**Searching in large image databases**

Authors: Dima Vlad MOC1

Aciobanitei Bogdan MOC1

Chiriac George MISS1

**Cuprins**

1. Introduction
   1. Purpose of the system
   2. Scope of the System
   3. Objectives and success criteria of the Project
2. Proposed system
   1. Overview
   2. Functional requirements
   3. Nonfunctional requirements
      1. User interface and human factors
      2. Documentation
      3. Performance characteristics
      4. Error handling
      5. Security issues
3. System models
   1. Scenarios
   2. Use case model
   3. Class diagrams
4. **Introduction**

The idea of this project (build an image retrieval system) came from the fact that very few systems exist that allows users to query an image database via images. One big player in image retrieval system is Google which uses a set of algorithms to analyze image attributes such as color, shape, texture along side with keywords to refine searches on extremely large collections.

* 1. **Purpose of the system**

The main purpose of the system is to allow users to search through a image database using a query image based on the color information. They can do this using our web application or via an API provided by the system for registered users.

* 1. **Scope of the System**
  2. **Objectives and success criteria of the Project**

The main objective is to provide a fast, robust and scalable system for searching in big databases of images based on the color information of an query image. As success criteria we will use the speed of response and also the amount of space required by the system to provide the desired functionality.

1. **Proposed system**

We propose a distributed system that is divided in 3 main modules: Web Application, File Master and File Slave. We chose a distributed system because it allows us to execute our search queries in a paralel manner. Also, the system will provide a tool for testing it`s performance under different stress conditions such as overloading it with requests, searching speed in a big database etc.

* 1. **Overview**

The main modules and their sub-modules:

**Web Application** - The application users use it via a browser to access our searching system

**File Master** – provides an API for the Searching System that will be used by the Web Application or registered users.

**File Slave** – provides an API for File Master to execute queries on it`s database of images.

**Query Protocol** – Implements a standardized communication between the system modules

**Cache System** – Provides a caching functionality for similar query images.

**Searching System** (File Master) – Extracts features from the query images and sends them using Query Protocol to all File Slaves registered to that File Master.

**Clusterization System** (File Master) – provides a clusterization algorithm for all the File Slaves registered to the File Master based on the color information of the images stored on File Slaves.

**Searching System** (File Slave) – provides functionalities for storing and searching through the images from the File Slave.

**Searching Algorithms** – provides different algorithm for searching through the database. Also, must provide functionalities to extract feature vectors from a big database of images and store them in the database.

* 1. **Functional requirements**
  2. **Nonfunctional requirements**
     1. **User interface and human factors**

For a image search using our system, a simple interface is provided by the Web Application component, that allows users easy access to it without the need of understanding how it works. For testing the system, we provide an interface for File Slave module that requiers an abstract understanding of the algorithms used for the search.

* + 1. **Documentation**

Before any implementation was done, UML diagrams were created (Use Case Diagram, Sequence Diagram, State Diagram, Deployment Diagram, Package Diagram, Class Diagram) to allow an easy and fast understanding of the system. When it comes to code, we used a naming convention and in places where code was not suggestive enough, comments were added.

* + 1. **Performance characteristics**
    2. **Error handling**

In the process of implementing the system we tried to make it robust to errors as much as possible. Because it is almost impossible to oversee all errors that can occure, we implemented a logging system that allows us to identifie the source of errors fast and some additional information about it.

* + 1. **Security issues**

When it comes to security, we implemented a Query Protocol that allows modules to register members in the system (for example a File Slave wants to register to a File Master), to accept queries only from trusted members of the system.

1. **System models**
   1. **Scenarios**
   2. **Use case model**
   3. **Class diagrams**