, Winter 2002, Found at: http://www.cb.uu.se/~ingela/Teaching/ImageAnalysis/Texture2002.pdf

11. G. D. Magoulas, S. A. Karkanis, D. A. Karras and M. N. Vrahatis, "Comparison Study of Textural Descriptors for Training Neural Network Classifiers", in Proceedings of the 3rd IEEE-IMACS World Multi-conference on Circuits, Systems, Communications and Computers, vol. 1, 6221-6226, Athens, Greece, July 1999, Found at: http://www.brunel.ac.uk/~csstgdm/622.pdf
12. Pravi Techasith, "Image Search Engine," Imperial College, London, UK, Proj. Rep., July 2002, Found at: http://km.doc.ic.ac.uk/pr-p.techasith-2002/Docs/OSE.doc
13. Bjorn Johansson, "QBIC (Query By Image Content)", November 2002, [Online Document], Available at: http://www.isy.liu.se/cvl/Projects/VISIT-bjojo/survey/surveyonCBIR/node26.html
14. FOLDOC, *Free On-Line Dictionary Of Computing*, "wavelet," May 1995, [Online Document], Available at: http://foldoc.doc.ic.ac.uk/foldoc/foldoc.cgi?query=wavelet
15. Lexico Publishing Group, LLC, "shape", [Online Document], Available at: http://dictionary.reference.com/search?q=shape
16. Benjamin B. Kimia, "Symmetry-Based Shape Representations," Laboratory for Engineering Man/Machine Systems (LEMS), IBM, Watson Research Center, October 1999, Found at: http://www.lems.brown.edu/vision/Presentations/Kimia/IBM-Oct-99/talk.html
17. Marinette Bouet, Ali Khenchaf, and Henri Briand, "Shape Representation for Image Retrieval", 1999, [Online Document], Available at: http://www.kom.e-technik.tu-darmstadt.de/acmmm99/ep/marinette/