# FaceMatch2 System Requirements

Updated: May 17, 2017

This document presents the high level requirements and system overview for the re-architected FaceMatch (FM) system, referred to as FaceMatch2 or FM2, to support the People Locator (PL) face matching capabilities. However, FM2 should be flexible enough to serve other clients in future.

This document does not address the requirements for the actual image processing and face detection software, referred to here as the FMCoreLib.

## Requirements

**A.1) Functional Requirements**

The functional capabilities to be provided by FM2 are as follows:

1. Maintain a repository of image datasets, with each image identified by a specific group ID (e.g. PL Event ID) and a unique handle (the URL), along with its unique features set (descriptors). Do not store the actual image in the FM repository. (The should be retrievable by the client from its own repository using the URL)
2. Provide a service for the client to ingest images to this repository - either singly or in a batch, but with only one image at a time.

* Allow additional metadata (such as gender and age) to be provided for the image, if available, so as to speed up retrieval of the image.
* Localize faces in each image and store the information (face descriptors) for each image at the time of ingest.

1. Provide a service to the client to find and return faces in an input image.
2. Provide services to the client to remove an image, or a face region from the repository.
3. Support user queries against the image repository to find one or more images matching an input image, and with specified search criteria. Also provide the same capability for a specific face region only.

**A.2) Performance Requirements**

The FM system must meetcertain performance criteria, so as to be operationally viable. The important ones include service time for:

* 1. Finding faces in an image
  2. Searching for a matching image , *or a face*, in the repository
  3. Storing/retrieving records for an image in the repository and the database

Note that these functions, especially 1 and 2, use various image processing algorithms, software logic and different system hardware components. So contribution of each entity to the overall performance should be measured and reported separately.

**A.3)** **Architectural Requirements**

FaceMatch2 system should be implemented as a set of standard Web services, available through a CEB public facing Web Server.

* + The code developed by CEB should include:
  + C++ modules to perform image processing functions, already implemented and tested by FM, and any future enhancements to them - referred to as FMCoreLib.
  + Java modules to provide the high level Web service protocols, and interface with the FMCoreLib. These modules are collectively referred to here as FM2WebServices.
  + Depending upon the type of repository chosen (see Section A.5), the image data sets may be stored/retrieved directly by the FM2WebServices and passed on to the FMCoreLib, specifically to keep the image processing modules independent of the repository structure.
  + Code for both FMCoreLib and FM2WebServices should be portable so as to run either on Linux/Red Hat (RHEL 6 or later) machines or a Windows-based (W7 or later) machines - so that if external hardware/software packages used by FM2 have implementations available for these operating systems, the FM2 service may run on either one.
* FMCoreLib will use the OpenCV open source computer vision package to implement the low level image processing functions. Other requirements involving OpenCV are discussed under section A.4 below.
  + The Web services provided by FM2 should be mapped to corresponding Java servlets managed by the Tomcat Web Server. FMCoreLib and associated C++ functions should be implemented as one or more Shared Object libraries (.so) on Linux and Dynamic Link Libraries (.DLL) on Windows platforms.

**A.4) Hardware and Computing Requirements**

* FaceMatch2 system must use the recommended general purpose Graphic Processing Unit (GPU) for performing various computation-intensive tasks, specifically for:
  + Extracting features from an image and storing it in a structure defined as Image Descriptor
  + Finding faces in an image and matching whole images
* For querying images, GPUs may be used to help in search, if memory and other requirements are met.
* For other tasks, parallel processing using the system’s multiple CPUs should be implemented wherever appropriate, using the OpenMP directives provided by the C++ compiler. For Java modules, these may be implemented through *concurrent* threads.
* The FM2 system should be able to discern whether a given function is performed using the GPU or the CPU. In general, all access to the GPU would be directed through the OpenCV library (except at FM initialization time, if needed, to check the system configuration).
* Although FM2 would use the currently available NVidia GPUs, to allow the use of other GPUs in future, it should not use the NVidia CUDA interface directly or through OpenCV, but instead use the generalized OpenCL interface, which is supported by NVidia drivers.

*(Note: This requires redesign/implementation of FMCoreLib, and therefore removed from FM2 direct requirement list.)*

**A.5) Repository Requirements for Image Dataset**

The choice of the FaceMatch2 repository to store image datasets depends upon the anticipated maximum number of images to be stored, and acceptable search performance. Under the current FaceMatch system (FM1), the repository consists of a set of flat files, called “buckets”, in the FM host file system, named using the external metadata (age, gender,…) associated with an image. Descriptors for an incoming image are stored in the corresponding bucket, and linear search for an image at retrieval time is confined to that bucket, enhancing performance. Still it represents a weak link in the FaceMatch system due to the following main reasons:

* It is not a formal solution for large scale image search where large number of image datasets must be read into memory from buckets prior to any search.
* Writing indexed data for ingested images from memory to file system at certain intervals is not an operationally viable procedure. If the system goes down within an indexing interval, all images ingested during that period would be lost.
* There is no transparency or high level information w.r.t. the image data stored at the FM facility.

Thus, following alternatives should be investigated, in the specified order of preference:

* Migrating the entire repository to a NoSQL database- where descriptors for each image/face are kept as a separate record.
* Storing each descriptor as a separate file (bit stream) and keeping all ancillary information in a simple SQL database with pointer to the corresponding file.

Furthermore improved search mechanism using clustering algorithms as well as utilizing the GPU rather than the CPU should also be investigated in conjunction with FMCoreLib updates.

## FM2 Web Services

FaceMatch2 Web services will be implemented as standard Java servlets running under Tomcat and handling *HTTP-based REST* (REpresentational State Transfer) *requests* from the client. As in the current version of FM, there would be three Web services, equivalent to current FaceFinder, FaceMatchRegions, and WholeImageMatcher, with no disruption to PL, except as noted below.

Under FM2, PL’s FM Web client will have to use REST-based API rather than the current SOAP protocols. This decision is made based upon the simplicity of using REST vs. SOAP and the current trend of Web services moving away from SOAP, unless there are compelling reasons to use it.( For example: see <http://searchsoa.techtarget.com/tip/REST-vs-SOAP-How-to-choose-the-best-Web-service>)

High level function of each of these Web services should be the same as the corresponding FaceMatch service of the same name so as not to impact the existing FM client (PL). For completeness, these functions are mentioned below:

1. FaceFinder: Find faces in a given image and return the corresponding annotation.
2. FaceMatchRegions:
3. Ingest an image: Find faces in the image and store the information (descriptors) against the image ID and its group ID (eventID for PL).
4. Remove a previously ingested image
5. Query: retrieve an image or a face matching the search criteria
6. Index: write the stored image data for an ingested image to the repository. However, this would be combined with (2.a) rather than a client invoked service.
7. WholeImageMatcher: Perform same functions as in 2 above without attempting to find faces. Applicable to any type of imagery. Note that presently there are no clients for this service as it is not used by PL. Its implementation may be deferred if necessary.

## System Overview

The components of the system and their hierarchical dependencies, compliant with system requirement discussed in Sections A and B, are presented in Figure 1.

**Figure 1 – FaceMatch2 system components**

**Boxes indicate CEB/FM developed software modules**

***FaceMatch Platform (Linux)***

***Index files***

FaceMatch Data Repository

***References***

**FaceMatch2 Servlets**

**Java**

**OpenCL**

**C++**

**C++**

**OpenCV**

**FaceMatch2 Server**

***GPU (NVidia)***

**FaceMatch Core Lib (.SO)**

**FaceMatch Lib Interface (.SO)**

**Tomcat Web Server**

Response

RESTful Request