Software Design Document for Thermal Face Recognition System

Ahmed Essam, Basil Essam, Mohammed Amer

June 12, 2017

1 Introduction

1.1 Purpose

This software design document describes the architecture and system design of thermal face recognition system and how it will be developed based on its functionalities, use cased and architecture diagram.

1.2 Scope

Develop a system that will be able to recognize human faces based on their thermal images after making a comparative study between the algorithms that have been used in related systems and testing them on our database to come up with the best algorithm to be used. The application can be specialized in making universities control rooms more secured by having a thermal face recognition system that allows desired people to enter the rooms based on their authorities.

1.3 Overview

The document is composed of sections describing the software design components that are needed in order to implement this system, these sections are;

A. Purpose of the document: This section describes the purpose of this software design document and how it will be developed.

B. Scope of the project: This section describes the scope of the system and to what extendibles will it be able to work properly.

C. System Overview: This section describes the system's overview and how it is going to work based on its components and its materials.

D. System Architecture: This section describes the architecture design of this project that explains the relationships between the modules to achieve the

complete functionality of the system.

E. Data Design: This section describes the database design of the system and how they are connected and related to each others.

F. Component Design: This section describes each component and how it works giving a summary of the algorithms used with graphs and pseudo-codes.

G. Human Interface Design: This section describes the interface design of the system that the user is going to be interacting with in order to deal with the system giving images of it.

H. Requirements Matrix: This section provides a reference that traces the components and structure to the requirements in SRS document.

2 System Overview

2.1 User Enrollment

Enrolling the users into the system in order to give the system the ability to recognize them and be able to allow them to enter the desired rooms or not.

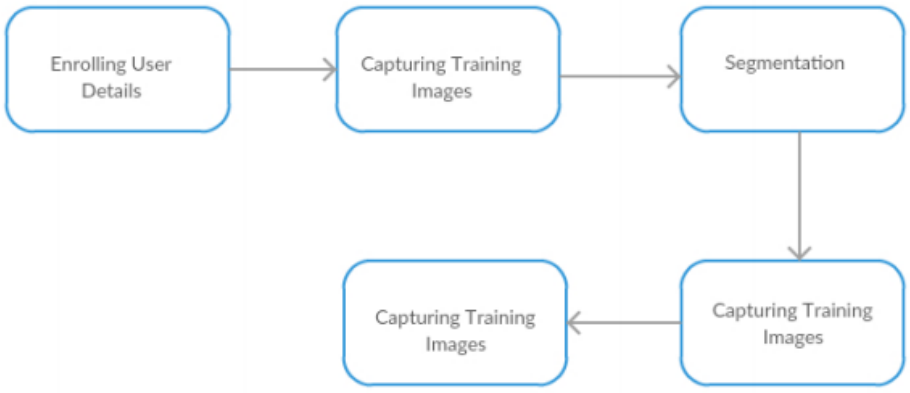


Figure 1: System Overview of User Enrollment

2.2 User Recognition

The system's overall process in order to identify the users' faces and taking an action based on their authorities.

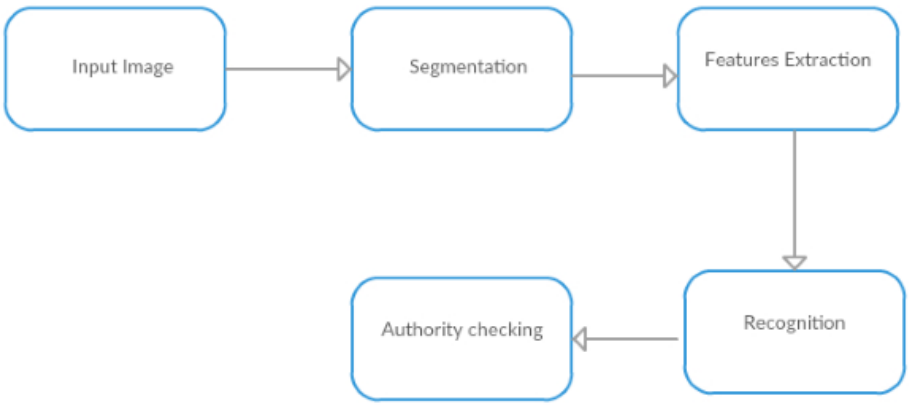


Figure 2: System Overview of User Recognition

3 System Architecture

3.1 Architectural Design

The system's architecture diagram used layered approach as each layer works based on the provided services it receives from the layer before it. Data layer is the lowest layer of an application. It is responsible of communicating with the used data storage like the thermal images, extracted features and the details of the users. Business layer includes the core business functionality of the application, which includes two components; the user enrollment stage, in which the admin/s of the system fill/s the user details, authorities and capture/s training images if the desired users in order to use it later on. The other stage is the system processing stage or the real application stage, in which the images pass by three main steps; image segmentation, features extraction and classification. Presentation layer contains components for users to interact with the application. It is responsible of processing user's input and returning the correct response back to the user which can be the user enrollment form that the admin/s fill or the form where they can login to the system.

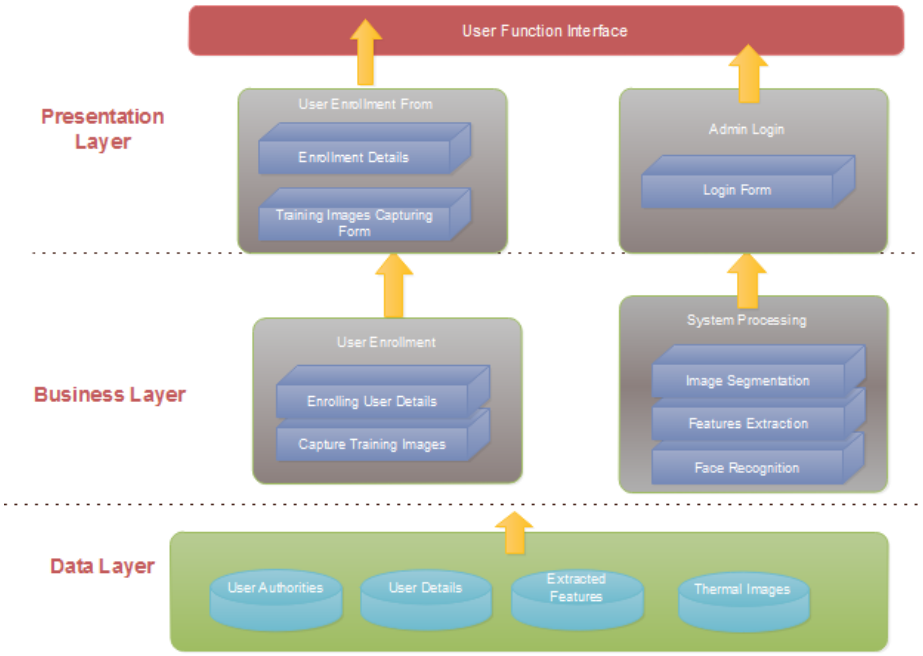


Figure 3: Architecture Diagram of the system

3.2 Decomposition Description

3.2.1 Admin Login

First sequence diagram begins as the admin enters his credentials to be checked by the system.

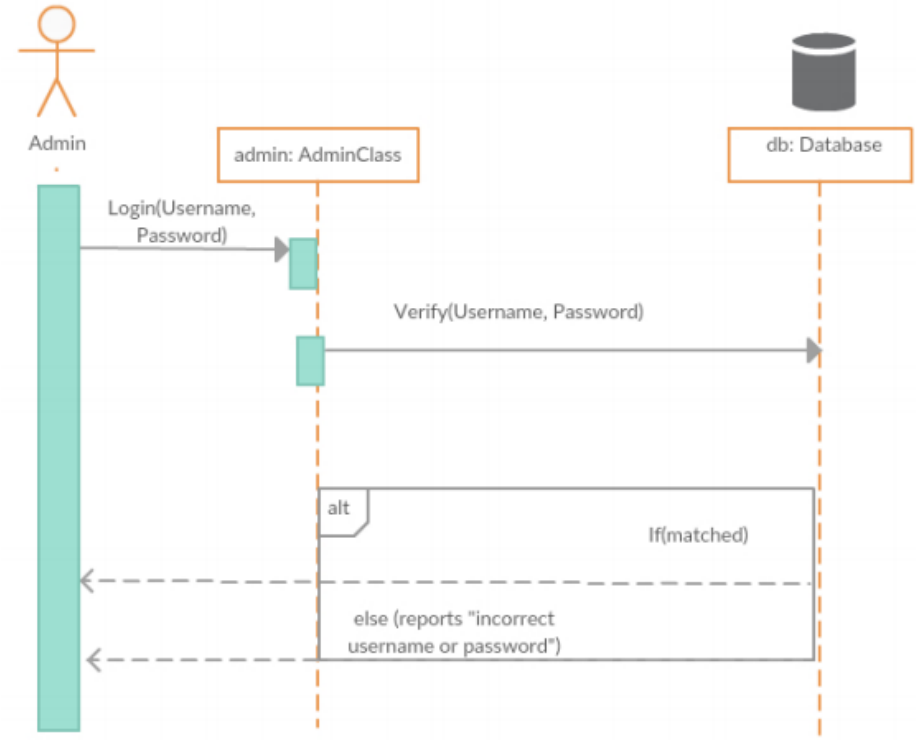


Figure 4: Admin Login Sequence Diagram

3.2.2 Enrolling user details

This sequence diagram describes the sequence as the admin starts enrolling the user details into the system. First the admin creates an object from the user class.

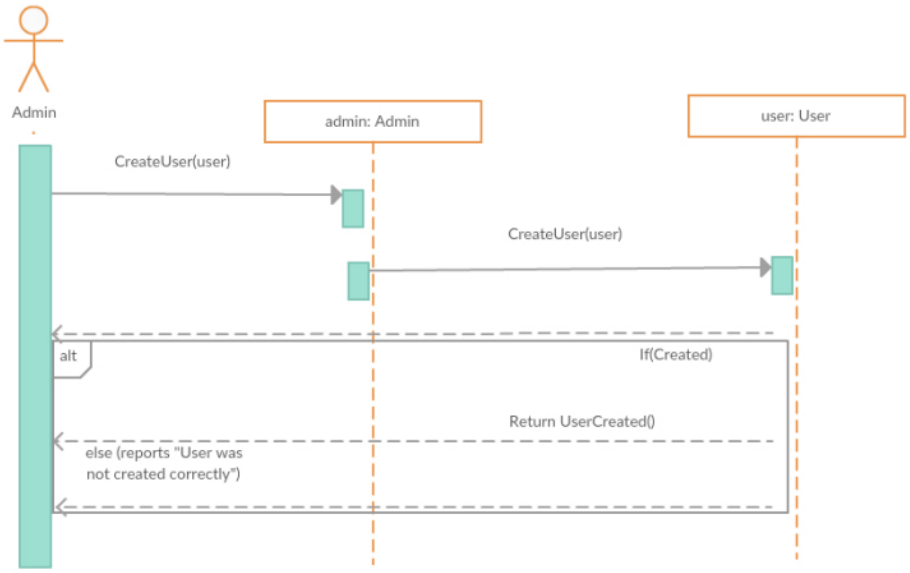


Figure 4: User Enrollment Sequence Diagram

Then the admin fills the user's details to be stored to his profile in the database.

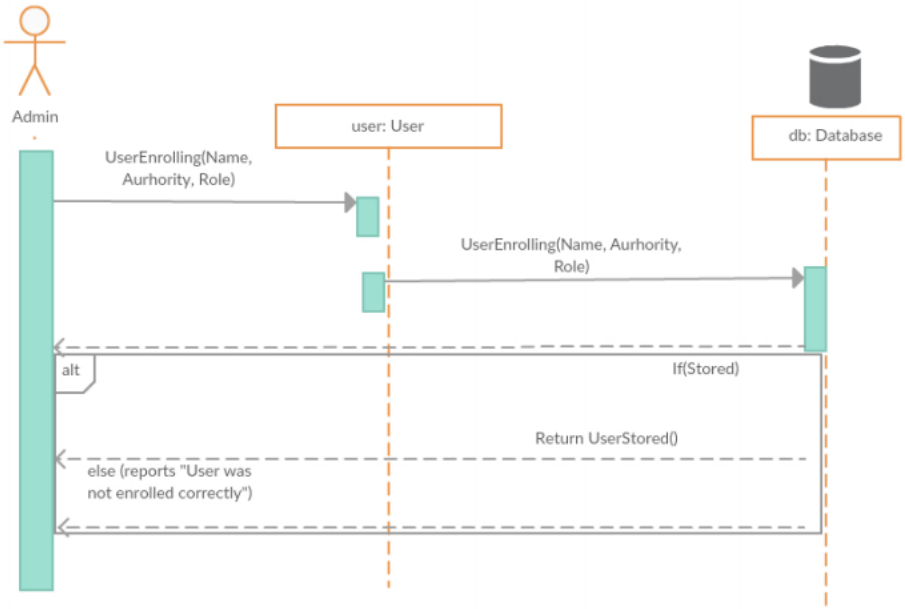


Figure 5: Storing User's details Sequence Diagram

3.2.3 Capturing training images

This sequence diagram describes as the admin stores the captured images of the users in the database.

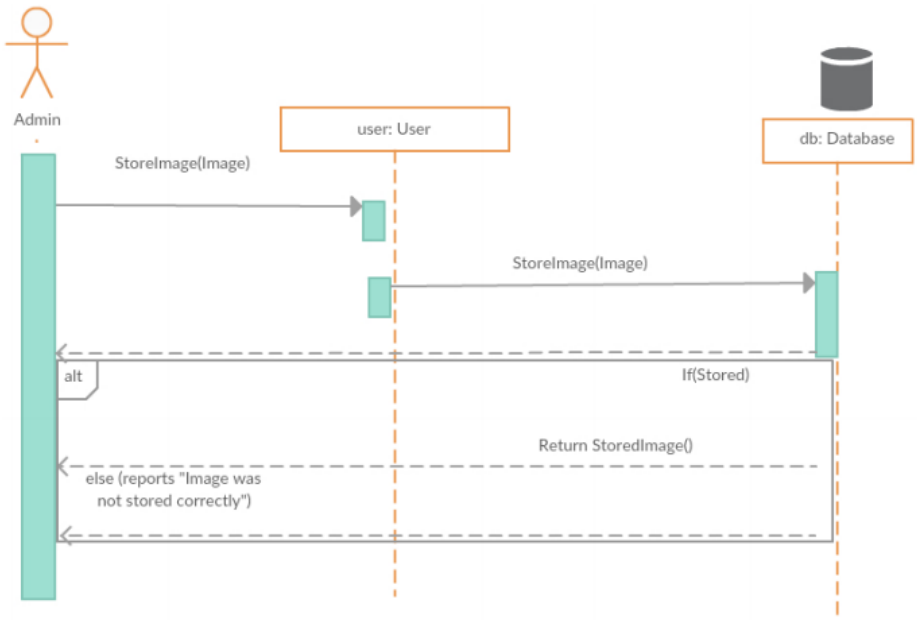


Figure 6: Capturing Training Images Sequence Diagram

3.2.4 Segmentation in the training stage

In this sequence diagram, the admin chooses to segment the training images after capturing it.

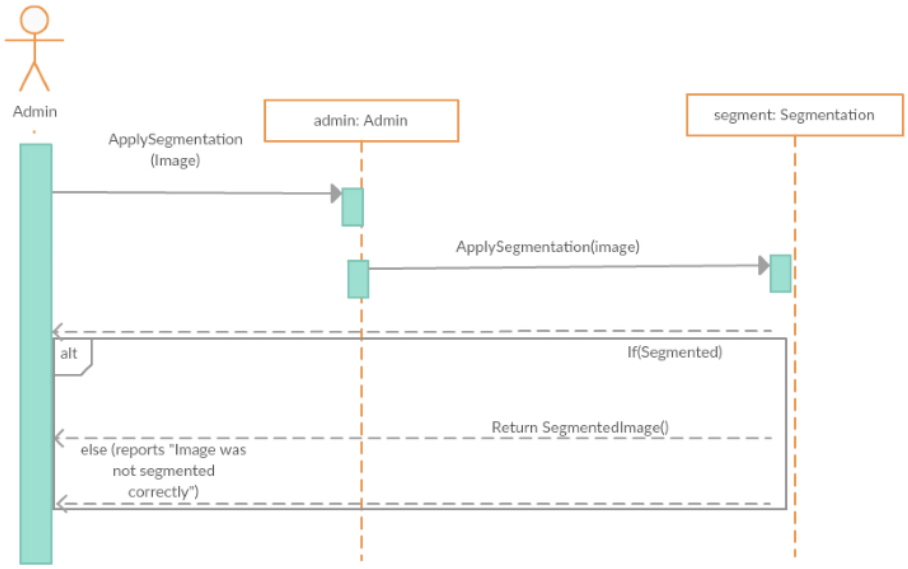


Figure 7: Training Images Segmentation Sequence Diagram

3.2.5 Features extraction in the training stage

In this sequence diagram, the admin chooses to extract features from the segmented images.

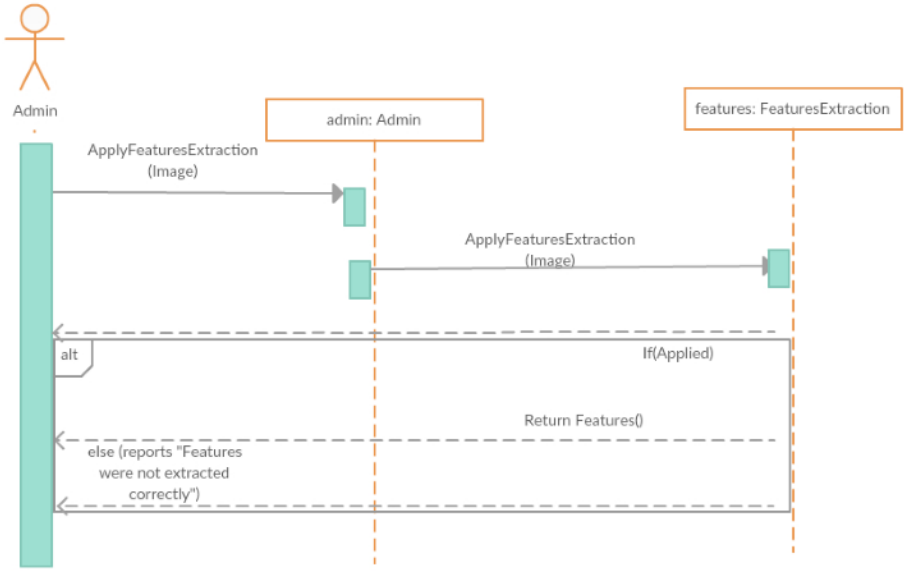


Figure 8: Training Images Features Extraction Sequence Diagram

Then in this sequence, the admin stores the extracted features in the database.

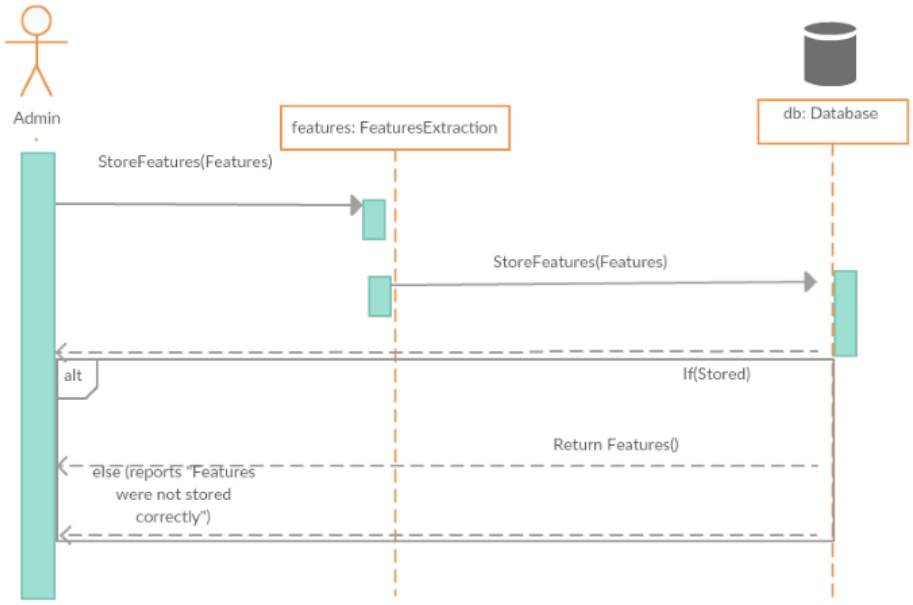


Figure 9: Storing Training Images Features Sequence Diagram

3.2.6 Face detecting in testing stage

This sequence diagram shows when the system detects a face in order to begin the recognition process.

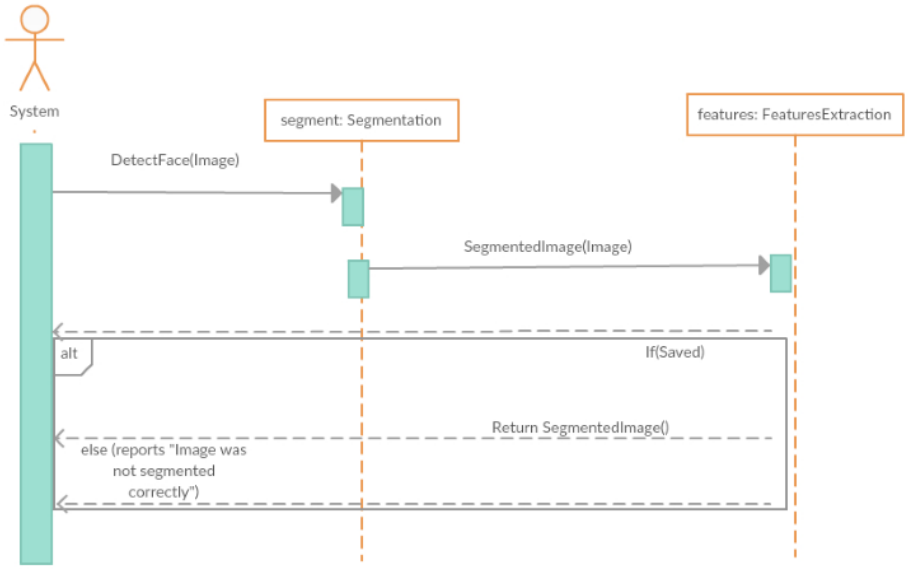


Figure 10: Face Detecting Sequence Diagram

3.2.7 Image segmentation in the testing stage

In this sequence, the system begins the segmentation stage after detecting the face area.

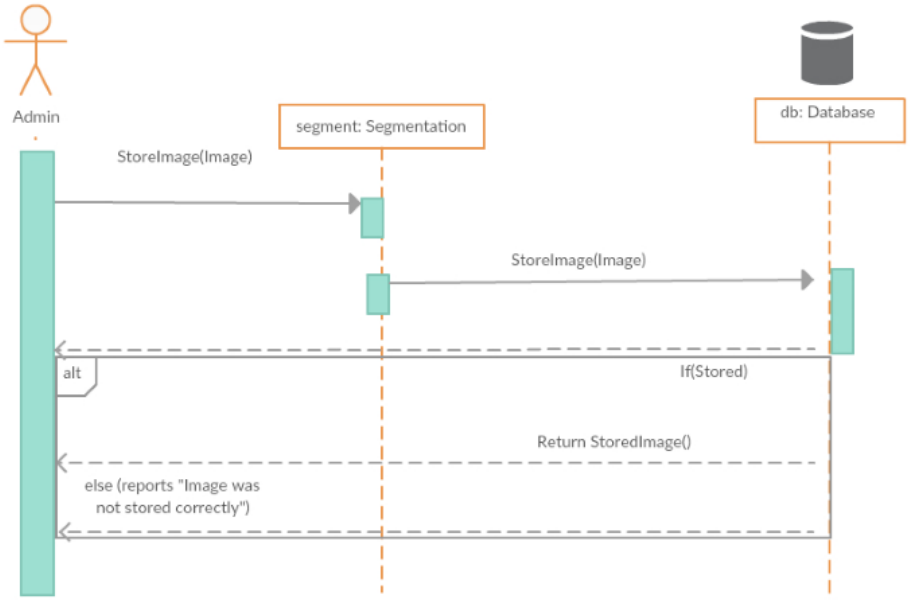


Figure 11: Testing Images Segmentation Sequence Diagram

3.2.8 Features Extraction in the testing stage

This sequence diagram shows the sequence when the system extracts features from the segmented face.

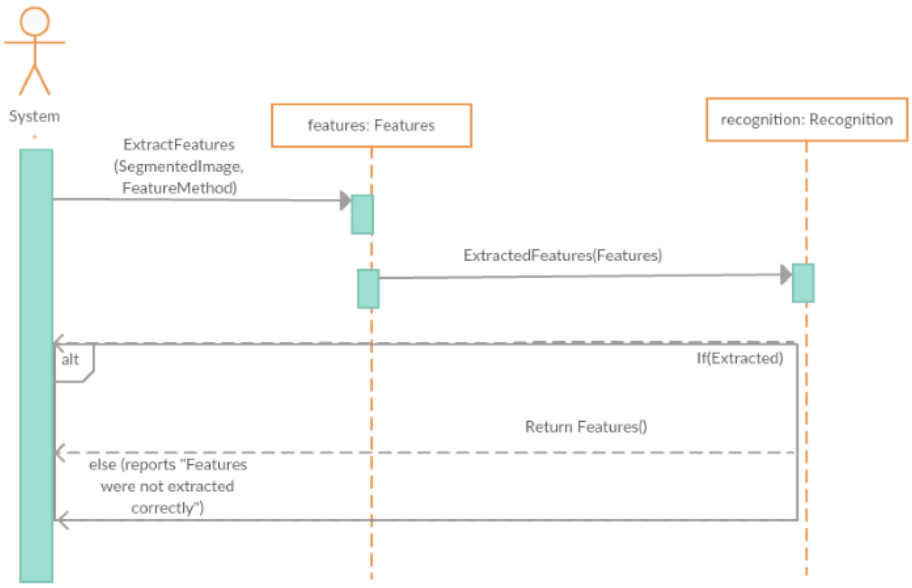


Figure 12: Testing Images Features Extraction Sequence Diagram

3.2.9 Face Recognition in the testing stage

This sequence diagram shows when the system tries to recognize the face after comparing the extracted features with the already stored ones.

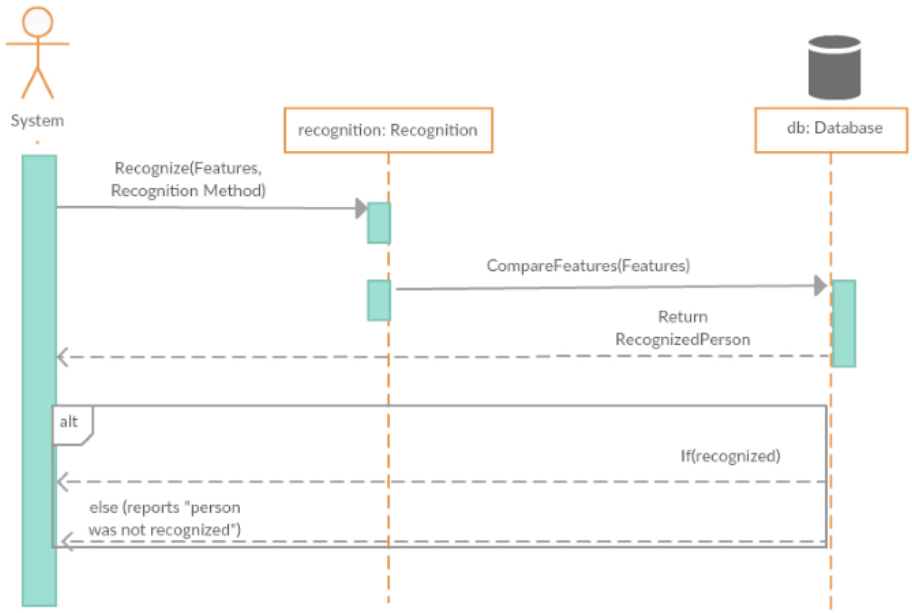


Figure 13: Face Recognition Sequence Diagram

3.3 Design Rationale

Using design patterns in constructing the structure of the system makes communication between designers more efficient. It provides a way to solve issues based on software development and also makes the overall system easier to understand and maintain. Using strategy design patterns specifically was a decision based on the runtime of the system, it enables an algorithm behavior to be selected at one time, and that is exactly what we are trying to achieve, using different features extraction algorithms and classification algorithms in order to have a comparative study between these various algorithms in order to come up with the best combination between them to use it later on.

4 Data Design

4.1 Data Description

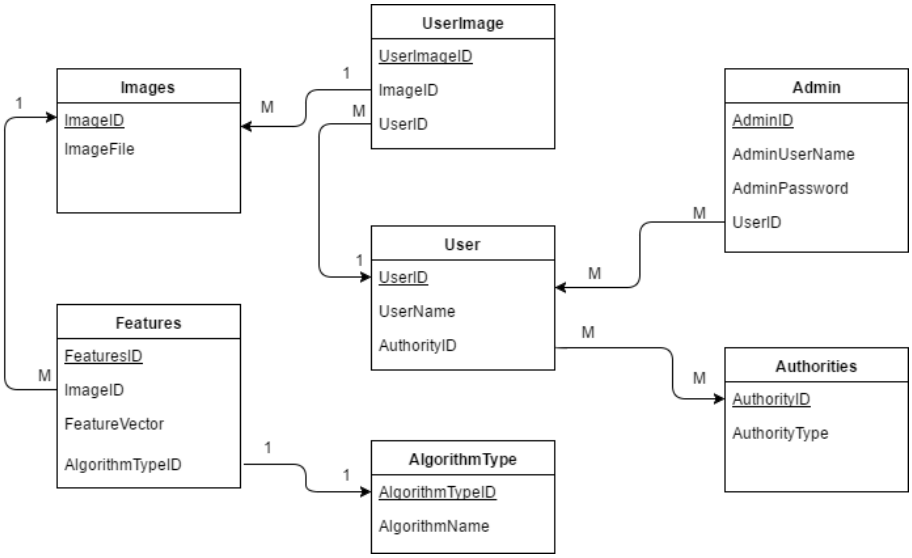


Figure 14: Database Tables of the system

The system's database is composed of seven tables which we believe will be enough to achieve the system's requirements and benefits. We have an Admin class, which will be responsible for the system's management and it has an object of the User class. The User class has an object from the Authority class, in order to check it later on in the recognition stage. The UserImage contains an object from the User class and the Images class, it contains the user's details and his images. Features table has an object from the Images class to be stored by its features and also has an object from AlgorithmType table to determines which algorithm are we using in extracting the features.

4.2 Data Dictionary

Class Name	Function	Type
Images	Images()	void
	setImage()	void

Figure 15: Images Class

Class Name	Function	Type
Admin	Admin()	void
	setUserName()	void
	setPassword()	void

Figure 16: Admin Class

Class Name	Function	Type
Authority	SetAuthority()	void
	GetAuthotiy()	string

Figure 17: Authority Class

Class Name	Function	Type
SIFT	ExtractFeatures()	float
	Compute()	mat
	Detect()	mat

Figure 18: SIFT Class

Class Name	Function	Type
HOG	ExtractFeatures()	float
	Compute()	mat
	Detect()	mat

Figure 19: HOG Class

Class Name	Function	Type
SVM	SetType()	void
	SVMTrain()	void
	SetKernel()	void
	Predict()	float

Figure 20: SVM Class

Class Name	Function	Type
NeuralNetworks	SetLayerSize()	void
	SetNumberNeurons()	void
	SetActivationFunction()	void
	Train()	void

Figure 21: Neural Networks Class

5 Component Design

A. Class Diagram

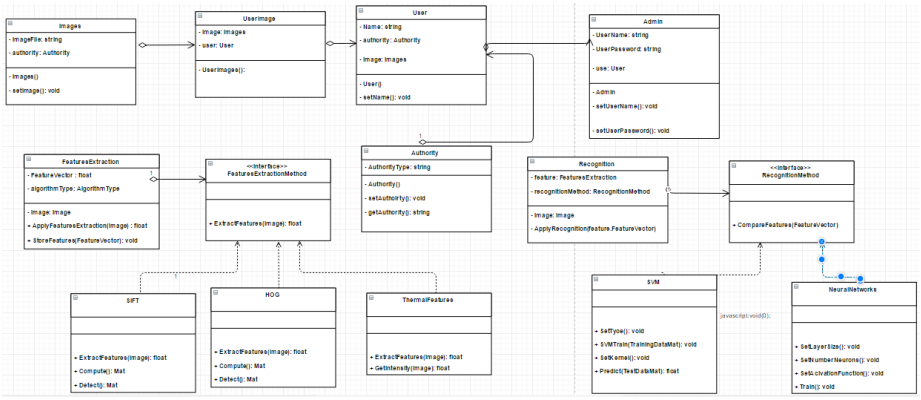


Figure 22: System's Class Diagram

B. Segmentation: In computer vision, image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels).The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze [1].The simplest method of image segmentation is called the thresholding method. This method is based on a threshold value to turn a gray-scale image into a binary image. In our case, this threshold value is calculated based on the average of the images' pixels, if the pixel is greater than the threshold value; it is assigned to the maximum grey value, else; it is assigned to the minimum grey value. After completing this, now the thermal image is converted into a binary one (black and white), and the thermal face is distinguished from the rest of the image having the large white area. These white pixels (the face region) are then restored from the original image's pixels to get the original face from the thermal image not the binarized one in order to process on the face area or region only. After restoring the face's pixels, there are a lot of black pixels still remained in the image, in order to remove these pixels; the white pixels are stored in a vector, discarding the black ones. Then the image is returned with the area of interest which is the white pixels.

If g(x, y) is a thresholded version of f(x, y) at some global threshold T,

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) \geq T \\ 0 & \text{otherwise} \end{cases}$$

Simple threshold function pseudo code
if src(x,y) ¿ thresh
dst(x,y) = maxValue
else
dst(x,y) = 0

C. Feature Extraction : Facial feature extraction plays an important step in automated visual interpretation and human face recognition. Detecting facial feature is a crucial role in a wide variety of application such as human computer interface, facial animation and face recognition, etc. Once you have detected a face (and possibly chosen an aligner for it), you need to extract a feature which you can then use for recognition or similarity comparison In our project we used 4 algorithms and compared between them to get the best accuracy (Hog, Gabor, Thermal Features and Sift)

1) Hog: The histogram of oriented gradients is a feature descriptor used in computer vision and image processing for the purpose of increasing the recognition. The technique counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, but differs in that it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy.

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

where * here denotes the 2-dimensional signal processing convolution operation.

$$\mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}}$$

The x-coordinate is defined here as increasing in the "right"-direction, and the y-coordinate is defined as increasing in the "down"-direction.

Output He[M, N]: histogram equalized image

```
Calculate histogram P for image I
T[0] = P[0]
For k = 1 to L
        T[k] = T[k-1] + P[k]
End for
For r  = 0 to L
        S[r] = round(T[r]*L)
End for
For y = 0 to M
        For x = 0 to N
                r = I[y, x]
                He[y, x] = S[r]
        End for
End for
```

Input I[M, N]: image, L: Maximum gray level Output He[M, N]: histogram equalized image

2) Gabor: is a linear filter used for edge detection. Frequency and orientation representations of Gabor filters are similar to those of the human visual system, and they have been found to be particularly appropriate for texture representation and discrimination. In the spatial domain, a 2D Gabor filter is a Gaussian kernel function modulated by a sinusoidal plane wave.

Complex

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \exp\left(i\left(2\pi\frac{x'}{\lambda} + \psi\right)\right)$$

Real

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2\pi\frac{x'}{\lambda} + \psi\right)$$

Imaginary

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \sin\left(2\pi\frac{x'}{\lambda} + \psi\right)$$

where

$$x' = x\cos\theta + y\sin\theta$$

and

$$y' = -x\sin\theta + y\cos\theta$$

In this equation, represents the wavelength of the sinusoidal factor, represents the orientation of the normal to the parallel stripes of a Gabor function, is the phase offset, is the sigma/standard deviation of the Gaussian envelope and is the spatial aspect ratio, and specifies the ellipticity of the support of the Gabor function.

$$G_c[i, j] = Be^{-\frac{(i^2+j^2)}{2\sigma^2}} \cos(2\pi f(i\cos\theta + j\sin\theta))$$

$$G_s[i, j] = Ce^{-\frac{(i^2+j^2)}{2\sigma^2}} \sin(2\pi f(i\cos\theta + j\sin\theta))$$

where B and C are normalizing factors to be determined. 2-D Gabor filters have rich applications in image processing, especially in feature extraction for texture analysis and segmentation. f defines the frequency being looked for in the texture. By varying , we can look for texture oriented in a particular direction. By varying , we change the support of the basis or the size of the image region being analyzed.

3) Thermal Features: It use the thermal pixels of the image as the features. High resolution thermal imaging refers to the fine detail and clarity of a thermal

image. This means it contains a large number of pixels per unit of area. More pixels mean greater temperature measurement accuracy, particularly for small objects.

4) Sift : Scale-invariant feature transform (SIFT) is an algorithm in computer vision to detect and describe local features in images. Applications include object recognition, robotic mapping and navigation, image stitching, 3D modeling, gesture recognition, video tracking, individual identification of wildlife and match moving.

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma),$$

where $L(x, y, k\sigma)$ is the convolution of the original image $I(x, y)$ with the Gaussian blur $G(x, y, k\sigma)$ at scale $k\sigma$, i.e.

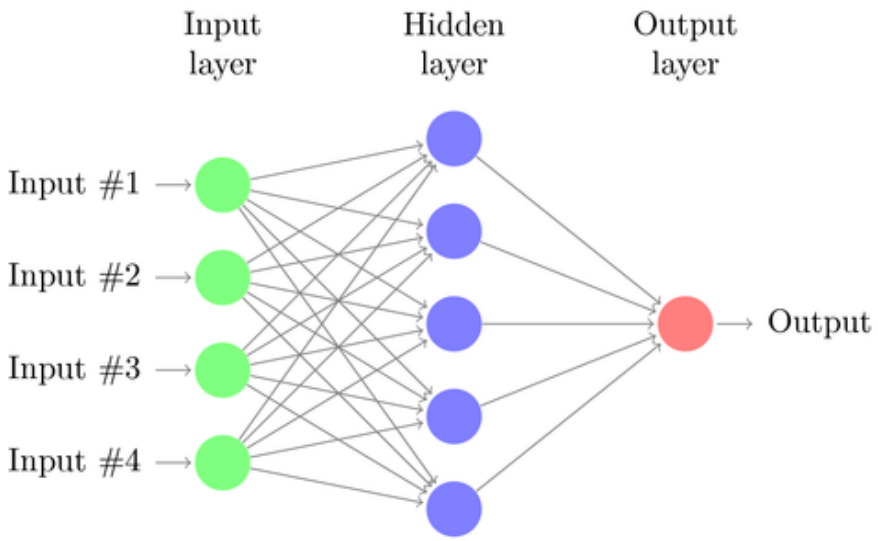
$$L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y)$$

Problem	Technique	Advantage
key localization / scale / rotation	DoG / scale-space pyramid / orientation assignment	accuracy, stability, scale & rotational invariance
geometric distortion	blurring / resampling of local image orientation planes	affine invariance
indexing and matching	nearest neighbor / Best Bin First search	Efficiency / speed
Cluster identification	Hough Transform voting	reliable pose models
Model verification / outlier detection	Linear least squares	better error tolerance with fewer matches
Hypothesis acceptance	Bayesian Probability analysis	reliability

Figure 23: Table Description for the techniques and their advantages

5) Kaze : A novel 2D feature detection and description method that operates completely in a nonlinear scale space. Previous methods such as SIFT or SURF find features in the Gaussian scale space (particular instance of linear diffusion).

D. Classification: Classification is dividing certain data into categories or classes, to be used in it's most effective way, which helps in retrieving certain data from the whole dataset in accurate way. One of the strongest Classifiers and the one we used is Neural Network. A Neural Network is consisted of 3 main layers : 1) Input Layer - 2) Hidden Layer and 3) Output layer Input Layer takes the input data , send it to the hidden layer which here the calculation and the network starts learning , then the output layer which gives the class related to the input. So here is how Neural network works, by using the concept of hidden layers



but Neural network has different types , we are using Multilayer perceptron supervised Neural Network.
Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output. Given a set of features X=x1,x2,..,xm and a target y, it can learn a non-linear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers The leftmost layer, known as the input layer, consists of a set of neurons representing the input features. Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation, followed by a non-linear activation function R - like the hyperbolic tan function. The output layer receives the values from the last hidden layer and transforms them into output values.
We teach the network through back propagation.Learning occurs in the perceptron by changing connection weights after each piece of data is processed, based on the amount of error in the output compared to the expected result. This is an example of supervised learning, and is carried out through backpropagation,

a generalization of the least mean squares algorithm in the linear perceptron, and as the network is a feedforward artificial neural network. This means the signal inside the neural network flows from input layer passing hidden layers to output layer. While training the error correction of neural weights are done in the opposite direction. This is done by the backpropagation algorithm. At first a cumulative input is calculated by the following equation:

$$s = \sum_{k=1}^{n} i_k \cdot w_k$$

Considering the BIAS value the equation is:

$$s = (\sum_{k=1}^{n} i_k \cdot w_k) + BIAS \cdot w_k$$

Sigmoid activation function:

$$o = \mathrm{sig(s)} = \frac{1}{1 + \mathrm{e}^{-\mathrm{s}}}$$

If the neural network is initialized by random weights it has of course not the expected output. Therefore training is necessary. While supervised training known inputs and their corresponded output values are presented to the network. So it is possible to compare the real output with the desired output. The error is described as the following algorithm:

$$E = \frac{1}{2} \sum_{i=1}^{n} (t_i - o_i)^2$$

The learning algorithm of a single layer perceptron is easy compared to a multilayer perceptron. The reason is that just the output layer is directly connected to the output, but not the hidden layers. Therefore the calculation of the right weights of the hidden layers is difficult mathematically. To get the right delta value for changing the weights of hidden neuron is described in the following equation:

$$\Delta w_{ij} = -\alpha \cdot \frac{\partial E}{\partial w_{ij}} = \alpha \cdot \delta_j \cdot x_i$$

The interpretation of output values just makes sense for the output layer. The interpretation is depending on the use of the neural network. If the network is used for classification, so binary output is used. Binary has two states: True or false. The network will produce always linear output values. Therefore these values has to be converted to binary values:

$$o < 0.5 : False$$

If using linear output the output values have to be normalized to a real value the network is trained for:

$$f = o \cdot (f_{max} - f_{min}) + f_{min}$$

The same normalization equation for input values is used for output values while training the network:

$$o = \frac{f - f_{min}}{f_{max} - f_{min}}$$

After all that we create a Confusion Matrix,A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.We can easily calculate the accuracy, that is the ratio of correctly predicted samples, by simply summing the diagonal of our confusion matrix (number of correct predictions) and diving by the sum of our cells of our confusion matrix (number of test samples) as follows:

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

Figure 24: Confusion Matrix Table

Let's now define the most basic terms, which are whole numbers (not rates):

- **True Positives (TP):**These are cases in which we predicted yes and they are yes.

- **True Negatives (TN):**We predicted no,and they are no.
- **False Positives (FP):**We predicted yes, and they aren't.
- **False Negatives (FN):**We predicted no ,and they are yes.
  After reading the test data , it has it's own Confusion matrix, then we compare between them getting the accuracy by incrementing the prediction based on getting true for every yes and no we predicted in its own place.

6 Humnan Interface Design

6.1 Overview of User Interface

The user interface of the system is composed of two main stages; training stage and testing stage. In training stage, the admin has the ability to login into the systems by entering his credentials. Then, the admin will have the ability to enroll a new user into the system by filling his details, capturing training images of him and enrolling their authorities. The admin will have the ability to segment, extract features from the training images and saving the images into the database.

6.2 Screen Images

Here are Screen Shoots for our application. The admin will be asked to enter his credentials in order to login so he can add a new profile.

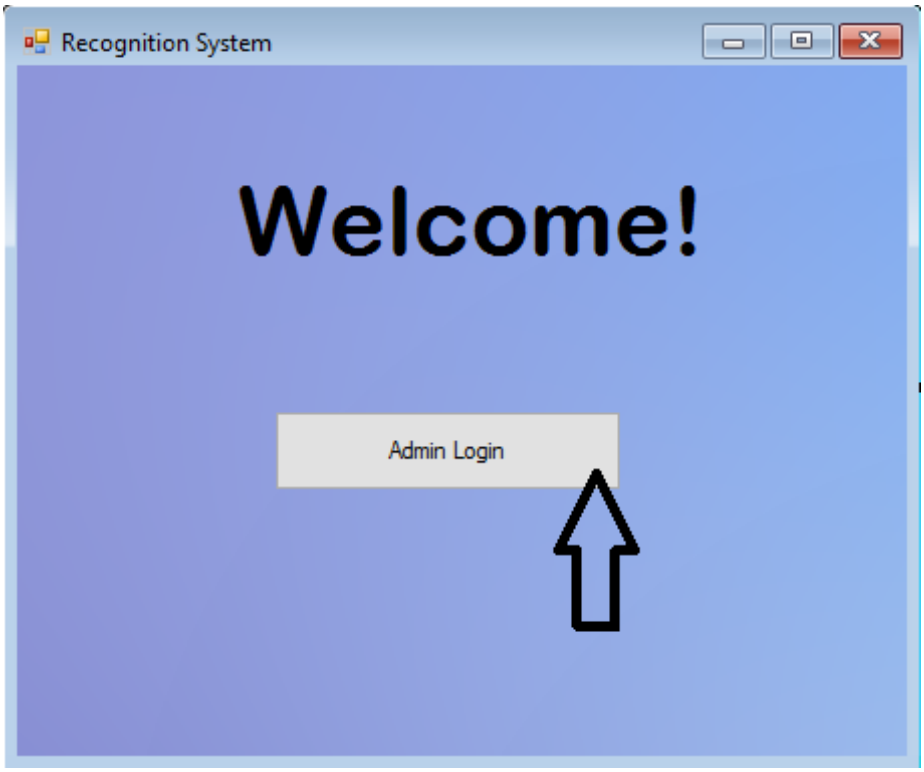


Figure 25: Homepage Screen

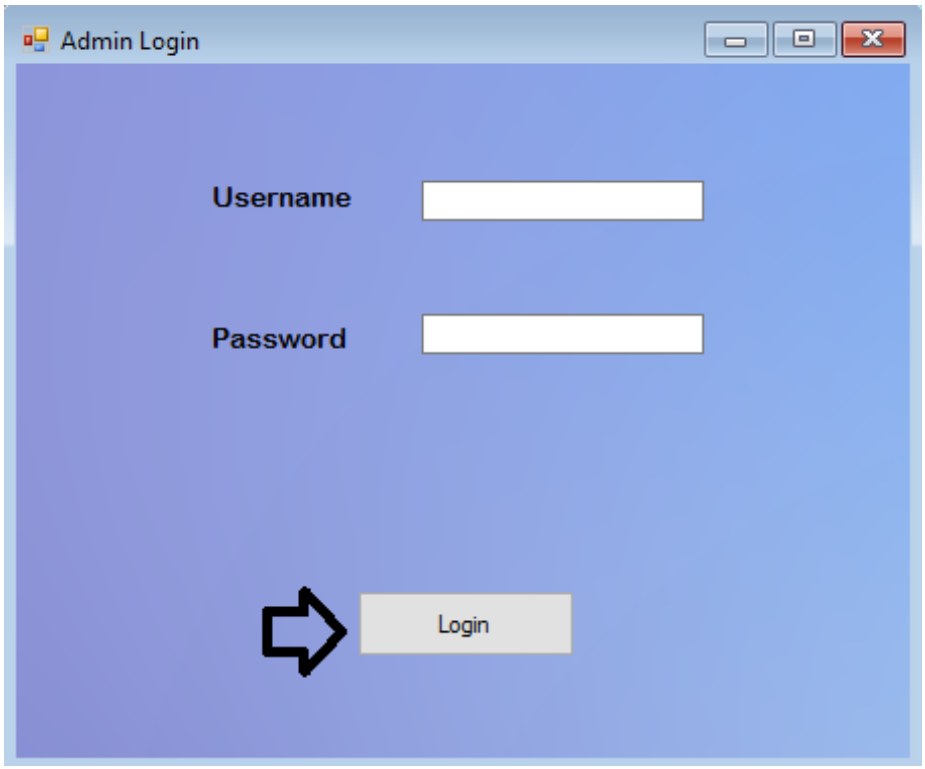


Figure 26: Admin Login Screen

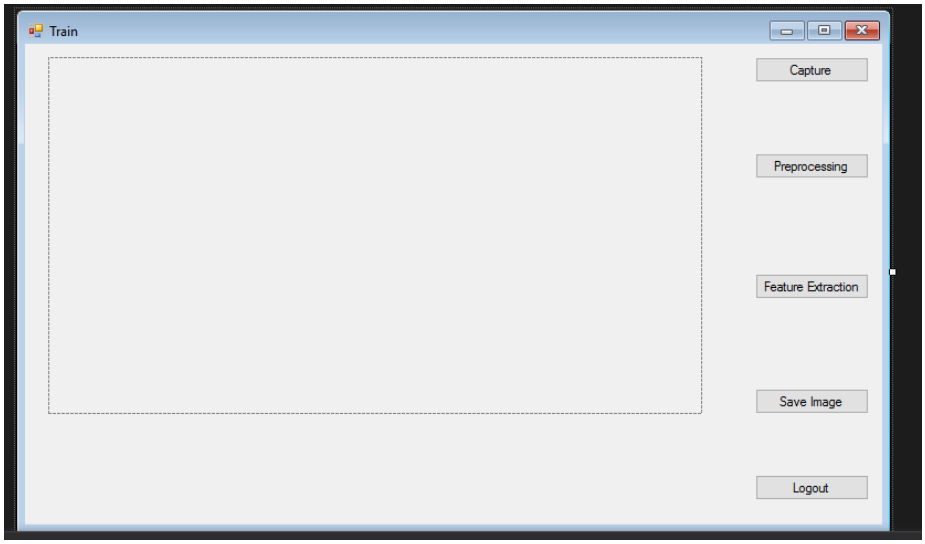


Figure 27: Training Images Capturing Screen

Figure 28: Training Images Screen

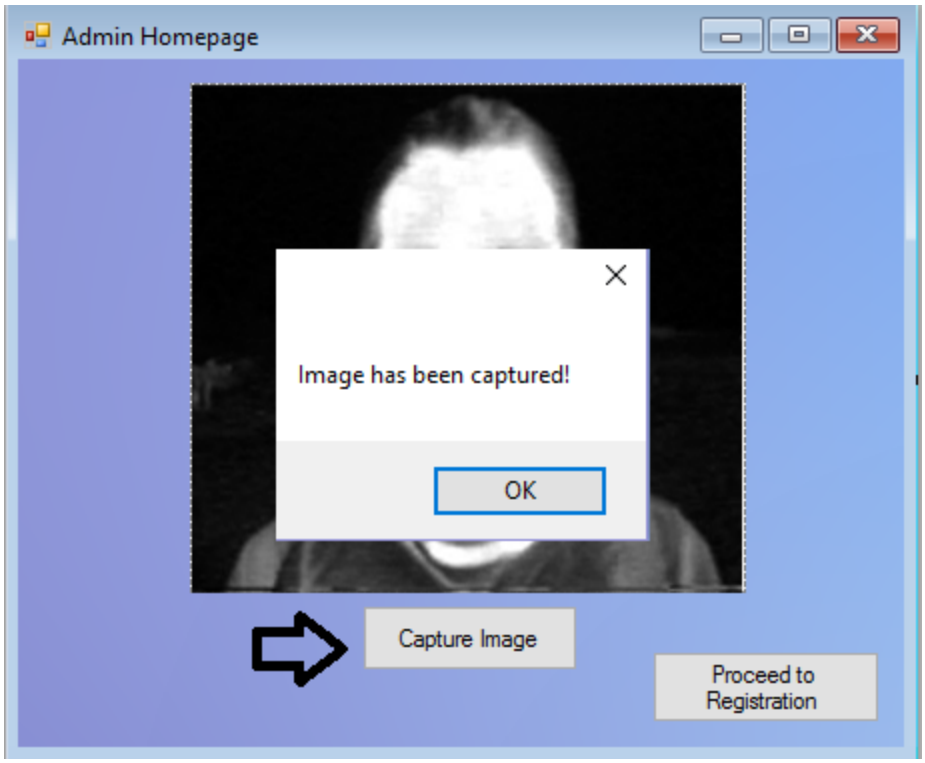


Figure 29: After Capturing Images Screen

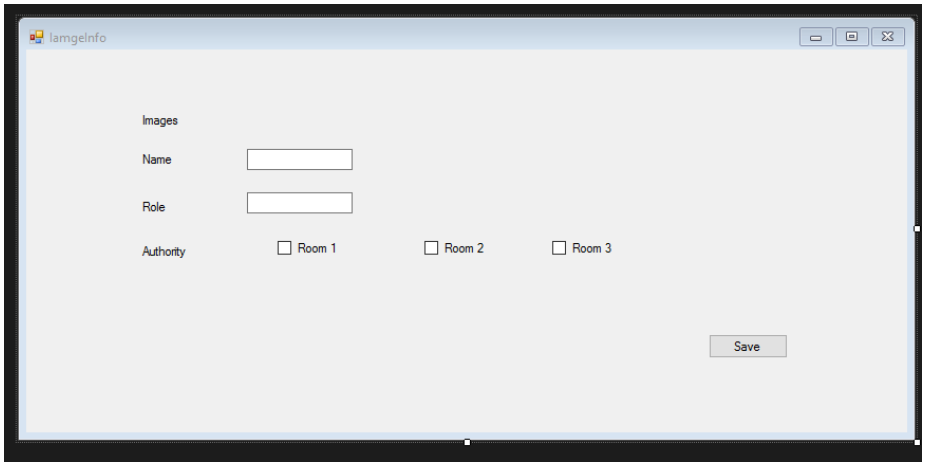


Figure 30: Enrolling User Screen

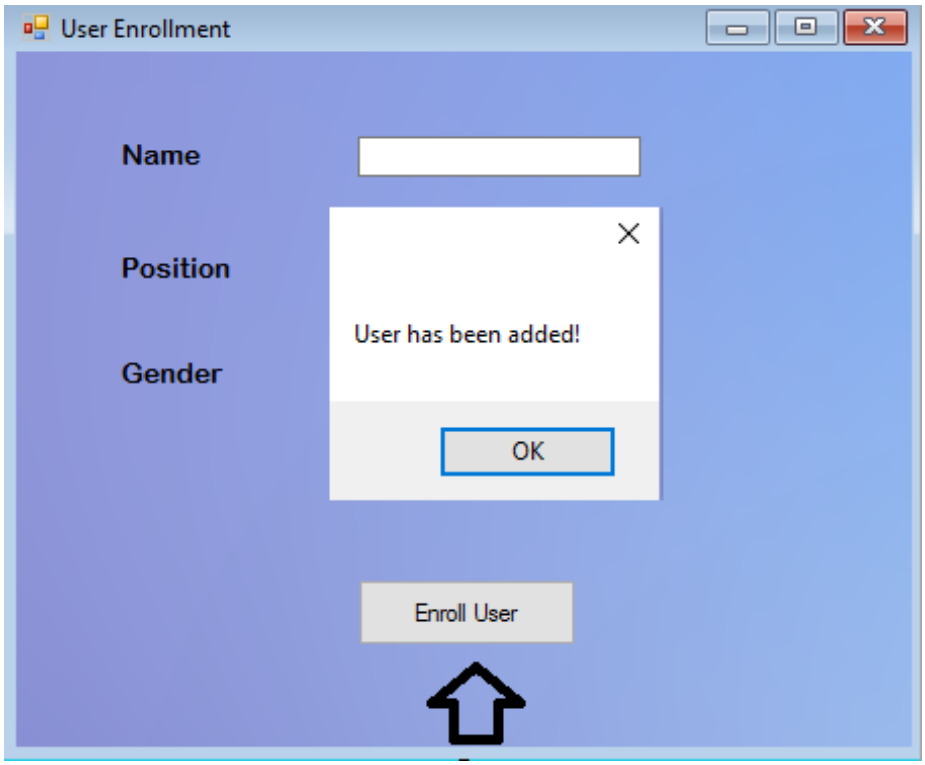


Figure 31: Informing Message that the user has been added

6.3 Screen Objects and Actions

First the admin has to enter his username and password then he will be able to access the capturing page, where he can add or capture a new image for a new user and use the preprocessing and the feature extraction algorithms and then save the image, next he will add the information of the user and the rooms he can access,Finally the data will be saved in the database.

7 Primitive Results

Algorithm	Number of training images	Accuracy
SIFT	5	60%
	7	71.42%
	10	**80%**
Thermal Features	5	40%
	7	57%
	10	60%
HOG	5	60%
	7	57%
	10	70%
KAZE	5	
	7	
	10	

Figure 32: Primitive Results Table

We started training an testing the Neural Network with 3 sets , 50 ,70 and 90 images, taking 70% for training and 30% for testing. Using Neural network with 2 Feature extractors, we came up with those results.

- NN with HOG Descriptors;

  - 150 input neurons: 46%
  - 512 input neurons: 45%
  - 1250 input neurons: 53%

- NN with KAZE Descriptors;

  - 150 input neurons: 100%
  - 512 input neurons: 86%
  - 1250 input neurons: 53%

Features	Input network Size	Dataset		Accuracy
		Training (70%)	Testing (30%)	
HOG Descriptor	150	35	15	46%
		49	21	-
		63	27	-
	512	35	15	45%
		49	21	-
		63	27	-
	1250	35	15	53%
		49	21	-
		63	27	-
KAZE Descriptor and Detector	150	35	15	100%
		49	21	-
		63	27	-
	512	35	15	86%
		49	21	-
		63	27	-
	1250	35	15	73%
		49	21	-
		63	27	-

Figure 33:Primitive Results Table

8 Requirements Matrix

Functional Requirements	Status	SRS	SDD	Technical Specification
Admin Login	In progress	_	_	
User Enrollment	In progress	_	_	
Capture Training Images	Completed	x	x	
Segmenting Training Images	Completed	x	x	Adaptive threshold
Features Extraction from training images	Completed	x	x	Extract feature vectors
Uploading testing images	Completed	x	x	
Segmenting Testing Images	Completed	x	x	Adaptive threshold
Features Extraction from testing images	Completed	x	x	Extract feature vectors
Recognizing testing images	Completed	_	x	Classifying based on feature vectores

Figure 34: Requirements Matrix

9 References

[1] Linda G. Shapiro and George C. Stockman (2001): Computer Vision, pp 279-325, New Jersey, Prentice-Hall. [2] Gaber, Tarek, et al. "Human Thermal Face Recognition Based on Random Linear Oracle (RLO) Ensembles." Intelligent Networking and Collaborative Systems (INCOS), 2015 International Conference on. IEEE, 2015.

[2] C. Ding and D. Tao, A comprehensive survey on poseinvariant face recognition, arXiv preprint arXiv:1502.04383, 2015.

[3] Wu, Zhan, Min Peng, and Tong Chen. "Thermal face recognition using convolutional neural network." Optoelectronics and Image Processing (ICOIP), 2016 International Conference on. IEEE, 2016.

[4] OTCBVS ThermalnVisible Face Database: http://www.cse.ohiostate.edu/OTCBVS-BENCHibench.html

[5] Equinox Infrared Face Database¿ http://www.equinoxsensors.comlproductsIHID.html