



# REQUIREMENT DOCUMENT

*Attendance system using facial  
recognition*

By Reina JAAFAR and Ali BAYDOUN





## TABLE OF CONTENTS

---

Purpose of Document.....	3
1. Features and functionalities .....	4
1- Normal User.....	4
2- Admin .....	5
2. Usability .....	6
3. Scenarios .....	7
1- Normal User.....	7
2- Admin User .....	10
4. Sequence diagrams.....	15
1- Check In / Check out .....	15
2- Check Log History .....	16
3- Check information .....	17
4- Change Password .....	18
5- Check Dashboard .....	19
6- Admin Add employee.....	20
7- Admin Update employee.....	21
5. Face Recognition model .....	22
6. The model ... in details .....	27
7. Data Base Schema.....	32
8. API Documentation.....	33
1-Login API .....	33
2 -Compare faces API.....	34
3 -Add Employee API.....	35
4 -Change Password API .....	36
5 -Check for spoofing API .....	37
6 -Get Employee by Email API.....	38
7 -Get Employee Information API.....	39
8 -Get Employee Logs API.....	40
9 -Get Employee Role API .....	41
10 -Get Employees API.....	42
11 -Get Face embedding API.....	43
12 -Get Filter Logs API.....	44
13 -Get Role API .....	45
14 -Register API.....	46
15 -Update Employee API.....	47
9. Wireframes .....	48
1- Employee Pages.....	48
2- Admin pages.....	52

---

# PURPOSE OF DOCUMENT

---

This is a Requirements Specification document for a new attendance system using facial recognition (ASFR). ASFR allows companies to track employees' attendances using facial recognition instead of the usual fingerprint method. The goal of this applications is to minimize physical contact since after covid everything tactile wasn't recommended.

This document describes the scope, objectives, and the functional requirements of the system we are building.

## ***Who are we targeting?***

Our main target is companies that want to integrate an attendance system or replace the fingerprint attendance system.

# 1. FEATURES AND FUNCTIONALITIES

---

## I- Normal User

- a. Checking In: Allows the user to check in using the facial recognition system.
- b. Checking Out: Allows the user to check out using the facial recognition system.
- c. Logging in to website: the user logs on the website using his credentials.
- d. Check personal information: the user checks his information such as email, first name, last name and if his is active on My information page.
- e. Check Dashboard: the user will be able to check their attendance details.
- f. Check Logs: user checks his log history on My logs page with filtering options by year, status, or month.
- g. Change password: user change his password by putting his old password and verifying new password twice on Change Password page.
- h. Log out: when user clicks on Log Out in the navigation side bar he is successfully logged out.

## 2- Admin

- a. Checking In: Allows the user to check in using the facial recognition system.
- b. Checking Out: Allows the user to check out using the facial recognition system.
- c. Logging in to website: the user logs on the website using his credentials.
- d. Check personal information: the user checks his information such as email, first name, last name and if his is active on My information page.
- e. Check Logs: user checks his log history on My logs page with filtering options by year, status, or month.
- f. Change password: user change his password by putting his old password and verifying new password twice on Change Password page.
- g. Check Dashboard: the user will be able to check their attendance details.
- h. Check employees: user checks list of employees.
- i. Update employee's information: the admin should be able to modify any employee's information.
- j. Check personal log history: the admin should be able to see his log history, what dates he checked in and out and at what time.
- k. Check all employees log history: the admin should be able to see all the employees log details in addition to his personal log details.
- l. Add new employees: the admin should be able to add new employees to the database, with all their information, including the vectorized value of the employee's face image.
- m. Log out: when user clicks on Log Out in the navigation side bar he is successfully logged out.

## 2. USABILITY

---

There are 2 main usabilities for this application: Admin and Employees:

- a. Employees who will use the system to check in and out of work, and to check their personal log history, personal information, and attendance details.
- b. The Admin who will be able to add, update an employee to the database, and check all employee's log history, in addition to the normal employee's usability.

## 3. SCENARIOS

---

The following scenario describes how the employee will check in using the attendance system. Furthermore, the actors presented in the following scenario are as presented below:

- The user: the employee checking in / checking out.
- The face recognition attendance system.

### I- Normal User

#### 1.1 Normal Scenario

##### A- Checking-In

- 1- User positions face in front of the system's camera.
- 2- Facial recognition technology captures the user's facial features.
- 3- The web interface calls an API that contains the face recognition model, sending the frame containing the user's face.
- 4- The attendance system searches for the employee in the database by comparing the employee's vectorized face value with the saved vectorized face value in the database of the record found previously.
- 5- The system displays a green "Verified" indicator and shows the user's name.
- 6- The system logs the current timestamp into the attendance database.
- 7- The attendance system updates the company's database with the user's check-in time.
- 8- Check-in process is complete.

##### B- Checking out

- 1- User positions face in front of the system's camera.
- 2- Facial recognition technology captures the user's facial features.
- 3- The web interface calls an API that contains the face recognition model, sending the frame containing the user's face.
- 4- The attendance system searches for the employee in the database by comparing the employee's vectorized face value with the saved vectorized face value in the database of the record found previously.
- 5- The system displays a green "Verified" indicator and shows the user's name.

- 6- The system logs the current timestamp into the attendance database.
- 7- The attendance system updates the company's database with the user's check-out time.
- 8- Check-out process is complete.

### **C- Checking personal log history**

- 1- The employee enters his email and password.
- 2- The website checks in the database the validity of the credentials entered.
- 3- Credentials are validated, the website checks in the database the role of the user logging in.
- 4- The role of the user is verified as employee, the user is redirected to the employee index page.
- 5- The user clicks on the “My Logs” page, the website redirects the user to that page.
- 6- The website sends a query that returns the logs of the user logged in.
- 7- The user can filter the data received depending on a certain period.

### **D- Checking my information**

- 1- The employee enters his email and password.
- 2- The website checks in the database the validity of the credentials entered.
- 3- Credentials are validated, the website checks in the database the role of the user logging in.
- 4- The role of the user is verified as employee, the user is redirected to the employee index page.
- 5- The user clicks on the “My Information” page, the website redirects the user to that page.
- 6- The website sends a query that returns my information page of the user logged in.

### **E- Change password**

- 1- The employee enters his email and password.
- 2- The website checks in the database the validity of the credentials entered.
- 3- Credentials are validated, the website checks in the database the role of the user logging in.
- 4- The role of the user is verified as employee, the user is redirected to the employee index page.
- 5- The user clicks on the “Change password” page, the website redirects the user to that page.
- 6- The website sends a query that returns change password page of the user logged in.
- 7- The user writes his old password.
- 8- The user writes his new password two times.
- 9- The user clicks on the submit button.
- 10- The new password is saved in the database using a hashing system.



#### **F- Check dashboard**

- 1- The employee enters his email and password.
- 2- The website checks in the database the validity of the credentials entered.
- 3- Credentials are validated, the website checks in the database the role of the user logging in.
- 4- The role of the user is verified as employee, the user is redirected to the employee index page.
- 5- The user clicks on the “Dashboard” page, the website redirects the user to that page.
- 6- The website sends a query that returns the logs of the user logged in.
- 7- The user can filter the data received depending on a certain period.

### **1.2 Alternative Scenario Employee**

#### **A- Checking-In**

- 2.a- The employee’s face couldn’t be recognized by the attendance system.

#### **B- Checking out**

- 2.a- The employee’s face couldn’t be recognized by the attendance system.

#### **C- Checking personal log history**

- 1.a- The credentials couldn’t be validated, returns to log in page.

#### **D- Checking my information**

- 1.a- The credentials couldn’t be validated, returns to log in page.

#### **E- Change password**

- 1.a- The credentials couldn’t be validated, returns to log in page.
- 7.a- The user’s old password does not match his actual password.
- 8.9.a- The two new passwords do not match.

#### **F- Check Dashboard**

- 1.a- The credentials couldn’t be validated, returns to log in page.

## 2- Admin User

### 2.1 Normal Scenario Admin

#### A- Checking-In

- 1- User positions face in front of the system's camera.
- 2- Facial recognition technology captures the user's facial features.
- 3- The web interface calls an API that contains the face recognition model, sending the frame containing the user's face.
- 4- The attendance system searches for the employee in the database by comparing the employee's vectorized face value with the saved vectorized face value in the database of the record found previously.
- 5- The system displays a green "Verified" indicator and shows the user's name.
- 6- The system logs the current timestamp into the attendance database.
- 7- The attendance system updates the company's database with the user's check-in time.
- 8- Check-in process is complete.

#### B- Checking out

- 1- User positions face in front of the system's camera.
- 2- Facial recognition technology captures the user's facial features.
- 3- The web interface calls an API that contains the face recognition model, sending the frame containing the user's face.
- 4- The attendance system searches for the employee in the database by comparing the employee's vectorized face value with the saved vectorized face value in the database of the record found previously.
- 5- The system displays a green "Verified" indicator and shows the user's name.
- 6- The system logs the current timestamp into the attendance database.
- 7- The attendance system updates the company's database with the user's check-out time.
- 8- Check-out process is complete.

### **C- Checking my information**

- 1- The employee enters his email and password.
- 2- The website checks in the database the validity of the credentials entered.
- 3- Credentials are validated, the website checks in the database the role of the user logging in.
- 4- The role of the user is verified as employee, the user is redirected to the employee index page.
- 5- The user clicks on the “My Information” page, the website redirects the user to that page.
- 6- The website sends a query that returns my information page of the user logged in.

### **D- Change password**

- 1- The employee enters his email and password.
- 2- The website checks in the database the validity of the credentials entered.
- 3- Credentials are validated, the website checks in the database the role of the user logging in.
- 4- The role of the user is verified as employee, the user is redirected to the employee index page.
- 5- The user clicks on the “Change password” page, the website redirects the user to that page.
- 6- The website sends a query that returns change password page of the user logged in.
- 7- The user writes his old password.
- 8- The user writes his new password two times.
- 9- The user clicks on the submit button.
- 10- The new password is saved in the database using a hashing system.

### **E- Checking personal log history**

- 1- The employee enters his email and password.
- 2- The website checks in the database the validity of the credentials entered.
- 3- Credentials are validated, the website checks in the database the role of the user logging in.
- 4- The role of the user is verified as employee, the user is redirected to the employee index page.
- 5- The user clicks on the “Check my Logs” page, the website redirects the user to that page.
- 6- The website sends a query that returns the logs of the user logged in.
- 7- The user can filter the data received depending on a certain period.

## **F- Checking all employee**

- 1- The employee enters his email and password.
- 2- The website checks in the database the validity of the credentials entered.
- 3- Credentials are validated, the website checks in the database the role of the user logging in.
- 4- The role of the user is verified as employee, the user is redirected to the employee index page.
- 5- The user clicks on the “Employees” page, the website redirects the user to that page.
- 6- The website sends a query that returns the employees.

## **G- Checking employee’s log history**

- 7- The employee enters his email and password.
- 8- The website checks in the database the validity of the credentials entered.
- 9- Credentials are validated, the website checks in the database the role of the user logging in.
- 10- The role of the user is verified as employee, the user is redirected to the employee index page.
- 11- The user clicks on the “Logs” page, the website redirects the user to that page.
- 12- The website sends a query that returns the employees logs.
- 13- The user can filter the data received depending on a certain period.

## **H- Adding new employee to the database**

- 1- The user enters his email and password.
- 2- The website checks in the database the validity of the credential enters.
- 3- Credentials are validated, the website checks in the database the role of the user logging in.
- 4- The role of the user is verified as admin, the user is redirected to the admin index page.
- 5- The user clicks on the “Add new employee” page, the website redirects the user to that page.
- 6- The user fills a form containing the new employee’s information and clicks the “Add” button.
- 7- The website validates the information entered in the form and create a new record in the database.

## **I- Updating employee’s information**

- 1- The user enters his email and password.
- 2- The website checks in the database the validity of the credential enters.
- 3- Credentials are validated, the website checks in the database the role of the user logging in.
- 4- The role of the user is verified as admin, the user is redirected to the admin index page.
- 5- The user clicks on the “Update employee” button located in “Employees Page”, the website redirects the user to that page.

- 6- The user chooses which employee to update.
- 7- The website sends queries to the database to retrieve the employee's current information.
- 8- The employee was found, the database returns the employee's information to the website, which displays the information to the user.
- 9- The user updates the employee's information and clicks on the "Update" button.
- 10- The website validates the information entered by the user and updates the employees' record in the database.

#### **J- Check dashboard**

- 8- The employee enters his email and password.
- 9- The website checks in the database the validity of the credentials entered.
- 10- Credentials are validated, the website checks in the database the role of the user logging in.
- 11- The role of the user is verified as employee, the user is redirected to the employee index page.
- 12- The user clicks on the "Check Dashboard" page, the website redirects the user to that page.
- 13- The website sends a query that returns the logs of the user logged in.
- 14- The user can filter the data received depending on a certain period.

## **2.2 Alternative Scenario Admin**

### **A- Checking-In**

- 2.a- The employee's face couldn't be recognized by the attendance system.

### **B- Checking-out**

- 2.a- The employee's face couldn't be recognized by the attendance system.

### **C- Checking personal log history**

- 3.a- The credentials couldn't be validated returns to log in page.

### **D- Checking employee's log history**

- 3.a- The credentials couldn't be validated returns to log in page.

### **E- Checking all employee**

- 3.a- The credentials couldn't be validated returns to log in page.



## **F- Adding new employee to the database**

- 3.a The credentials couldn't be validated returns to log in page.
- 7.a- The information entered was not valid, returns to step 6.
  - b- The user does fill all required fields, returns to step 6.

## **G- Updating employee's information**

- 3.a- The credentials couldn't be validated returns to log in page.
- 8.a- The employee couldn't be found in the database, returns to step 6.
- 10.a- The information entered were not valid, returns to step 9.
  - b- The user does fill all required fields, returns to step 9.

## **G- Change password**

- 1.a- The credentials couldn't be validated, returns to log in page.
- 7.a- The user's old password does not match his actual password.
- 8.9.a- The two new passwords do not match.

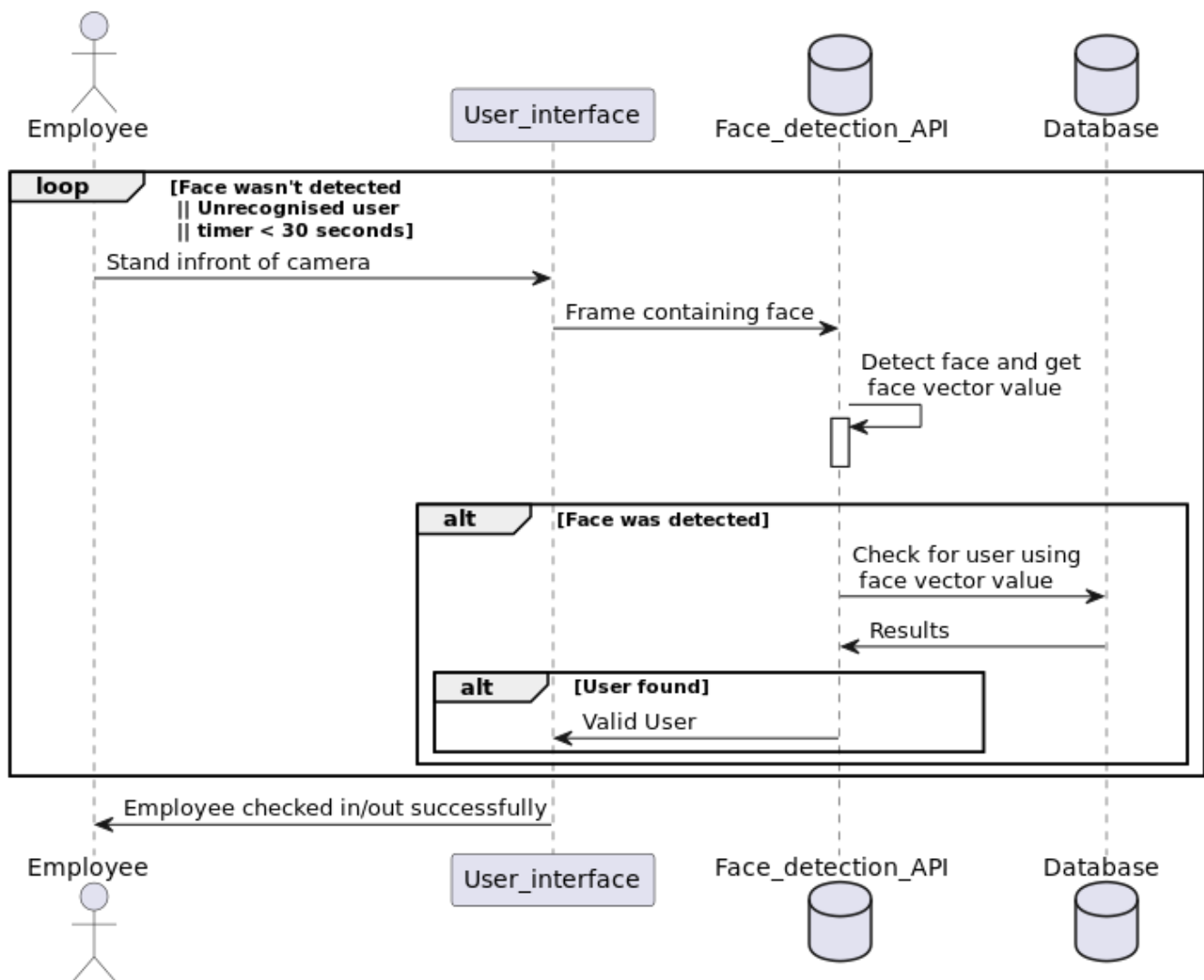
## **H- Check Dashboard**

- 1.a- The credentials couldn't be validated, returns to log in page.

## 4. SEQUENCE DIAGRAMS

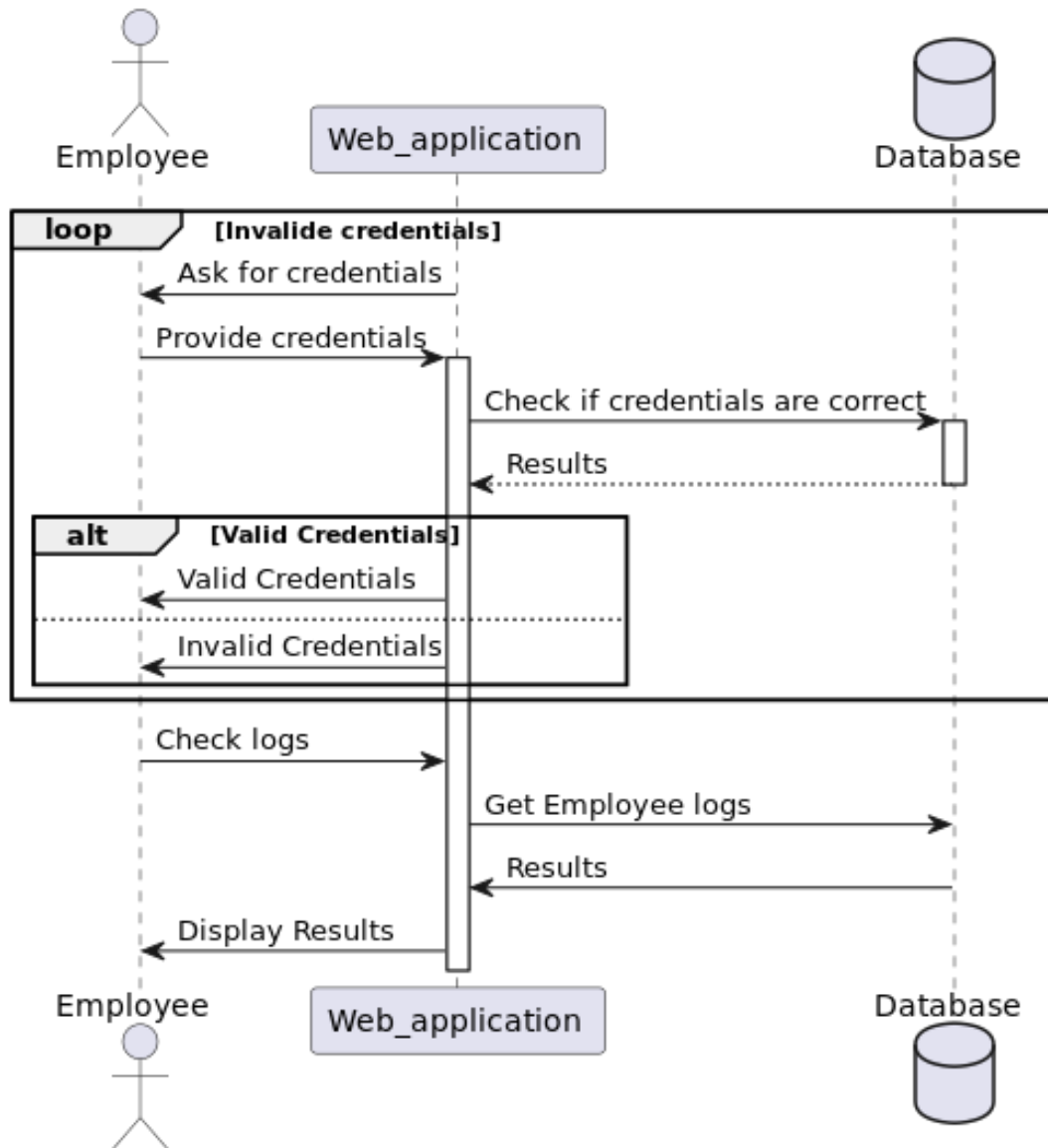
### I- Check In / Check out

The following diagram showcases the employee using the attendance system to check-in/check-out.



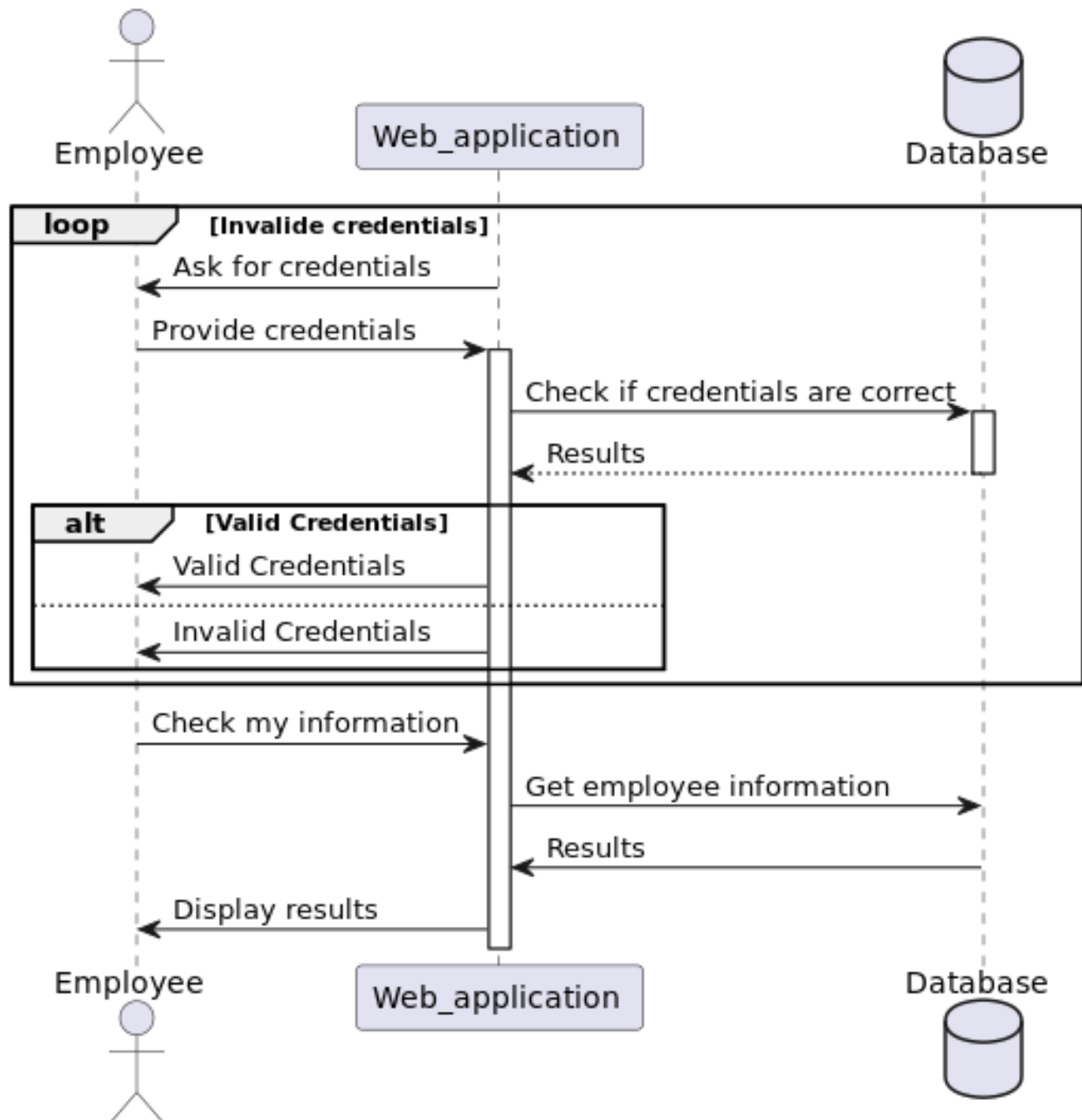
## 2- Check Log History

This diagram describes the interaction between the database, the user interface, and a user trying to check their log history.



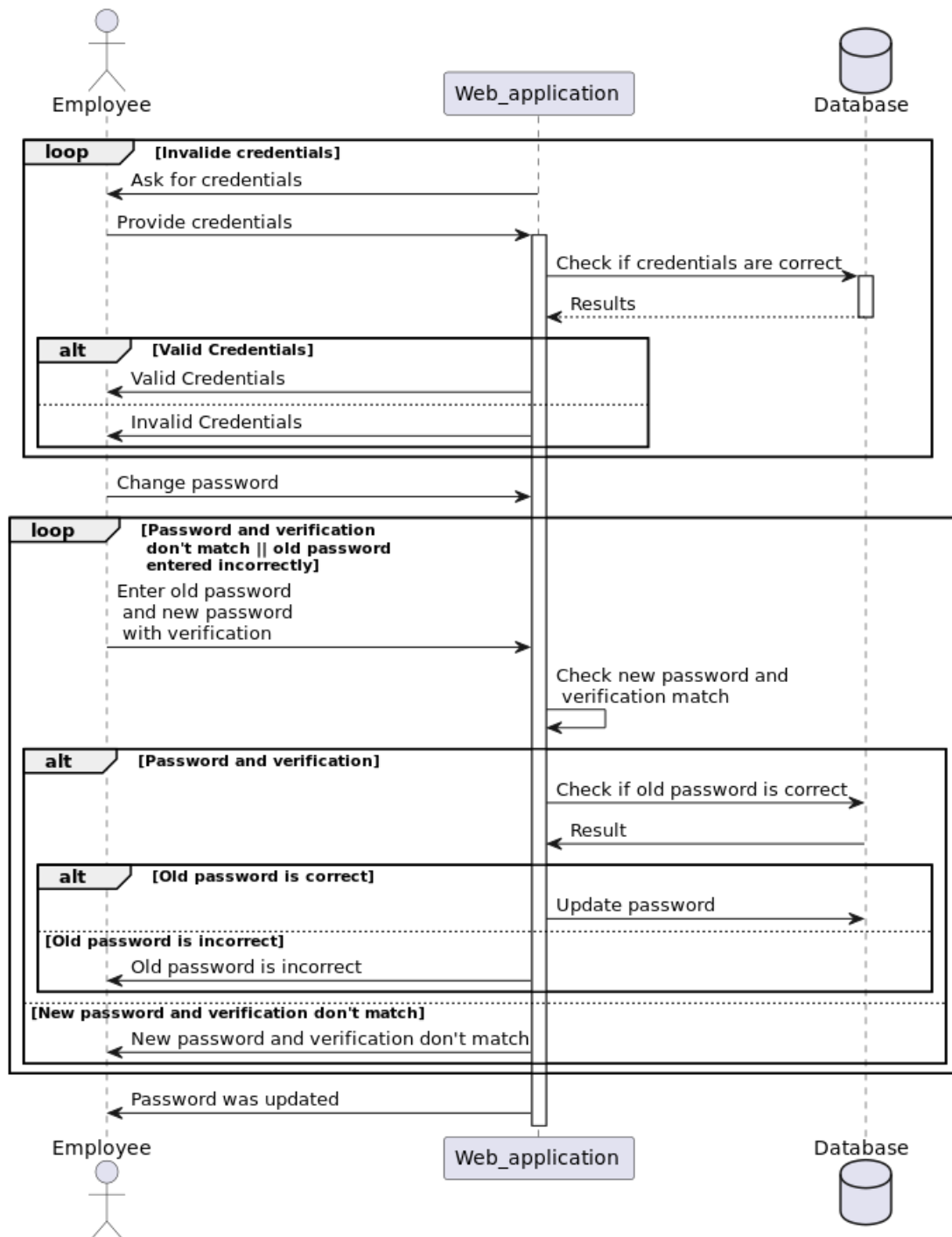
### 3- Check information

This diagram describes the interaction between the user interface, database, and the employee when the user is trying to check his information.



## 4- Change Password

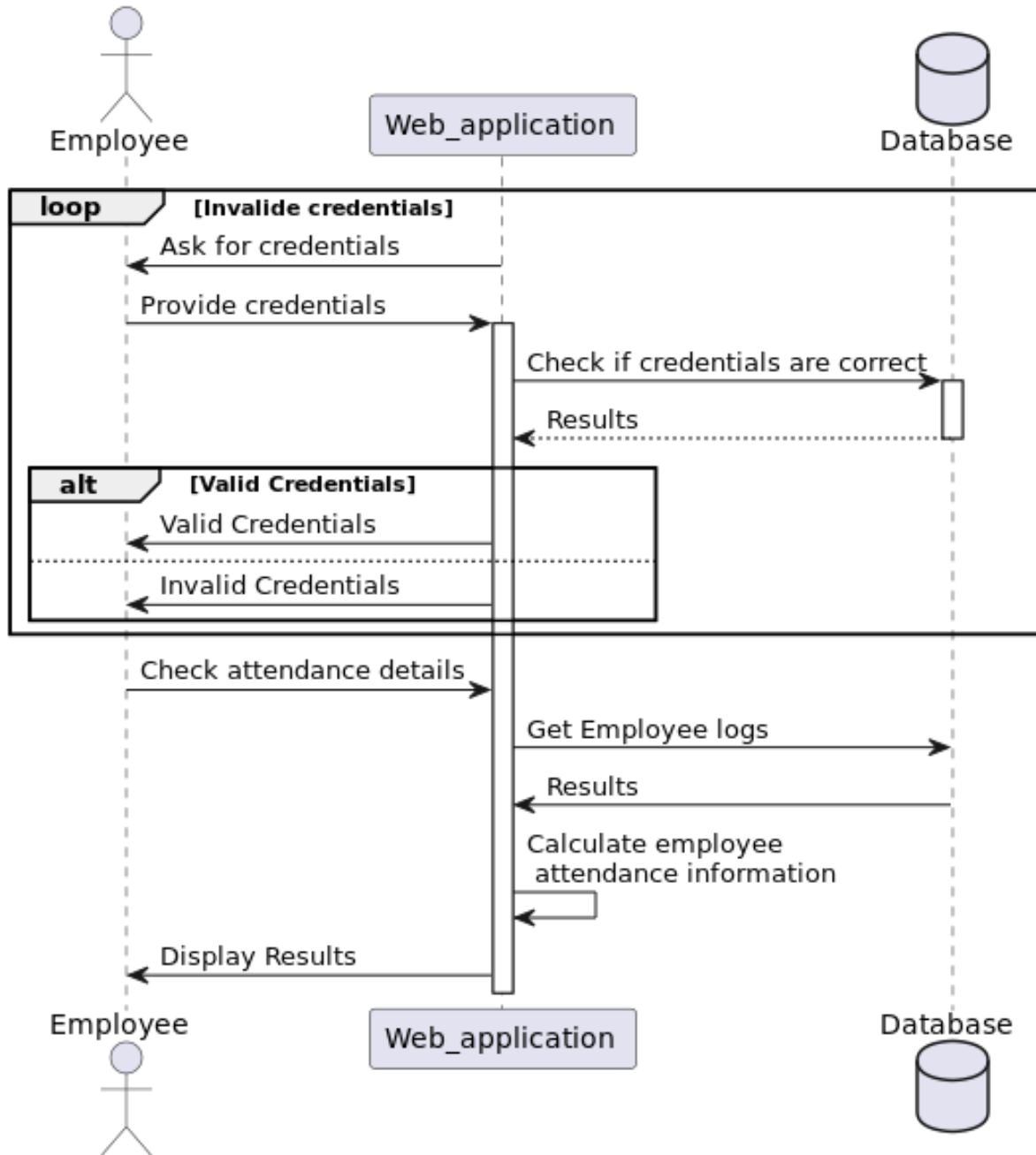
This diagram describes the interaction between the user interface, database, and the employee when the user is trying to change his password.





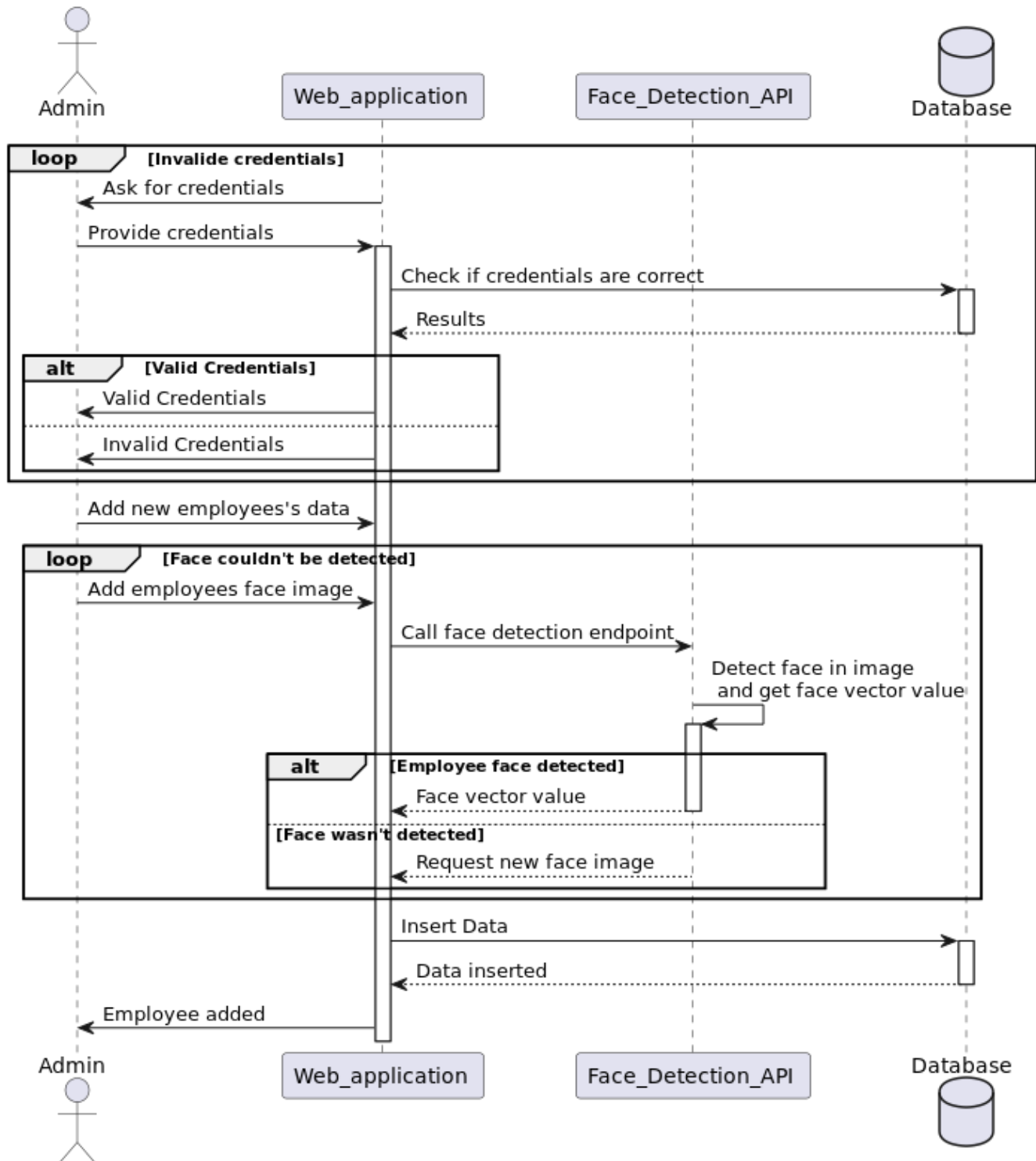
## 5- Check Dashboard

This diagram describes the interaction between the user interface, database, and the employee when the user is trying to check dashboard.



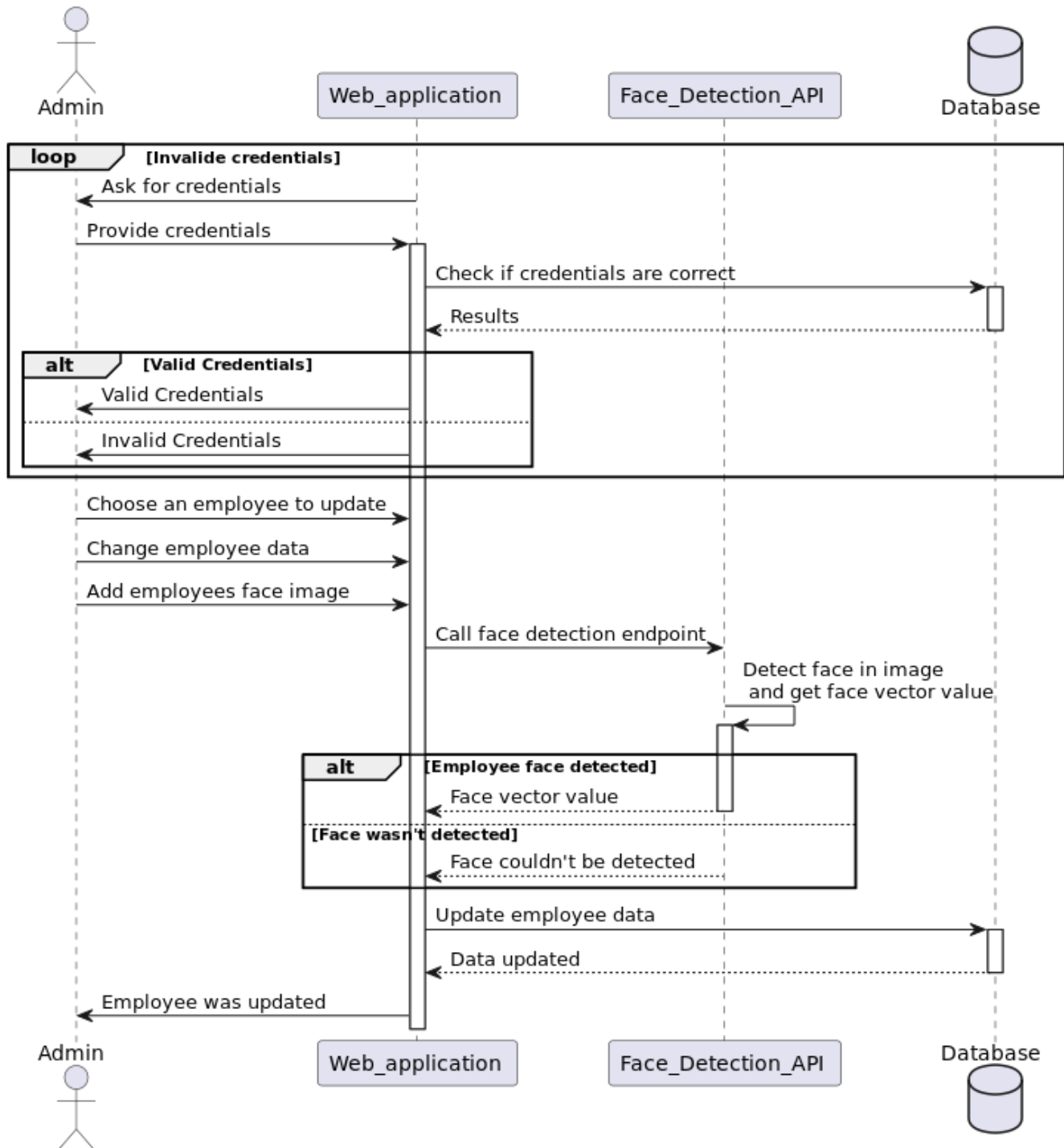
## 6- Admin Add employee

This diagram describes the interaction between the user interface, database, and the admin when the admin is trying to add an employee.



## 7- Admin Update employee

This diagram describes the interaction between the user interface, database, and the admin when the admin is trying to update employee information.



## 5. FACE RECOGNITION MODEL

---

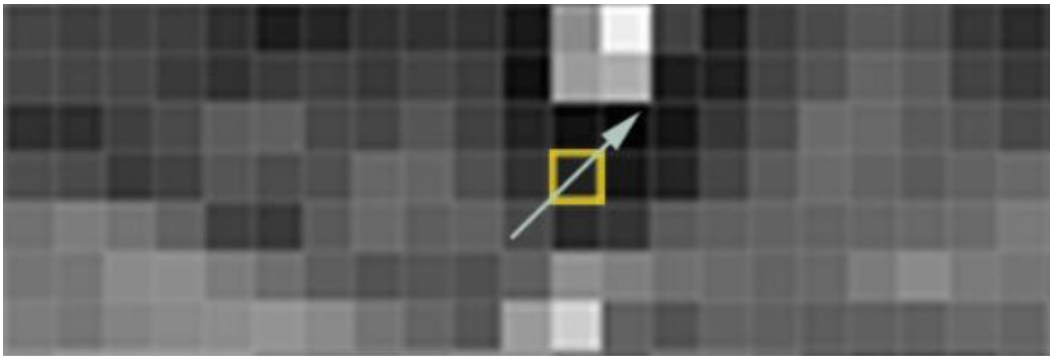
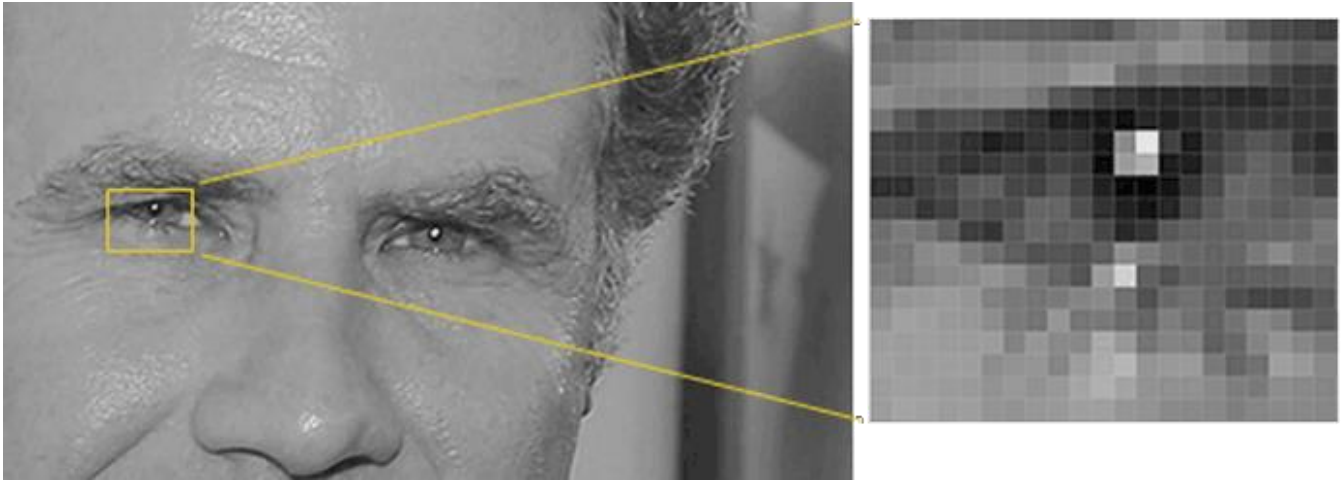
Face recognition is a series of several related problems:

- First, look at a picture and find all the faces in it
- Second, be able to recognize the same person no matter which direction his face is facing.
- Third, be able to pick out unique features of the face that the model can use to tell it apart from other people.
- Finally, compare the unique features of that face to the face embedding of the employee trying to check in/out (identified by the QR code scanned).

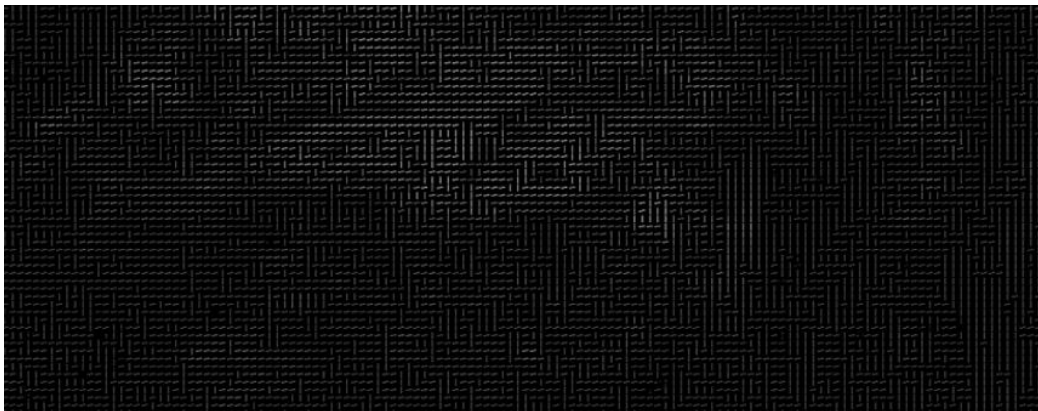
Firstly, the system must locate all the faces in an image. This can be done by converting the image to black and white:



and then using an algorithm called Histogram of Oriented Gradients (HOG) to analyze each pixel in the image. HOG looks at the pixels surrounding each pixel, and determines the direction in which the image is getting darker:



This process is repeated for every pixel, resulting in an image with gradients showing the flow from light to dark pixels:



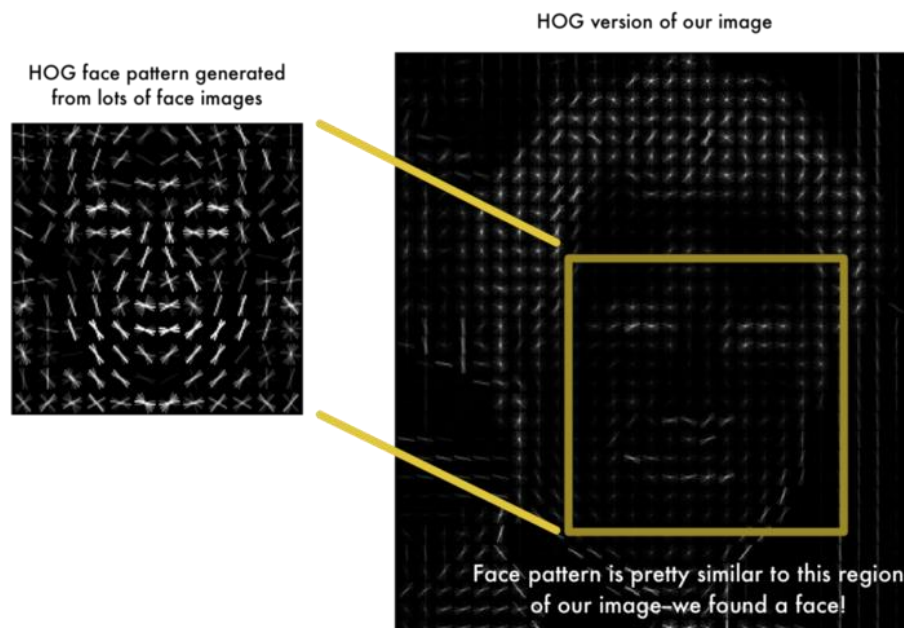


However, using gradients for every pixel is too detailed, so the image is divided into small squares and the gradients in each square are counted and replaced with the strongest arrow direction.



The original image is turned into a HOG representation that captures the major features of the image regardless of image brightness.

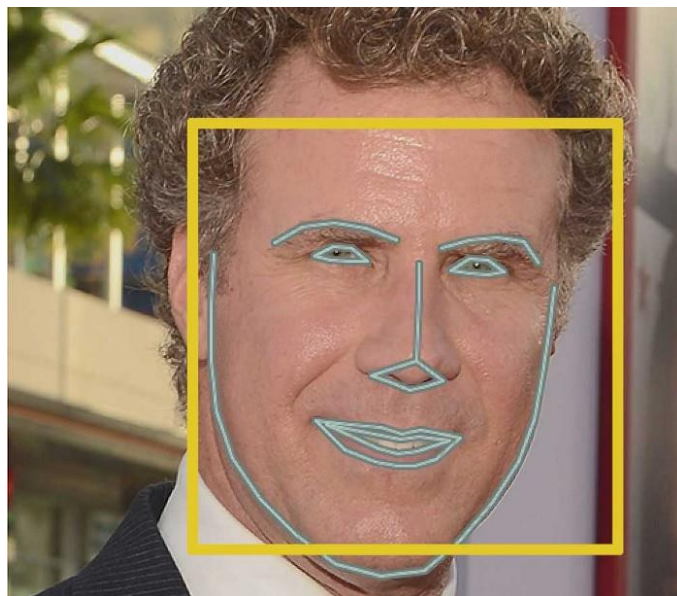
This simplified representation, called the HOG image, can then be used to locate faces by finding parts of the image that resemble a known HOG pattern.



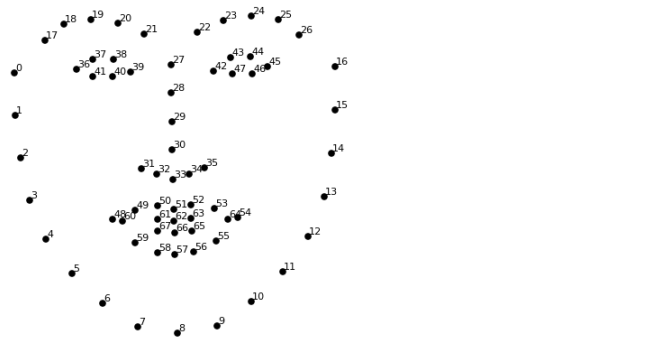
Using this technique, we can now easily find faces in any image:



Once faces have been located, the system must be able to recognize the same person regardless of the orientation of their face. To do this, an algorithm called face landmark estimation is used to identify 68 specific points on the face, such as the top of the chin and the edges of the eyes. These landmarks are used to align the face so that the eyes and mouth are centered, making it easier to compare different faces

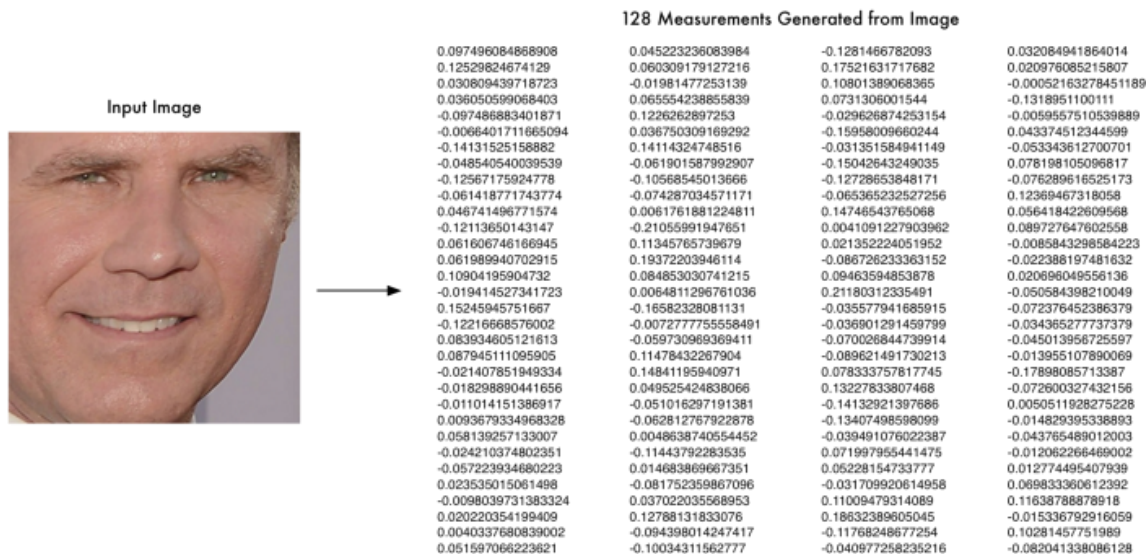


Here's the result of locating the 68 face landmarks on our test image:



Next, the system must identify unique features of the face that can be used to distinguish it from other faces. These features are called face embeddings, which are real-valued feature vectors, the output feature vector is a 128-d (a list of 128 real valued numbers) that is used to quantify the face. The face embeddings are extracted using a deep learning model trained on a dataset of labelled faces.

Our network quantifies the faces, constructing the 128-d embedding (quantification) for each.



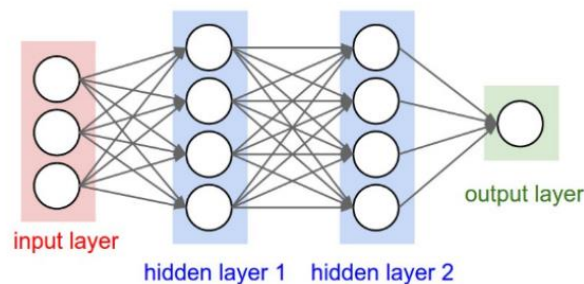
After creating a face embedding for an employee, the output will be stored in the database, that way each whenever an employee tries to check in or out, the model will create a face embedding value in real-time and compare it with the stored face embedding value.

## 6. THE MODEL ... IN DETAILS

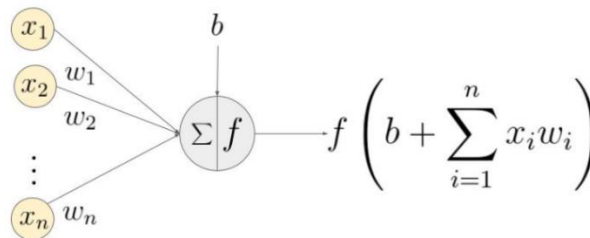
---

To dive deeper, we are going to explain neural network, then convolutional neural network and how we trained the model.

A neural network is a series of algorithms that try to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates.

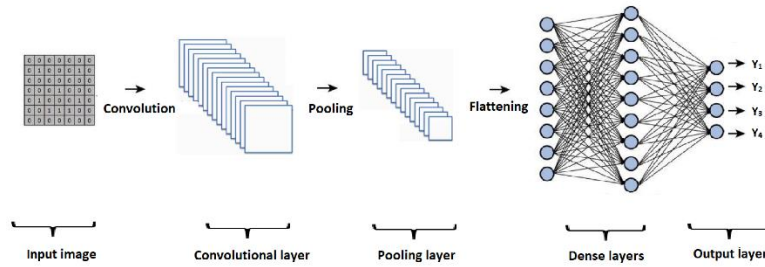


The output of a neuron is a function of the weighted sum of the inputs plus a bias. An activation function in a neural network defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network. The bias is used to shift the activation function by adding a constant to the input.



An example of a neuron showing the input ( $x_1 - x_n$ ), their corresponding weights ( $w_1 - w_n$ ), a bias ( $b$ ) and the activation function  $f$  applied to the weighted sum of the inputs.

A convolutional neural network (CNN) is a type of neural network specifically designed for visual data processing. It uses convolutional layers to extract features from images and pooling layers to down sample the data. These features are then passed through fully connected layers to make predictions or classifications. CNNs are widely used for tasks like image recognition and object detection.



How do we train the network?

By using FaceNet which is a pre-trained CNN which embeds the input image into an 128 dimensional vector encoding. It is trained on several images of the face of different people. Although this model is pre-trained. But, it still struggles to output usable encoding for unseen data. We want this model to generate encoding such that there is less distance between encoding of the images of same person, and more distance between encoding of the different persons. To achieve above goal on our own images we will train FaceNet model on Triplet Loss function. The triplet loss function takes face encoding of three images anchor, positive and negative. Here anchor and positive are the images of same person whereas negative is the image of a different person.

$$Loss = \sum_{i=1}^N \left[ \|f_i^a - f_i^p\|_2^2 - \|f_i^a - f_i^n\|_2^2 + \alpha \right]_+$$

As for the Histogram of Oriented Gradients (HOG) method which operates by calculating the gradient of the image intensity at each pixel, enabling the detection of significant changes in intensity. To compute the gradient, the image is divided into small cells, and for each pixel within a cell, the gradient orientation and magnitude are determined. The gradient orientation represents the direction of the steepest intensity change, while the magnitude corresponds to the rate of change in that direction. These computations are performed using the following mathematical equations:

$$G_x = I(x+1, y) - I(x-1, y)$$

$$G_y = I(x, y+1) - I(x, y-1)$$

$$Magnitude = \sqrt{G_x^2 + G_y^2}$$

$$Orientation = \text{atan2}(G_y, G_x)$$

Here,  $I(x,y)$  denotes the intensity of the pixel at position  $(x,y)$  in the image. By calculating the gradients for each pixel, the method captures local intensity variations.

Next, the gradient orientations are binned into a histogram. The range of orientations is divided into multiple bins, and each bin represents a specific range of orientations. The magnitudes of



the gradients are weighted according to their corresponding orientations and accumulated in the appropriate bin of the histogram. This process is expressed by the following mathematical equations:

$$\text{Histogram}[\text{bin}] += \text{Magnitude} * \text{Weight}$$

Here,  $\text{Histogram}[\text{bin}]$  represents the bin in the histogram,  $\text{Magnitude}$  is the magnitude of the gradient, and  $\text{Weight}$  is the weight associated with the gradient orientation and determines its contribution to the bin.

The resulting histogram is a vector that represents the distribution of orientations and serves as a feature descriptor for the cell.

To generate a feature vector for the entire image, the histograms from all cells are concatenated. Each cell's histogram contributes to the final feature vector, representing the distribution of gradient orientations throughout the image.

Subsequently, this feature vector is employed to train a classifier specifically designed to detect faces. The classifier learns patterns and discriminative information from the HOG feature vectors and can effectively identify faces in images. The training process involves applying mathematical equations and techniques specific to the chosen classifier algorithm, such as Support Vector Machines (SVM) or Neural Networks, to optimize the classifier's parameters and enhance its performance in face detection.

Face alignment is a crucial step in facial analysis that involves detecting and positioning facial landmarks in an image to a standardized position. To accomplish this, a popular approach is to employ a facial landmark detector, which is a machine learning algorithm trained to identify specific points on the face, such as the corners of the eyes and mouth. The detector utilizes a set of mathematical equations and algorithms to accurately locate these landmarks.

Once the facial landmarks are detected, the image can be transformed using various geometric operations like translation, rotation, and scaling. These transformations are aimed at aligning the detected landmarks with a predefined reference position or template. Different techniques can be employed for the actual transformation, such as affine transformations or thin-plate splines, which are mathematically formulated methods for warping the image.

The process of face alignment involves applying mathematical equations specific to each transformation. These equations manipulate the coordinates of the detected landmarks to align them with the reference position. For instance, translation involves shifting the image based on the difference between the detected and reference landmark positions. The translation equations can be expressed as:

$$x_{\text{new}} = x - (x_{\text{ref}} - x) \quad y_{\text{new}} = y - (y_{\text{ref}} - y)$$

Rotation, on the other hand, involves rotating the image around a specific point or axis. The rotation equations can be expressed as:

$$angle = atan2(y_{ref} - y, x_{ref} - x)$$

$$x_{new} = x * cos(angle) - y * sin(angle)$$

$$y_{new} = x * sin(angle) + y * cos(angle)$$

Lastly, scaling modifies the size of the image based on the ratio between the detected and reference landmark distances. The scaling equations can be expressed as:

$$x_{new} = x * scale\_factor$$

$$y_{new} = y * scale\_factor$$

By applying these mathematical equations, the facial landmarks can be accurately aligned with the reference position, enabling further analysis and processing of the face image.

Face embedding is a sophisticated technique that leverages the power of deep neural networks to map a face image into a high-dimensional vector space. This mapping is achieved by constructing a neural network architecture with multiple layers of interconnected neurons. Each neuron performs a linear transformation, followed by a non-linear activation function, allowing the network to capture complex relationships and extract meaningful features from the input face image.

During the training phase, the network learns to generate embeddings that effectively represent the unique characteristics of each face. This is accomplished by optimizing a loss function that encourages the embeddings of images belonging to the same person to be close together in the vector space while pushing the embeddings of different individuals further apart. By minimizing the distance between embeddings of images from the same person and maximizing the distance between embeddings of different individuals, the network becomes capable of discriminating between different faces.

The resulting embeddings obtained from the network can be employed for face recognition tasks. To compare the similarity between two face embeddings, distance metrics like Euclidean distance or cosine similarity are commonly used. Euclidean distance measures the straight-line distance between two embeddings in the high-dimensional space, and it can be calculated using the following equation:

$$d = \sqrt{\sum((x1 - x2)^2)}$$

where d represents the Euclidean distance between embeddings x1 and x2.

Cosine similarity, on the other hand, evaluates the cosine of the angle between two vectors and can be calculated using the following equation:

$$\text{similarity} = \text{dot\_product}(x1, x2) / (\text{norm}(x1) * \text{norm}(x2))$$

where  $\text{dot\_product}(x1, x2)$  represents the dot product between embeddings  $x1$  and  $x2$ , and  $\text{norm}(x)$  represents the Euclidean norm of embedding  $x$ .

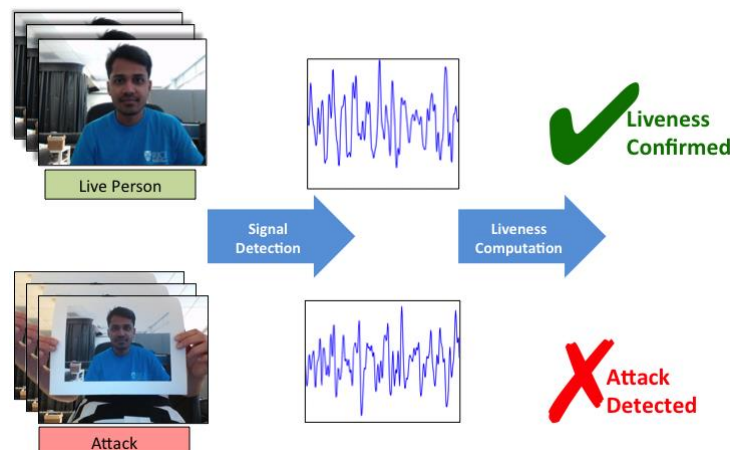
To train the network and optimize the embeddings, mathematical equations and techniques are utilized. One such technique is gradient descent optimization, which iteratively adjusts the network's parameters by computing the gradient of the loss function with respect to the embedding vectors. This gradient information is used to update the parameters in a way that progressively reduces the loss and improves the quality of the embeddings. By iteratively minimizing the loss function using gradient descent, the network gradually learns to generate embeddings that accurately capture the unique features of each face. The update step in gradient descent can be expressed using the following equation:

$$w_{\text{new}} = w - \text{learning\_rate} * \text{gradient}$$

where  $w_{\text{new}}$  represents the updated parameter values,  $w$  represents the current parameter values,  $\text{learning\_rate}$  is a hyperparameter that determines the step size in each iteration, and  $\text{gradient}$  represents the computed gradient with respect to the parameters.

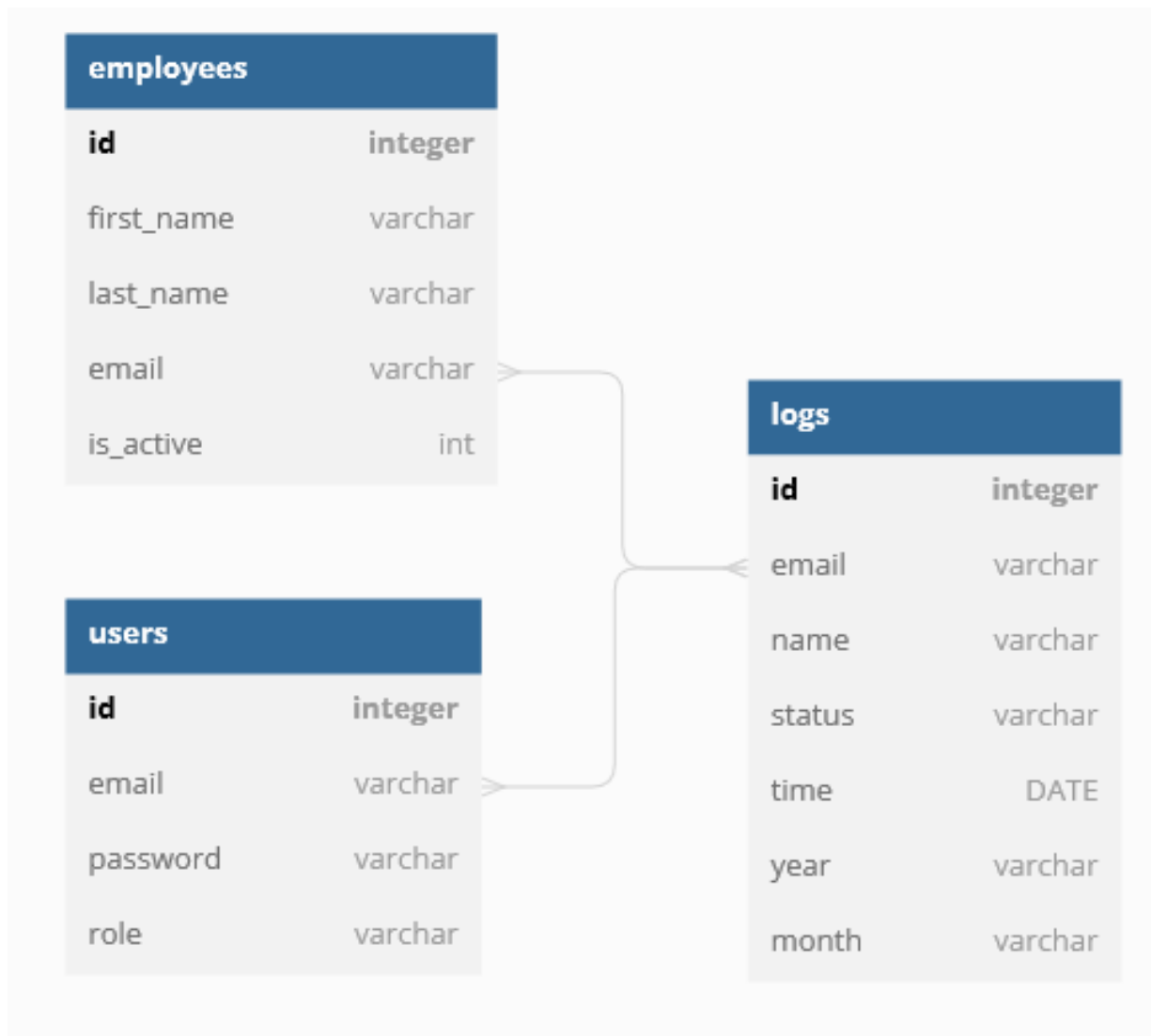
Through the application of these mathematical equations and optimization techniques, face embedding enables the generation of high-quality representations that can be used for various face-related tasks, such as identification, verification, and clustering.

Alongside the face recognition model, we developed a classification CNN that was trained to identify spoofing.



## 7. DATA BASE SCHEMA

---



## 8. API DOCUMENTATION

---

### I-Login API

**POST** /api/auth/login

#### Example Request

```
{
  "email": "varchar",
  "password": "varchar"
}
```

#### Example Response

```
{
  "msg": "varchar",
  "token": "varchar",
  "user": "array"
}
```

#### Example Error

```
{
  "error": "varchar"
}
```

## 2 -Compare faces API

**POST** /api/faceDetection/compareFaces

### Example Request

```
{  
  "image_base64": "varchar",  
  "known_face_embeddings": "varchar",  
  "known_names": "varchar"  
}
```

### Example Response

```
{  
  "msg": "varchar",  
  "name": "varchar",  
  "index": "varchar"  
}
```

### Example Error

```
{  
  "error": "varchar"  
}
```

### 3 -Add Employee API

**POST** /api/employee/addEmployee

#### Example Request

```
{
  "Token": "varchar",
  "email": "varchar",
  "first_name": "varchar",
  "last_name": "varchar",
  "face_embedding": "varchar"
}
```

#### Example Response

```
{
  "msg": "varchar"
}
```

#### Example Error

```
{
  "error": "varchar"
}
```

## 4 -Change Password API

**POST** /api/auth/changePassword

### Example Request

```
{
  "Token": "varchar",
  "old_password": "varchar",
  "new_password": "varchar",
  "email": "varchar"
}
```

### Example Response

```
{
  "msg": "varchar"
}
```

### Example Error

```
{
  "error": "varchar"
}
```



## 5 -Check for spoofing API

**POST** /api/faceDetection/checkForSpoofing

### Example Request

```
{  
  "image_base64": "varchar"  
}
```

### Example Response

```
{  
  "label": "int"  
}
```

### Example Error

```
{  
  "error": "varchar"  
}
```

## 6 -Get Employee by Email API

**GET** /api/employee/getEmployeeByEmail

### Example Request

```
{
  "Token": "varchar",
  "Email": "varchar"
}
```

### Example Response

```
{
  "email": "varchar",
  "first_name": "varchar",
  "last_name": "varchar",
  "face_encoding": "varchar"
}
```

### Example Error

```
{
  "error": "varchar"
}
```

## 7 -Get Employee Information API

GET /api/auth/getEmployeeLogInformation

### Example Request

```
{
  "Token": "varchar",
  "email": "varchar",
  "year": "varchar",
  "month": "varchar"
}
```

### Example Response

```
{
  "daysCameLate": "int",
  "daysLeftEarly": "int",
  "daysAbsent": "int",
  "avgCameLateTime": "int",
  "avgLeftEarlyTime": "int"
}
```

### Example Error

```
{
  "error": "varchar"
}
```

## 8 -Get Employee Logs API

GET /api/log/getEmployeeLogs

### Example Request

```
{
  "Token": "varchar",
  "email": "varchar",
  "year": "varchar",
  "month": "varchar",
  "status": "varchar"
}
```

### Example Response

```
{
  "logs": [
    {
      "id": "int",
      "email": "varchar",
      "name": "varchar",
      "status": "varchar",
      "time": "varchar",
      "year": "varchar",
      "month": "varchar"
    }
  ]
}
```

### Example Error

```
{
  "error": "varchar"
}
```

## 9 -Get Employee Role API

**GET** /api/employee/getEmployeeRole

### Example Request

```
{  
  "Token": "varchar",  
  "Email": "varchar"  
}
```

### Example Response

```
{  
  "role": "varchar"  
}
```

### Example Error

```
{  
  "error": "varchar"  
}
```

## 10 -Get Employees API

**GET** /api/employee/getEmployees

### Example Request

```
{  
  "Token": "varchar"  
}
```

### Example Response

```
{  
  "employees": [  
    {  
      "email": "varchar",  
      "first_name": "varchar",  
      "last_name": "varchar",  
      "face_encoding": "varchar"  
    }  
  ]  
}
```

### Example Error

```
{  
  "error": "varchar"  
}
```

## II -Get Face embedding API

**POST** /api/employee/getFaceEmbedding

### Example Request

```
{  
  "Token": "varchar",  
  "image_base64": "varchar"  
}
```

### Example Response

```
{  
  "face_embedding": "varchar"  
}
```

### Example Error

```
{  
  "error": "varchar"  
}
```

## I2 -Get Filter Logs API

GET /api/log/getFilteredLogs

### Example Request

```
{
  "Token": "varchar",
  "name": "varchar",
  "email": "varchar",
  "year": "varchar",
  "month": "varchar",
  "status": "varchar"
}
```

### Example Response

```
{
  "logs": [
    {
      "id": "int",
      "email": "varchar",
      "name": "varchar",
      "status": "varchar",
      "time": "varchar",
      "year": "varchar",
      "month": "varchar"
    }
  ]
}
```

### Example Error

```
{
  "error": "varchar"
}
```



## 13 -Get Role API

**GET** /api/auth/checkTokenAccess

### Example Request

```
{}
```

### Example Response

```
{  
  "role": "varchar"  
}
```

### Example Error

```
{  
  "error": "varchar"  
}
```

## I4 -Register API

**POST** /api/auth/register

### Example Request

```
{
  "Token": "varchar",
  "email": "varchar",
  "role": "varchar"
}
```

### Example Response

```
{
  "msg": "varchar"
}
```

### Example Error

```
{
  "error": "varchar"
}
```

## I5 -Update Employee API

**PUT** /api/employee/updateEmployee

### Example Request

```
{
  "Token": "varchar",
  "email": "varchar",
  "first_name": "varchar",
  "last_name": "varchar",
  "face_embedding": "varchar"
}
```

### Example Response

```
{
  "msg": "varchar"
}
```

### Example Error

```
{
  "error": "varchar"
}
```

## 9. WIREFRAMES

---

### I- Employee Pages

#### 1- Login Page

##### Attendance

##### for your business

Lorem ipsum dolor sit amet consectetur adipisicing elit. Eveniet, itaque accusantium odio, soluta, corrupti aliquam quibusdam tempora at cupiditate quis eum maiores libero veritatis? Dicta facilis sint aliquid ipsum atque?

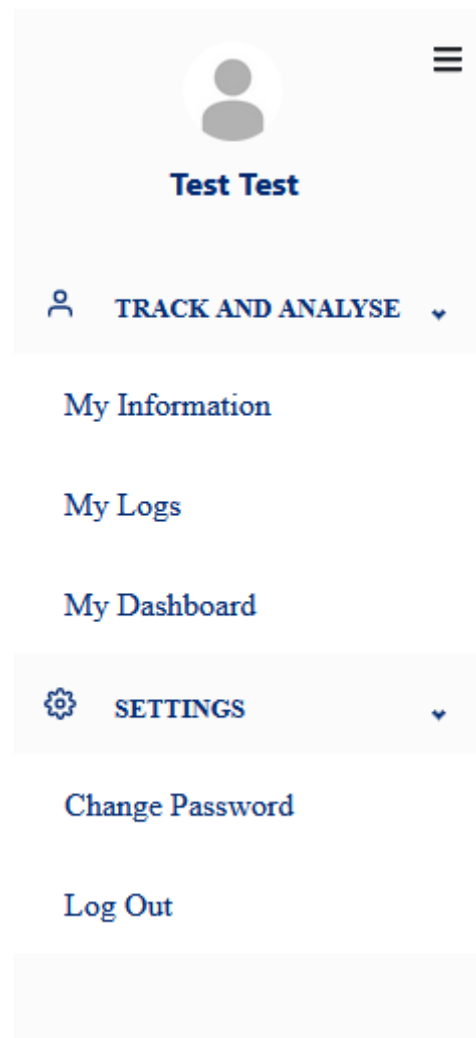
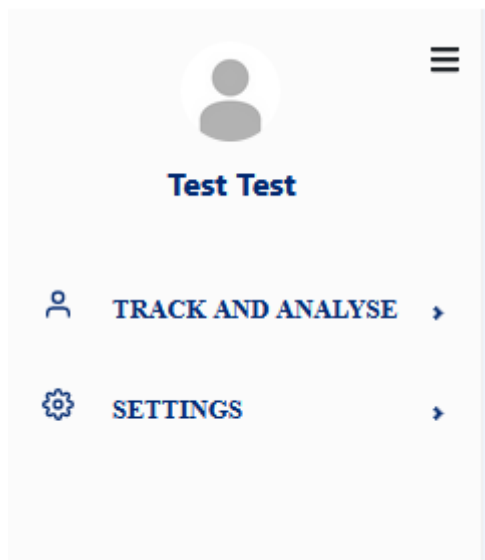
##### Login

Email

Password


LOGIN

## 2- Employee Side Bar Opened and Close



### 3- Employee My Information page

ASFR

**Ali Baydoun**

TRACK AND ANALYSE >

MANAGE >

SETTINGS >

#### Employee Information

Email

a.baydoun@company.com

First Name

ali

Last Name

baydoun

Active

Yes

### 4- Employee My Logs Page

#### My Logs

Year

Status

Months

GET LOGS

COLUMNS FILTERS DENSITY EXPORT


Status	Time	Year	Month
in	2023-05-01 16:48:02	2023	05
in	2023-05-13 18:30:01	2023	05
in	2023-05-20 13:40:47	2023	05
out	2023-05-20 13:47:01	2023	05

## 5- Employee My Dashboard Page


### Dashboard

GET LOGS INFORMATION


Days Came Late

 3


Days Left Early

 1


Days Absent

 14

Avg Came Late Time

 439

Avg Left Early Time

 252

## 6- Employee Change Password Page

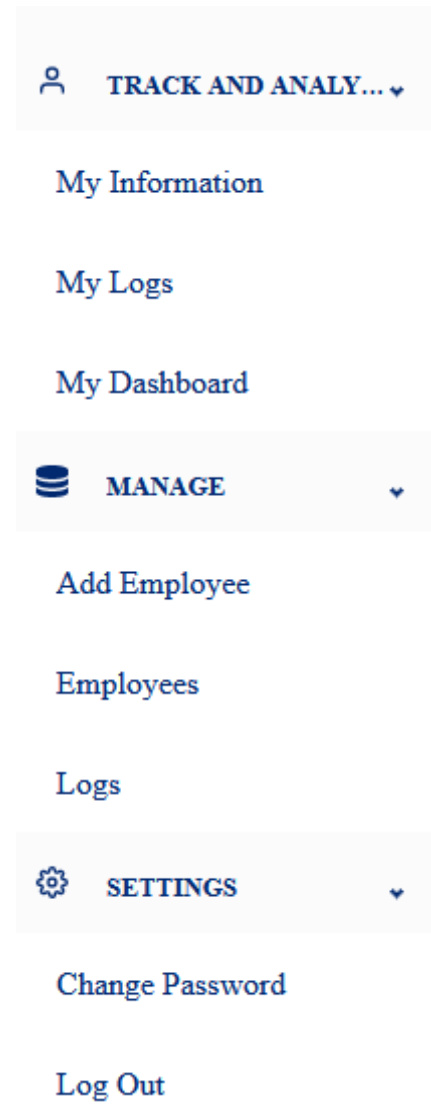
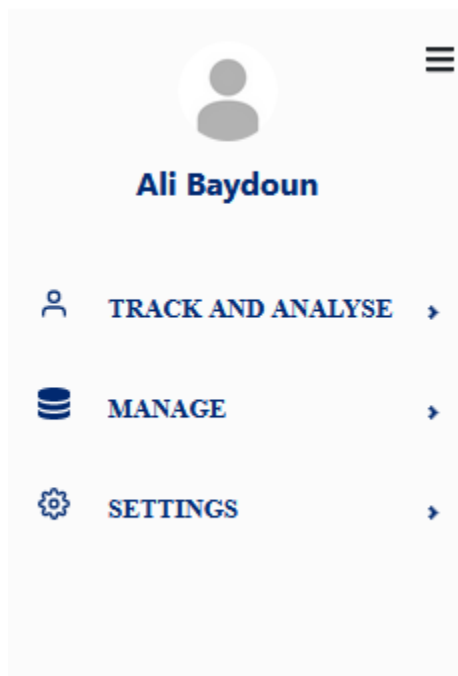
### Change Password

Submit



## 2- Admin pages

### 1- Admin Side Bar Opened and Closed



## 2- Admin Add Employee Page

### Add Employee

First Name

Last Name

Email

No file selected.

Is Active? ☒

Is Admin? ☐

**Submit**

### 3- Admin Employees and Update Pages

#### Employees

COLUMNS FILTERS DENSITY EXPORT				
First Name	Last Name	Email	Is Active	
ali	baydoun	a.baydoun@company.com	true	<button>Update</button>
test	test	test@company.com	true	<button>Update</button>
Rows per page: 100 1-2 of 2 < >				

#### Update Employee

First Name

ali

Last Name

baydoun

Email

a.baydoun@company.com

Employee Image

Browse... No file selected.

Is Active? ☒

Is Admin? ☒

Update

4- Admin Employee Logs Page

Employee Logs

Name

Email

Year

Status

Months

GET LOGS

<div>COLUMNS</div>	<div>FILTERS</div>	<div>DENSITY</div>	<div>EXPORT</div>		
Name	Email	Status	Time	Year	Month
Ali Baydoun	a.baydoun@company.com	in	2023-05-01 16:48:02	2023	05
Ali Baydoun	a.baydoun@company.com	in	2023-05-13 18:30:01	2023	05
Ali Baydoun	a.baydoun@company.com	in	2023-05-20 13:40:47	2023	05
Ali Baydoun	a.baydoun@company.com	out	2023-05-20 13:47:01	2023	05
Rows per page: 100 1-4 of 4 < >					