

**ANDROLOGISTS**

# PROJECT: FACIAL RECOGNITION SYSTEM FOR COMMUNITIES

ZANE BLOOM – 12236722

HEELIN MISTRY – 10299344

VERUSHKA MOODLEY – 29117454

GERARD LAMUSSE – 12206050

1 August 2014

**Contents**

[**PROJECT: FACIAL RECOGNITION SYSTEM FOR COMMUNITIES**](#h.30j0zll)

[**Vision and Scope**](#h.3znysh7)

[**Project Background and Vision**](#h.2et92p0)

[**Scope and Limitations/Exclusions**](#h.1t3h5sf)

[**Scope**](#h.4d34og8)

[**Exclusions/Limitations**](#h.17dp8vu)

[**Architectural Requirements**](#h.3rdcrjn)

[**Access and integration requirements**](#h.26in1rg)

[**Access channels**](#h.lnxbz9)

[**Protocols**](#h.23ckvvd)

[**Quality requirements for access and integration channels**](#h.35nkun2)

[**Architectural responsibilities**](#h.44sinio)

[**Quality requirements**](#h.8vzr91cpe6hb)

[**Security**](#h.z337ya)

[**Testability**](#h.3j2qqm3)

[**Usability**](#h.1y810tw)

[**Scalability**](#h.4i7ojhp)

[**Performance requirements**](#h.2xcytpi)

[**Architectural Constraints**](#h.1ci93xb)

[**Software Architecture**](#h.3whwml4)

[**Architectural Requirements**](#h.2bn6wsx)

[**Architectural scope**](#h.qsh70q)

[**Architectural Patterns/Styles**](#h.1pxezwc)

[**Architectural Tactics/Strategies**](#h.49x2ik5)

[**Reference architectures and frameworks**](#h.1xyjqls3oz03)

[**Technologies**](#h.ihv636)

# Vision and Scope

## Project Background and Vision

Quant Solutions requires an automated facial recognition system that will be implemented in communities/neighbourhoods. The main purpose of this system is to identify those people that are regulars in the neighbourhood and those that aren’t. This system will also allow for story verification (if a person is accused of a crime in the neighbourhood, this system should place him/her at the scene).

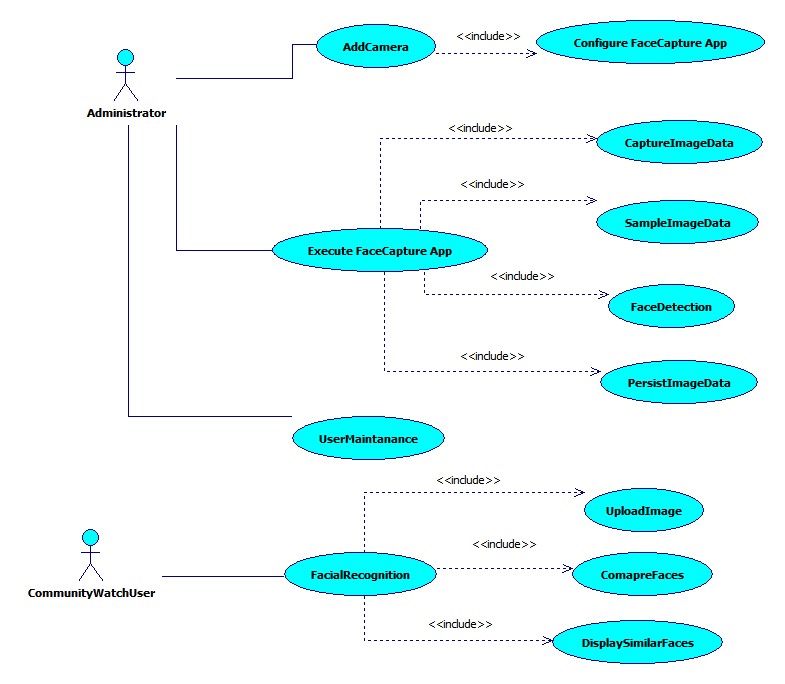
Quant Solutions already has 32 cameras placed around the neighbourhood in different street corners. An individual process will be started for each camera. This process will capture, process and persist images from the camera.

The system will also consist of a web application to query the backend system. The web application will be used to take a picture of the suspicious individual, and the picture will be compared to the data stored in database in the backend. The web application will then display possible matches and the user can scroll through these results and form a conclusion about the suspicious individual.

## 

## Scope and Limitations/Exclusions

### Scope



### 

### Exclusions/Limitations

* The system must detect frontal faces.
* The system is only required to work during the day (for lighting reasons).
* The system is not expected to be 100% accurate.

# Architectural Requirements

This section discusses the software architecture requirements that encompass the software infrastructure within which the application functionality is to be developed. The purpose of this infrastructure is to address the non-functional requirements. In particular, the architecture requirements specify

* The architectural responsibilities which need to be addressed
* The access and integration requirements for the system
* The quality requirements
* The architecture constraints specified by the client

## Access and integration requirements

This section discusses

1. The requirements for the different channels through which the system can be accessed by the people and the system
2. The integration channels which must be supported by this system.

### Access channels

The system will be accessible through the following channels:

The facial recognition will be made available through Restful web services.

A web-app frontend will act as the interface between the users and the web services.

The system must provide a web access channel, which will be mainly used by the administrator, as well as a mobile access channel, which will be used by the other users.

### Protocols

1. RESTful: The web application will use RESTful services to communicate data with the web server.
2. HTTPS: The web front end must use the HTTPS protocol to communicate data. This ensures secure transmission.
3. HTTP/RTSP: HTTP or RTSP protocols will be used to connect to the camera for live streaming.

### Quality requirements for access and integration channels

* All communication of sensitive data must be done securely using HTTPS. All other data can be communicated using HTTP.
* Connect to cameras via either HTTP or RTSP.
* FTP server.
* Postgre - message-based-protocol over TCP/IP
* Web App with web services

## Architectural responsibilities

The architectural responsibilities include the responsibilities of providing an infrastructure for

1. A web access channel, which will be HTTP or HTTPS
2. The Apache web server will be made use of and PHP will be used as a server side scripting language which will apply web services.
3. Hosting and providing the execution environment for the services of the system.

## Quality requirements

The quality requirements are the requirements around the quality attributes of the system and the services it provides. This includes performance, scalability, security, auditability, usability, and testability requirements.

### Security

Web Application

1. The user must be authenticated to use the facial recognition system, authentication will use a login detail with a unique password.
2. User should request from the management portal administrator their unique password.

Management Portal

1. Only one user (administrator) will be allowed access to the management portal, the user will only need to supply a password to access the management portal, the password will be supplied to them.
2. The management portal will be allowed to add/remove users to the web application, unique passwords will be issue for that user by the system.
3. The management administrator password will be supplied in the technical manual

### Testability

All services offered by the system must be testable through unit tests which test

* That the service is provided if all pre-conditions are met (i.e. that no exception is raised except if one of the pre-conditions for the service is not met)
* That all post-conditions hold true once the services has been provided.

### Usability

1. 90% of the users should be able to use the system without prior training. A user manual will be provided for the remaining 10%.

### Scalability

1. The system must be able to scale to 32 cameras, however multiple machines can be used to achieve this, thus the system needs to be scalable across multiple machines.
2. The deployed system must be able to operate effectively under the load of 20 concurrent users.
3. The software architecture should be such that it can, in future, be easily modified to scale by porting the system onto clustered and cloud-computing based architectures.

### Performance requirements

This system does not have particular stringent performance requirement, however, the data to be processed should not back up to a position where it will not be able to reach, due to new data to be processed incoming all the time.

Web Application Clients

1. Uploading of image depends on network being used and image size.

Facial Recognition system

1. Less than 100ms to compare two faces.

Facial Detection from camera

1. Important to process video streams in real time.
2. Processing of image frame:
   1. Less than 300ms to detect and crop a face in an image frame
   2. Less than 100ms to persist an image.
   3. Less than 300ms to pre-process face detected.

## Architectural Constraints

1. The system services must be made available to smart devices.
2. The system must use the *PostgreSQL* database.
3. Web services must be published as Restful-based web services.
4. Must make use of C++ and OpenCV for image processing.
5. Must be executable on Linux.
6. Must offer web services to smartphones and desktops.
7. Must only open source libraries.
8. Can be deployable on raspberry clients.

# Software Architecture

## Architectural Requirements

### Architectural scope

Provisions:

* Web-application access channel
* Capture images from cameras
* Persist facial images and metadata of the image to database
* Facial Recognition (Comparing faces)
* Infrastructure for process execution
* User interfaces with login, maintenance and functionality to query databases.

## 

## Architectural Patterns/Styles

The system will be built using the Pipes and filters architectural pattern. Each filter will be a stage in the pipeline that does a portion of the processing of an image from a video feed. The Strategy design pattern was used throughout our program to allow for inheritance and to allow for the subclasses to implement functions specific to the desired purpose. The system must be flexible in allowing

The Capturers (Pump/Producer):

* Stream capturing: capture images from a video stream.
* Directory capturing: capture images stored in directory.

The filters:

* Sampling: Sample the video stream for frames.
* Detection: Detect faces in images.
* Image Pre-processing: pre-process images to ensure more accurate results.

The Persister (Sink/Consumer):

* Local persistence: store the images locally on the same machine
* Database persistence: store the images and timestamp to the database

## 

## 

## Architectural Tactics/Strategies

1. Reduce coupling by using pipes and filters and decrease dependencies between modules of the system.
2. Increase Cohesion by giving each module a single responsibility.

## Reference architectures and frameworks

The system will be built using the Qt application framework. This framework offers features such as Qt SQL for database persistence, qmake build tool for cross platform deployment, Qt Test for testing and Qt network for network functionality. These features offered by Qt are the reason for it being the chosen framework.

The system will also make use of the OpenCV framework. It offers many features such as facial detection and facial recognition that the system will make use of. This framework was recommended by the client.

## Technologies

* The C++ and Java Programming languages will be used.
* PostgreSQL is the database management system that will be used.
* X86 servers will be used.
* Vivotek 8332 cameras will be used.
* Linux and Windows operating Systems will be used.

# Functional Requirements

# Change Log

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Person |
| 23-06-2014 | 0.1 | Initial Document | Zane Bloom |
| 23-06-2014 | 0.1.1 | Added Pipeline Image Processing | Zane Bloom |
| 24-06-2014 | 0.1.2 | Added Image PreProcessing Filter | Zane Bloom |
| 24-06-2014 | 0.1.3 | Added Image FaceDetect Filter | Heelin Mistry |
| 25-06-2014 | 0.1.4 | Added Sampling Filter | Verushka Moodley |
| 29-06-2014 | 0.1.5 | Added Capture | Heelin Mistry |
| 01-07-2014 | 0.2 | Added Persister |  |
| 02-07-2014 | 0.2.1 | Added Database Persister | Verushka Moodley |
| 23-07-2014 | 0.2.2 | Added Domain Objects | Zane Bloom |
| 31-07-2014 | 0.2.3 | Updated UML model and added Facial Recognition | Zane Bloom |

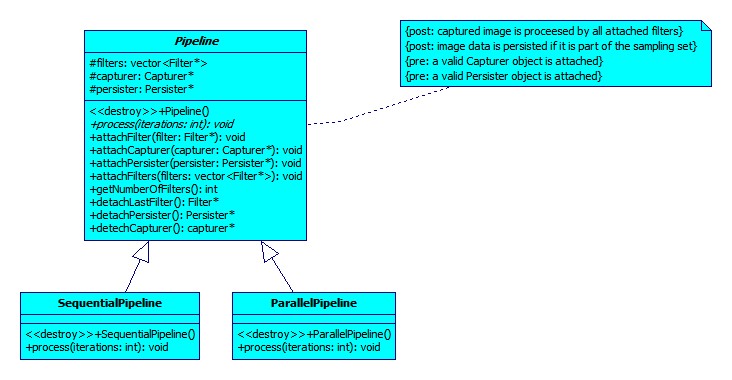
## Pipeline attachments

The user must be able to attach a Capturer, Persister and zero to many Filters to the pipeline.

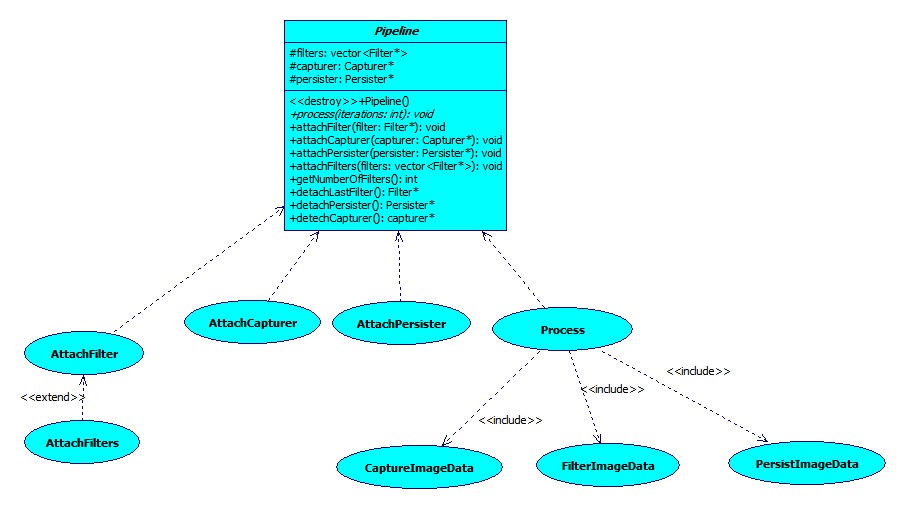
## Pipeline Image Processing

The pipeline must capture an image from a camera, push the image through the attached filters and persist the image and its data for reference later on.

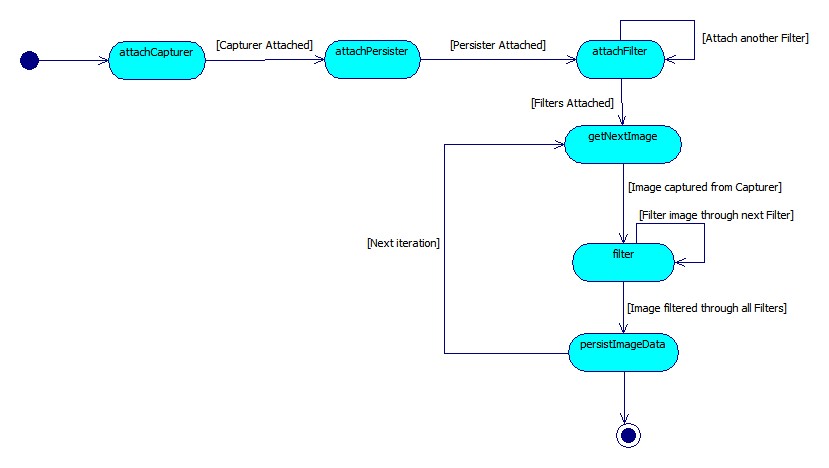
### Service Contract



### Functional Use Case

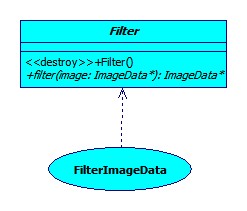


### Sequential Pipeline Process Specification

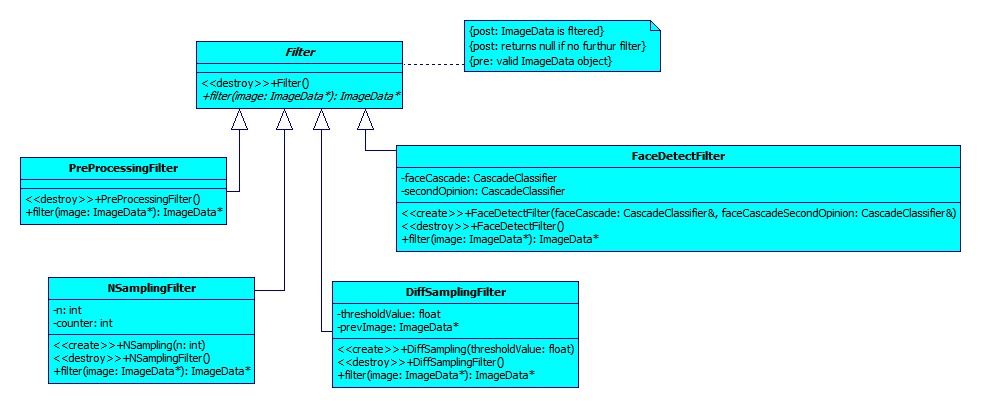


## Filtering an Image

### Functional Use Case



### Service Contract

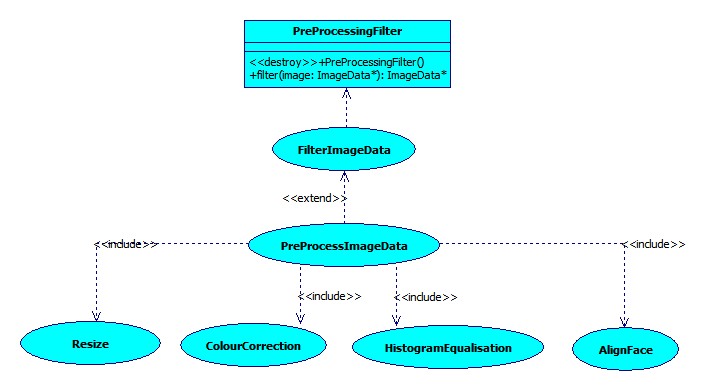


## PreProcessing an Image

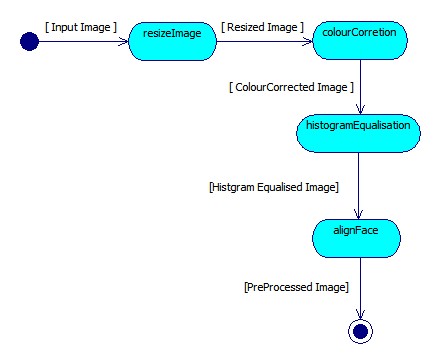
An image has to be pre-processed to ensure better facial recognition accuracy. Image pre-processing includes:

* Resizing the images to the same size. Images have to be the same dimensions in order to use them for facial recognition.
* Aligning the face.
* Correcting the colour of the image. Images must be made greyscale as colour images are more susceptible to lighting conditions.
* Histogram equalisation of the image. This includes standardizing the brightness and contrast of the image.

### Functional Use Case



### Image PreProcessing Process Specification

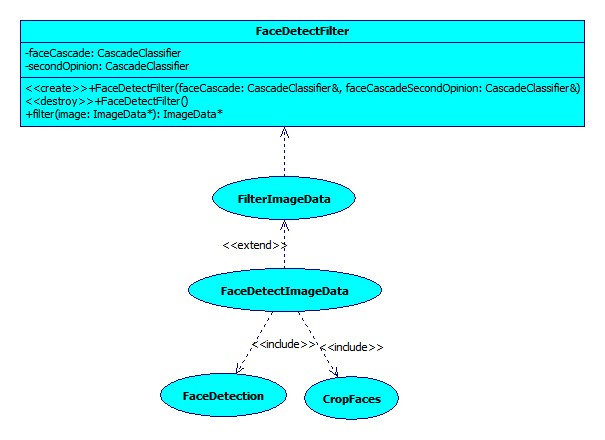


## FaceDetectFilter

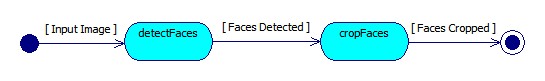
The FaceDetectFilter must be able to:

* Identify all faces within a frame.
* Crop facial images and add them to face vector for further processing.

### Functional Use Case



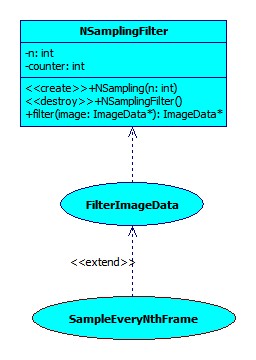
### FaceDetect Filter Process Specification



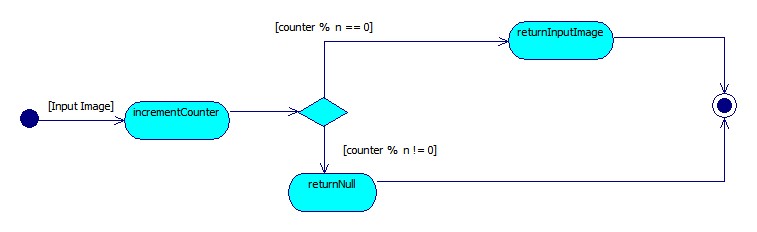
## NSamplingFilter

The NSamplingFilter is responsible for sampling every Nth frame captured.

### Functional Use Case



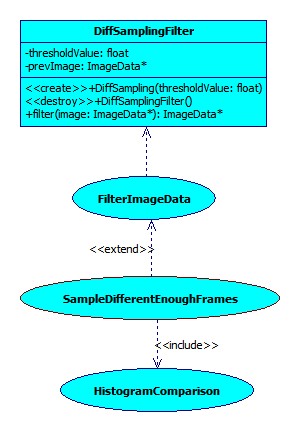
### Process Specification



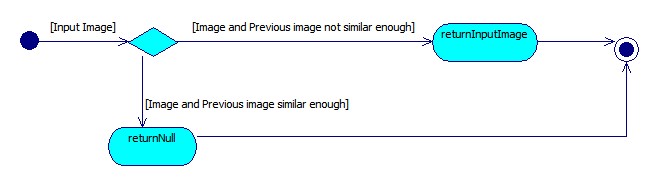
## DiffSamplingFilter

The DiffSamplingFilter is responsible for sampling a frame if it is different enough compared to the previously sampled frame.

### Functional Use Case



### Process Specification

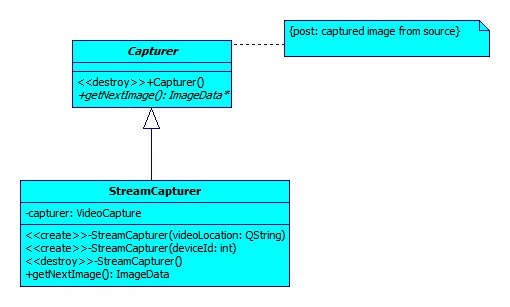


## Capturer

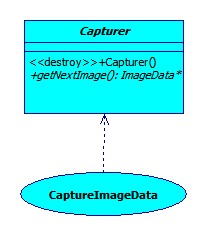
The Capturer will be used to get frames from a camera. It includes two subclasses:

* StreamCapturer: Connects to a video stream using a connection string to specify the location of the stream.
* DirectoryCapturer: Gets images stored in a directory.

### Capturer Service Contract



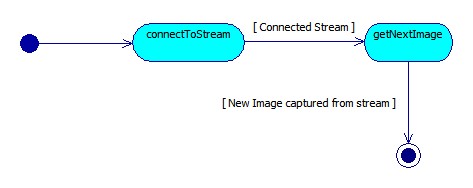
### Capturer Functional Use Case



### StreamCapturer Functional Use Case

### G:\University\2014\COS 301\Main Project\Documents\UML Pictures\StreamCapturerUC.jpg

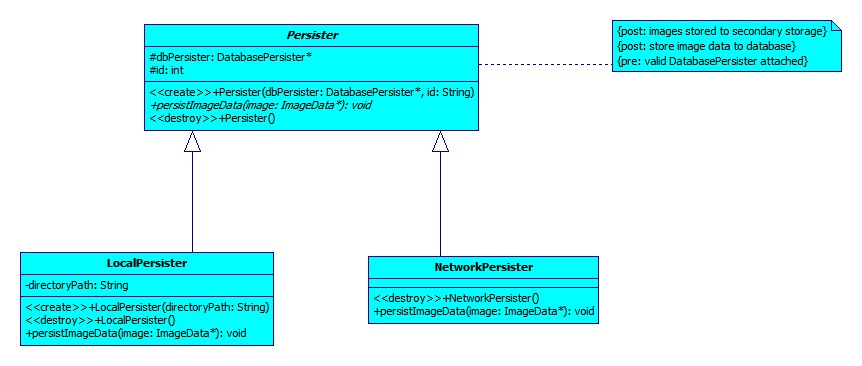
### StreamCapturer Process Specification



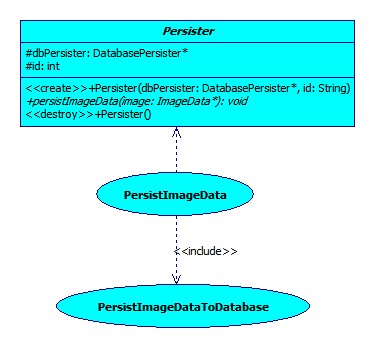
## Persister

A Persister is responsible for storing the image and its relative data to secondary storage.

### Service Contract



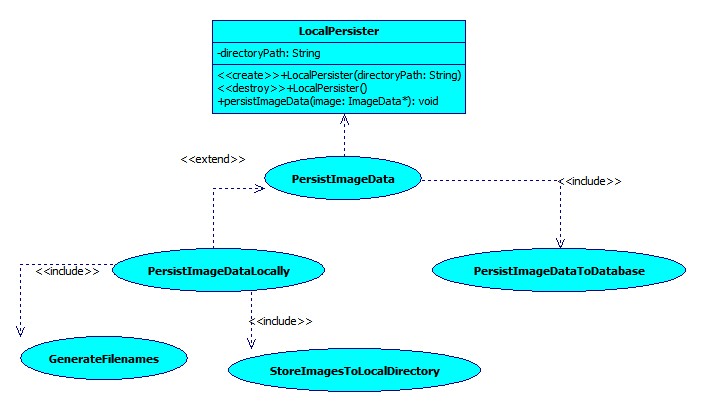
### Functional Use Case



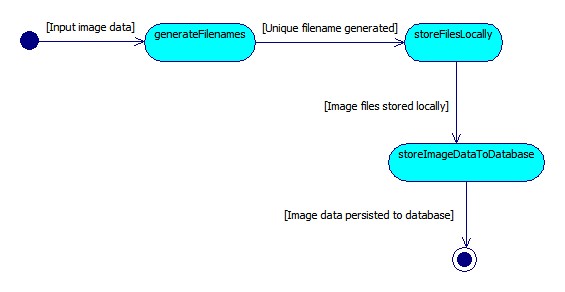
## LocalPersister

The LocalPersister is responsible for persisting the images locally (on the same machine).

### Functional Use Case



### LocalPersister Process Specification



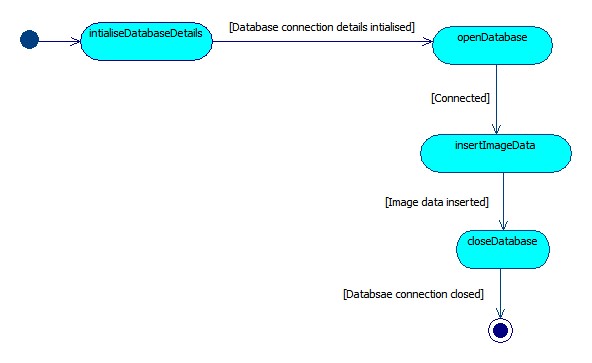
## DatabasePersister

The DatabasePersister will be used to store the image data such as filename and timestamp to the database.

### Functional Use Case

## G:\University\2014\COS 301\Main Project\Documents\UML Pictures\DatabasePersisterUC.jpg

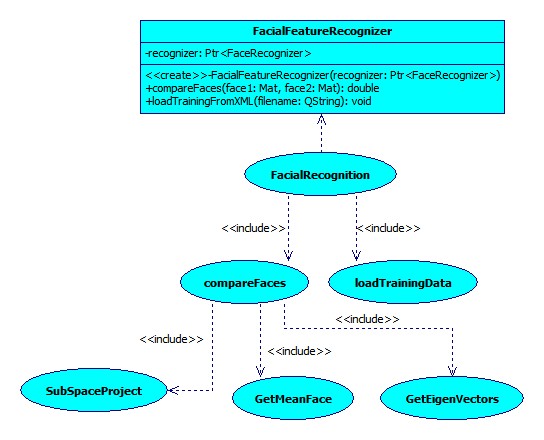
### Database Persister Process Specification



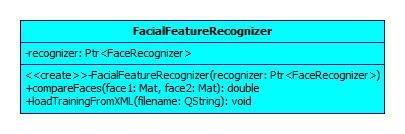
## FacialFeatureRecognizer

The FacialFeatureRecognizer is responsible for comparing the similarity between two faces.

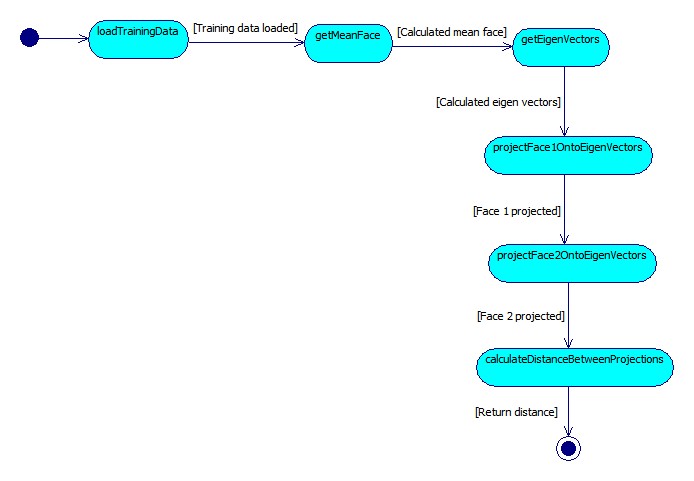
### Functional Use Case



### Design



### Process Specification



# Global Design

