TED UNIVERSITY

TEDU PASS

-CMPE 465 PROJECT REPORT-

Authors:

Alp Eren Keskin

Doğuhan Cumaoğlu

Doruk Arslan

Hasanalp Temiz

Contents

1. General Information:

A. Summary

Today, facial recognition and matching processes in image processing have made a lot of progress in terms of efficiency. But perhaps the integration of buildings into the entry and exit systems, which is one of the areas where facial recognition and matching will be most used, has not yet been properly performed. Areas such as private schools and company buildings are still accessed with a card and turnstile system, and this is considered to be a rather unnecessary effort for people entering or exiting. To eliminate all this unnecessary effort and further automate the systems, the efficient integration of facial recognition and matching technologies into security systems is a very important issue. For this reason, to take a step to solve this problem, the subject of the project has been determined as the proper integration of facial recognition and matching systems into school security systems. In the project, the previously developed facial recognition and matching technologies were used to communicate with the database in real-time, optimization of this situation was studied, and a

prototypical level of efficiency was achieved using different image processing and database communication techniques.

B. Introduction

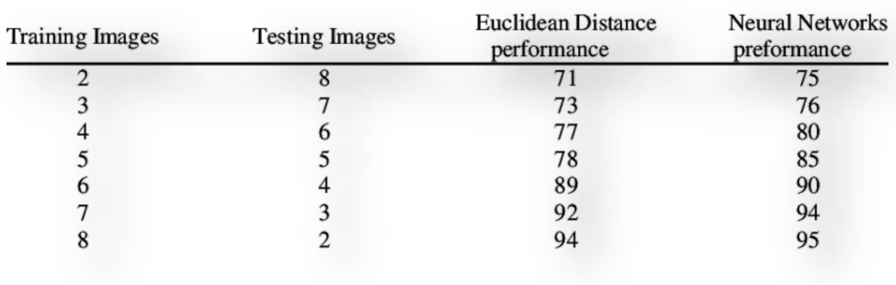
Firstly, in the **TEDU PASS** project, it was determined which technologies, frameworks, and libraries will be used to detect, recognize people's entrance and exit from buildings, and at the same time stay in live communication with the database. There are 4 technologies in the system. Arduino, python OpenCV, Mongo Database, FastApi. Since the system is still in the testing phase, Arduino connects the code to a system other than the computer and tests it is signaling with it. OpenCV performs the process of controlling the computer's camera, taking videos, and creating frames around the recognized faces. Mongo Database is used to keep the times when students log in and out and send them to the front when necessary. When the data created in the FastApi database is needed by both the backside and the front side, this information is pulled from the database and used to implement it into the code. After the technologies were determined, it came to the design stage of the system. As a result of various literature reviews, it was decided to convert a photo into a 128-dimensional array using the encoding method, which is one of the most efficient methods for face capture and recognition, and compare it with faces captured live with capturing video. After finding an efficient face recognition and capture method, it was thought about how the system elements would work. In this system designed for the school, the average time spent by Ted university students reading their cards and leaving school was calculated and it was determined that this time was 6 seconds for entry or exit. Because of this, the system is designed so that someone can enter or exit every 6 seconds. In addition, because the video capture and face recognition system has been designed to capture multiple faces at the same time, the possibility of students experiencing difficulties in the process of entering or leaving the school has arisen. To correct this problem, using the frame structure in image processing, it was ensured that only the faces of people who are at a certain proximity to the video camera were entered into the matching process. A semi-local system for database communication was designed at the final stage since it is quite costly for image processing areas to communicate with a real-time database. As a result, thanks to the literature review and proper system design, a project has emerged that can work as a prototype with a certain accuracy and efficiency ratio.

2. Proof of Method:

A. Literature Review

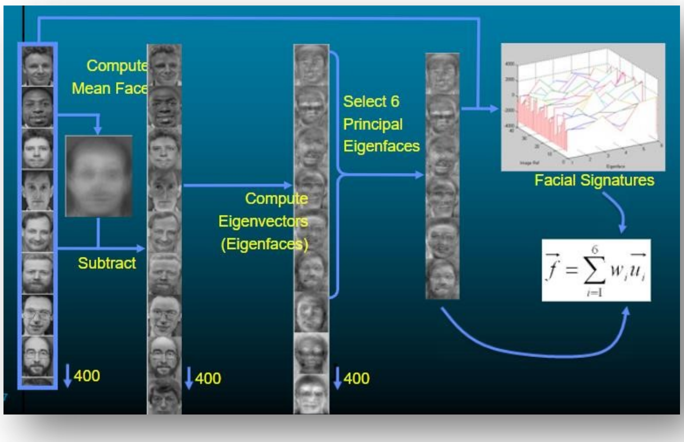
1. Face Recognition Method

- Currently, there are many methods for face recognition and detection. Many of these methods have advantages and disadvantages in certain ways. In terms of face recognition and detection method, the method that will be most suitable for the project, the most flexible, and the most efficient by certain standards have been selected while conducting research for the Tedu Pass project. In the face recognition section of your project, the basic principle components analysis method and Euclidean distance calculation, which is one of the highest accuracy and most optimized methods, are based. According to studies, although the use of a neural network in facial recognition involves a slightly higher accuracy than the use of a basic component analysis system and euclidian distance calculation, it has been found as a result of research that the use of basic component analysis will be more appropriate for the project, both in terms of cost and time.



Training Images	Testing Images	Euclidean Distance performance	Neural Networks preformance
2	8	71	75
3	7	73	76
4	6	77	80
5	5	78	85
6	4	89	90
7	3	92	94
8	2	94	95

- Creating a convolutional neural network system with the appropriate accuracy for a facial recognition system is both time-consuming and nonoptimized for a system that works in real-time. For these reasons, the Principal Components Analysis method and Euclidean distance calculation were used when creating a face recognition system. First of all, all images captured instantaneously in the system and all images stored in the database are reduced to a 128-dimensional array using the PCA method. Then, these arrays in the database and the array of the face image captured from the video are compared and the **Euclidean distance calculation** is used when making this comparison. According to research, multiple distance calculation theorems can be used when comparing these sequences. However, according to studies, among these theorems, the Euclidean distance calculation theorem has been determined as the most suitable calculation method for the **Principle Component Analysis**.


Compute Mean Face
Subtract
400
400
Compute Eigenvectors (Eigenfaces)
400
Select 6 Principal Eigenfaces
Facial Signatures
$\vec{f} = \sum_{i=1}^{6} w_i \vec{u_i}$

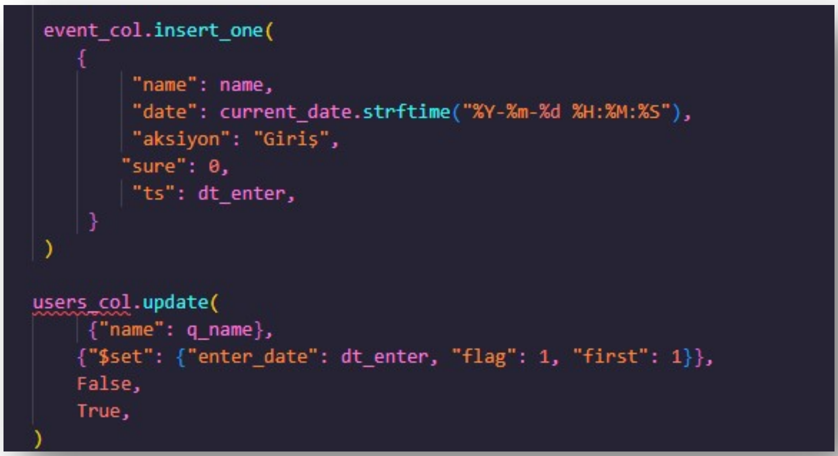
2. Database and Local Communication Methods

- Serious fps drop was observed in the first stage of the system. It was observed that this problem was caused by excessive communication with the database. To solve this problem, the system was run with half local and half database. Before this decision was taken, the following questions were taken into consideration.

- How often will the system be accessing the data during one program execution?
- How logical will the code look in six months? o How will the system be adding data? o How system consume the data?
- The system needs to deal with multiple users and concurrent updates. o Systemsem need flexibility to easily query the data from a variety of angles.
- The system has multiple users, and could gain from making use of an existing security model?

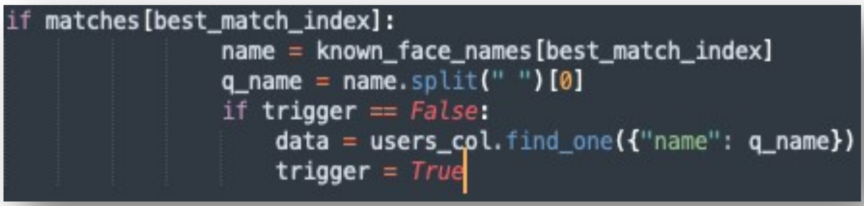
- Since this is the first stage of the system, it solved the local problem. However, it has been observed that in the advanced stages of the system, if the code runs for a long time (6-12 months), the computer will encounter a serious performance problem in ram. Therefore, a semi-local, semi-database system was used in the prototype phase. In the later stages of the project, a complete database-bound system can be used for technologies such as web socket and docker. However, since the number of data was low during the test phase, this semi-database semi-local solution works without any problems and a serious fps increase was observed.

- The code first starts by taking the faces and surnames of the students and staff registered in the system to the locale. This information is in the Mongo Database before this process and is performed with the find () function of Fastapi. After that, if the faces found here are also registered in the system, they are added to the face_names array in the locale. When the system detects that this student/staff is familiar, it creates a red square frame around his/her face. Under this frame, the name and surname of that person registered to the system are created as a separate frame. If the faces found here are also registered in the system, they are added to the face_names array in the locale. When the system detects that this student/staff is familiar, it creates a red square frame around his/her face. Under this frame, the name and surname of that person registered to the system are created as a separate frame. This frame created around the face grows and shrinks in proportion to the size of the person's face and the degree of proximity to the camera. By this point, problems were encountered in the system. First, the more the system communicates with the database, the lower the fps of our camera. The second problem is that when 2 different people enter the camera angle at the same time, the student/personnel in the front of the system should recognize it and perform its processing first. If this did not happen, someone who was not registered in the system in the front could enter the school with the face of the person registered in the system behind. To solve the problem, instead of dividing all the data that needs to be sent from the system to the database one by one and sending them to the database, all of them were collected in one query.



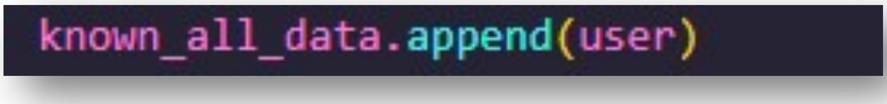
```
event_col.insert_one(
    {
        "name": name,
        "date": current_date.strftime("%Y-%m-%d %H:%M:%S"),
        "aksiyon": "Giriş",
       "sure": 0,
        "ts": dt_enter,
    }
)

users_col.update(
    {"name": q_name},
    {"$set": {"enter_date": dt_enter, "flag": 1, "first": 1}},
    False,
    True,
)
```

- Instead of communicating with the database one by one for each recognized face, all the information in the database was copied to the locale every time the system was turned on. Recognition and checks of their information were made from the locale.



```
if matches[best_match_index]:
                name = known_face_names[best_match_index]
                q_name = name.split(" ")[0]
                if trigger == False:
                    data = users_col.find_one({"name": q_name})
                    trigger = True
```

Old and slow version



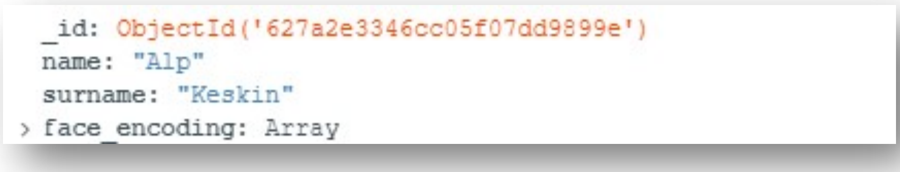
```
known_all_data.append(user)
```

The new and fast version

B. Detailed Explanation of Methods

1. Non-Relational Database

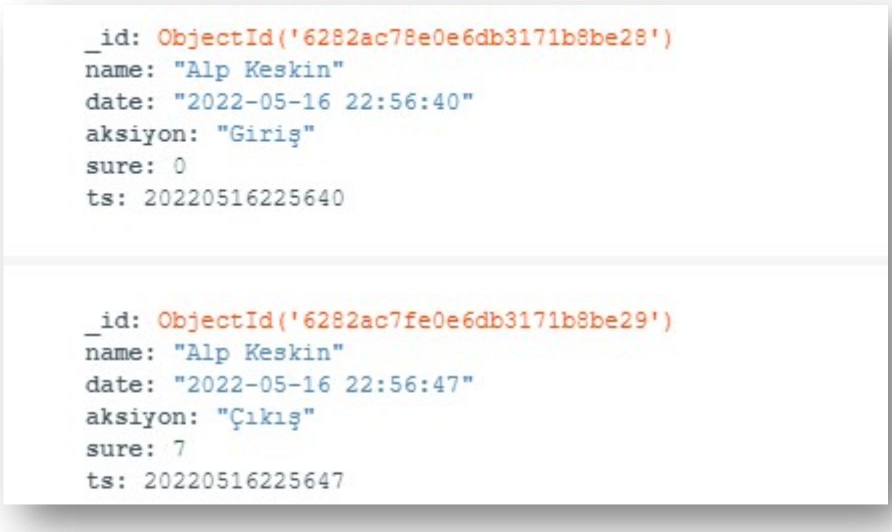
- In the **TEDU PASS** project, the MongoDB database, which is a nonrelational database, was preferred for speed and efficiency because the code is in constant communication with the camera and the database. There are two collections in the selected non-relational database. These are the event_log and users collections In the collection of users, the names, surnames, and facial recognition of all students and faculty members whose school has been registered and whose photos have been registered in the system using the interface of the system are kept in the 128-dimensional array version of the photo for use in face recognition.



```
  _id: ObjectId('627a2e3346cc05f07dd9899e')
  name: "Alp"
  surname: "Keskin"
> face_encoding: Array
```

-

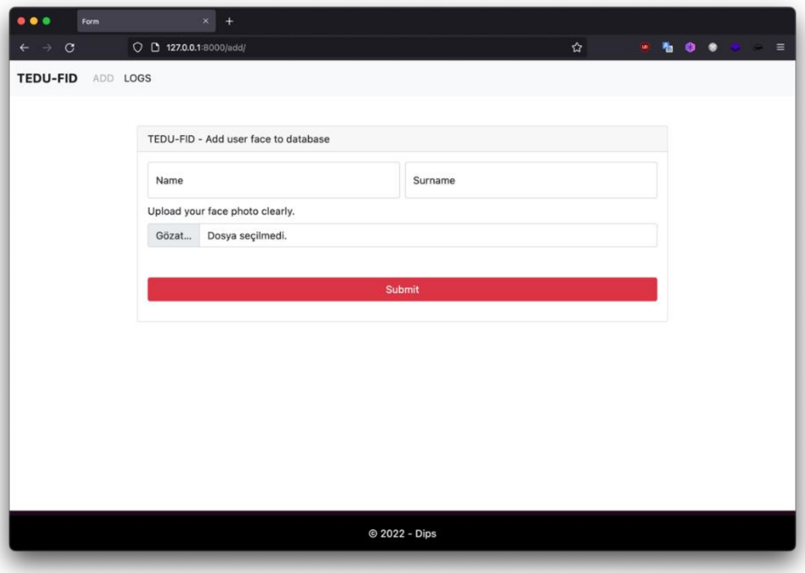
In the event_log collection, if the face of one of the users registered in the system is detected at the entrance or exit of the school, this collection is registered as a log, and only in the security-specific interface, these records are displayed live. In this way, when and how students and faculty members enter the school are recorded in the system.



```
_id: ObjectId('6282ac78e0e6db3171b8be28')
name: "Alp Keskin"
date: "2022-05-16 22:56:40"
aksiyon: "Giriş"
sure: 0
ts: 20220516225640

_id: ObjectId('6282ac7fe0e6db3171b8be29')
name: "Alp Keskin"
date: "2022-05-16 22:56:47"
aksiyon: "Çıkış"
sure: 7
ts: 20220516225647
```

2. User Interface

- In the **Tedu Pass** project, an interface has been developed for tracking logs and installing new users in the system. When designing this interface, a simple appearance was created with the help of HTML and CSS. To translate this web design info into a web application, the FastApi framework was used. The design was visualized in the browser with the Jinja2 templates function of Fast Api. An endpoint was created to add users, and the information received from the form in the interface was sent to the database. However, before this operation was performed, the user's photo taken from the form was brought to the format requested by the face_recognize library and sent to the database.



TEDU-FID - Add user face to database

Name

Surname

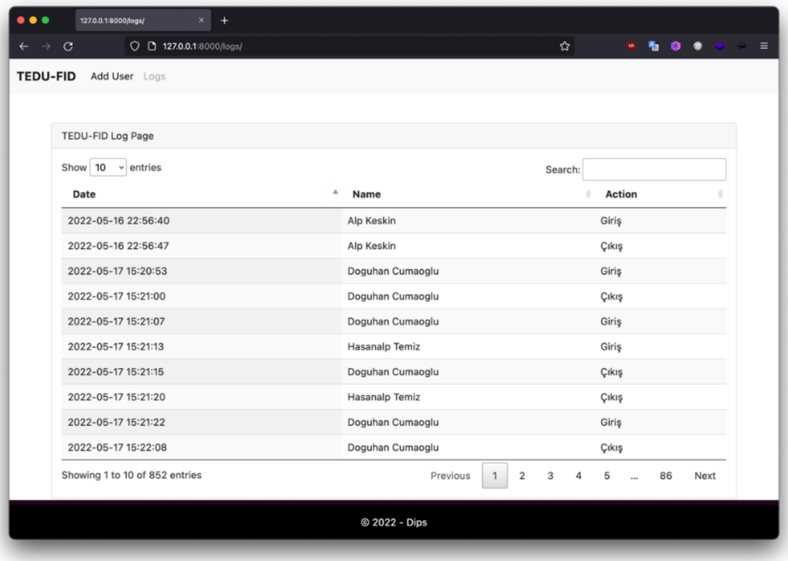
Upload your face photo clearly.

Gözat... Dosya seçilmedi.

Figure 1 User Interface - Add User Screen

- As seen in figure 1, if the system admin wants to create a user, the user's name and surname should be written, and a photograph of the user's face should be uploaded to the application.

- A different web page has been created to keep track of the logs in the project. The current logs on this page are from the project. Each time input and output are detected in the system, these actions are recorded daily in a collection in the database.



TEDU-FID Log Page

Show 10 entries

Search:

Date	Name	Action
2022-05-16 22:56:40	Alp Keskin	Giriş
2022-05-16 22:56:47	Alp Keskin	Çıkış
2022-05-17 15:20:53	Doguhan Cumaoglu	Giriş
2022-05-17 15:21:00	Doguhan Cumaoglu	Çıkış
2022-05-17 15:21:07	Doguhan Cumaoglu	Giriş
2022-05-17 15:21:13	Hasanalp Temiz	Giriş
2022-05-17 15:21:15	Doguhan Cumaoglu	Çıkış
2022-05-17 15:21:20	Hasanalp Temiz	Çıkış
2022-05-17 15:21:22	Doguhan Cumaoglu	Giriş
2022-05-17 15:22:08	Doguhan Cumaoglu	Çıkış

Showing 1 to 10 of 852 entries

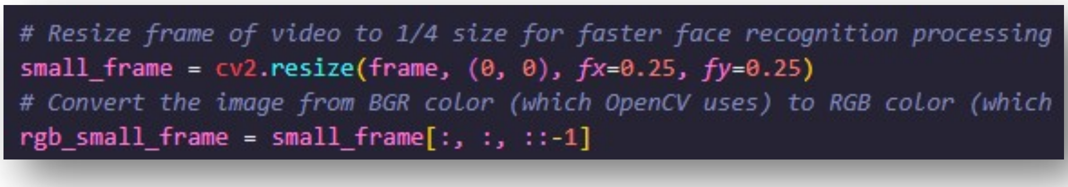
Previous 1 2 3 4 5 ... 86 Next

Figure 2 User Interface - Log Screen

- As seen in figure 2, the logs can be displayed in order. In a log, the name, date, and action information are written as a field. In addition, the system administrator can quickly access the desired log information by using the search bar.

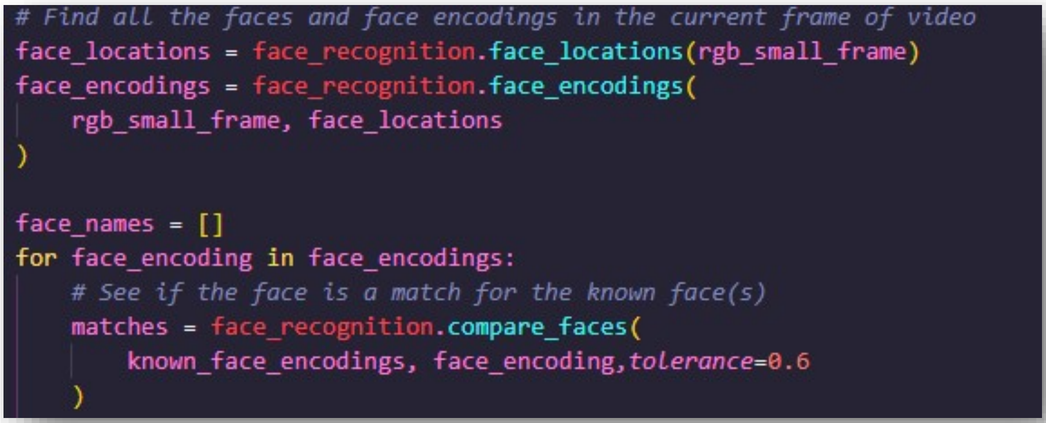
3. Face Detection and Recognize

- At the initial stage of face recognition and detection, a video capture system was installed as a priority. The Video Capture() function of OpenCV was used to perform the video capture process. This is the function to start the camera's power-on and image acquisition. After the video capture process is finished, the next step is to capture the faces in the video. For this purpose, the video frame is resized by 1/4, although it reduces the accuracy rate, primarily to capture and identify faces faster. Then, the images captured in the video are converted from BGR format to RGB format by reversing so that the face_recognition library can use it.



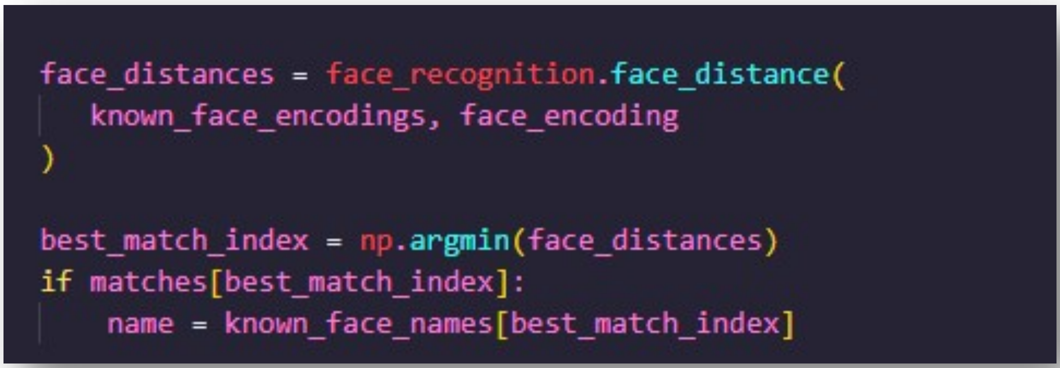
```
# Resize frame of video to 1/4 size for faster face recognition processing
small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
# Convert the image from BGR color (which OpenCV uses) to RGB color (which
rgb_small_frame = small_frame[:, :, ::-1]
```

- After the captured face images are properly formatted in the face_recognition library, the face recognition process begins. First of all, the location of the face captured on the video is found in real-time. Then the face image that is defined with location is converted into a 128dimensional array using the **encoding()** function in the face_recognition library. After all these stages, the encoded image is again uploaded to the database by students and faculty members using **compare_face()**, which is a function of the same library and compared with the images stored as a 128-dimensional array. As a result of comparing the sequences, the images closest to the sequence in the video capture are detected and kept in a list. When comparing arrays, they are compared according to a certain similarity ratio, and this value is sent as a tolerance value in the function parameter. According to the research, the similarity ratio of 0.6 as the tolerance value was determined as the most efficient value and was applied in the project.



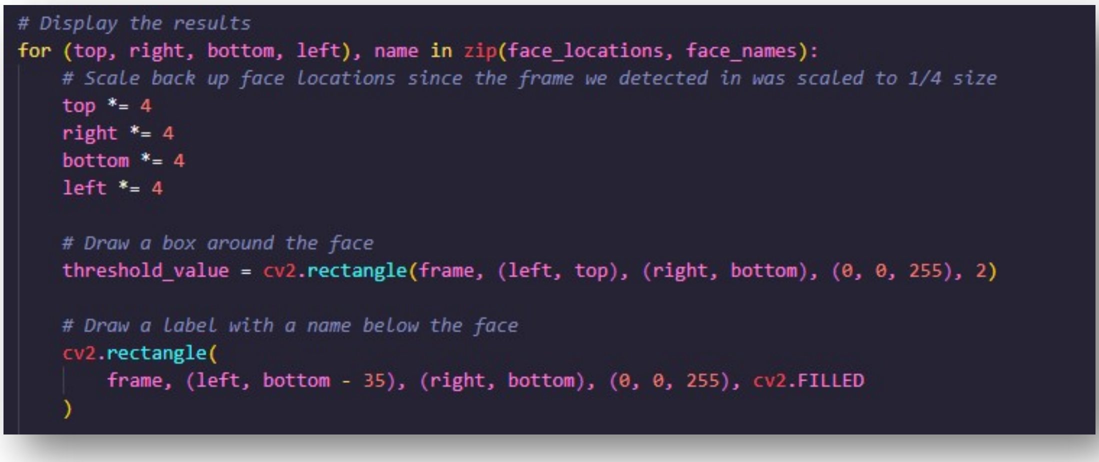
```
# Find all the faces and face encodings in the current frame of video
face_locations = face_recognition.face_locations(rgb_small_frame)
face_encodings = face_recognition.face_encodings(
    rgb_small_frame, face_locations
)

face_names = []
for face_encoding in face_encodings:
    # See if the face is a match for the known face(s)
    matches = face_recognition.compare_faces(
        known_face_encodings, face_encoding,tolerance=0.6
    )
```

- the lower the specified tolerance value of a 128-dimensional array, the greater the similarity to the image captured on the video. To calculate this similarity ratio, the face_distance() function, which is one of the functions of the face_recognition() library, is used. This function compares the images in the database with the image captured from the video and calculates a euclidean distance for each comparison. Then, by applying the argmin() function of the NumPy library to these calculated euclidean distances, the minimum euclidean distance, that is, the measure of the image closest to the image in the video recording, is determined. If this measurement belongs to one of the images that previously matched the image in the video from the database and is stored in the list, the face is detected as defined.



```
face_distances = face_recognition.face_distance(
    known_face_encodings, face_encoding
)

best_match_index = np.argmin(face_distances)
if matches[best_match_index]:
    name = known_face_names[best_match_index]
```

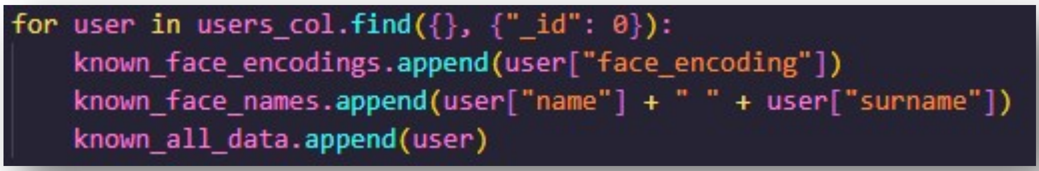
- When the system detects that this student/staff is familiar, it creates a red square frame around his/her face. Under this frame, the name and surname of that person registered to the system are created as a separate frame. This frame created around the face grows and shrinks in proportion to the size of the person's face and the degree of proximity to the camera. If the faces found here are also registered in the system, they are added to the face_names array in the locale. Also, The system treats unrecognized faces as unknown and exempts them from many operations. An unknown face system and recording it locally in a given dynamic way benefit the system in terms of speed and security. After the face detection, recognition, and frame creation processes for the face, the process of communicating with the database begins.



```
# Display the results
for (top, right, bottom, left), name in zip(face_locations, face_names):
    # Scale back up face locations since the frame we detected in was scaled to 1/4 size
    top *= 4
    right *= 4
    bottom *= 4
    left *= 4

    # Draw a box around the face
    threshold_value = cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

    # Draw a label with a name below the face
    cv2.rectangle(
        frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED
    )
```

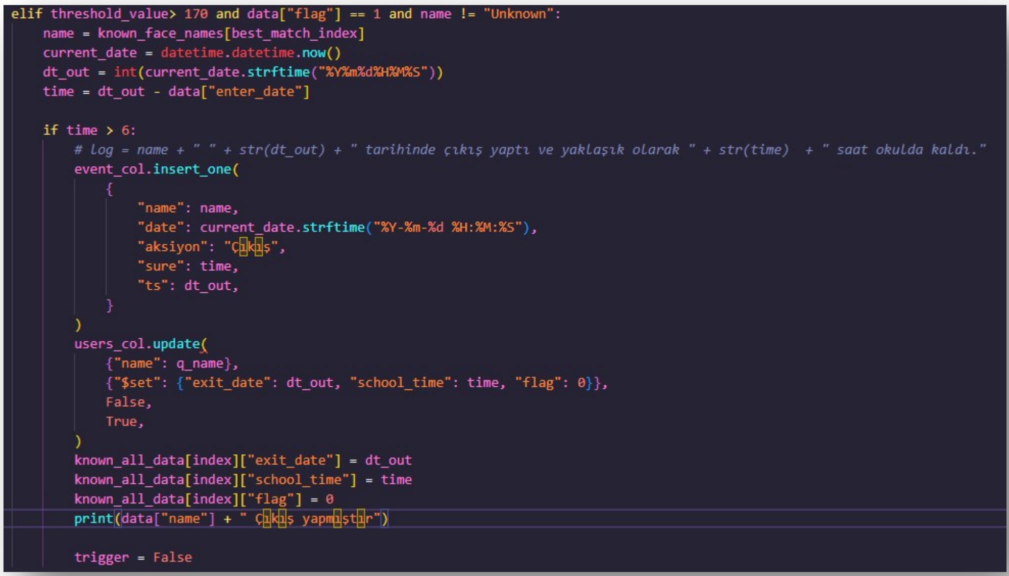
4. Database Communication

- The database communication starts by taking the faces and surnames of the students and staff registered in the system to the locale. This information is in the Mongo Database before this process and is performed with the find () function of FastApi.



```
for user in users_col.find({}, {"_id": 0}):
    known_face_encodings.append(user["face_encoding"])
    known_face_names.append(user["name"] + " " + user["surname"])
    known_all_data.append(user)
```

- There are 3 basic conditions for face detection and sending to the system. Your face must be within the camera's set proximity. Its name should be the Unknown (that is, it should be registered to the system) flag, and it is used to distinguish whether the detection made by the camera at that moment is for input or output. If this value is 0, it means that the student/staff is not at the school, if it is 1, it is inside the school. If these 3 conditions are met, the time that this person appears on the camera should be recorded in the database. In other words, it should be determined what time he came to school and what time he left. These data should be sent to both the front side and the database for the security system. In addition, after a student/staff's face is recognized, this face is not recognized for about 6 seconds. This is used to prevent the entrant from logging in more than once. The system is waiting for this person to come out from the camera’s point of view. If such a precaution were not taken, the same person would have been deemed to have logged in and out during his stay at the camera angle. To solve this problem, 2 separate input and output cameras could be used, but the single-camera system was preferred both to reduce the cost and to make the system safe. Within this expected 6 seconds, other staff/students can continue to enter and exit the school. This state of the process is personal only. In other words, there is no slowdown or disruption in the system. After this 6-second wait, if this person enters the camera again, the system now perceives that person as logging out. These 6 seconds can be changed according to the user and demand. While these operations are saved in the database, they are also saved in the locale. This increases the entry-exit times and security of the process.



```
elif threshold_value> 170 and data["flag"] == 1 and name != "Unknown":
    name = known_face_names[best_match_index]
    current_date = datetime.datetime.now()
    dt_out = int(current_date.strftime("%Y%m%d%H%M%S"))
    time = dt_out - data["enter_date"]

    if time > 6:
        # log = name + " " + str(dt_out) + " tarihinde çıkış yaptı ve yaklaşık olarak " + str(time)  + " saat okulda kaldı."
        event_col.insert_one(
            {
                "name": name,
                "date": current_date.strftime("%Y-%m-%d %H:%M:%S"),
                "aksiyon": "Çıkış",
                "sure": time,
                "ts": dt_out,
            }
        )
        users_col.update(
            {"name": q_name},
            {"$set": {"exit_date": dt_out, "school_time": time, "flag": 0}},
            False,
            True,
        )
        known_all_data[index]["exit_date"] = dt_out
        known_all_data[index]["school_time"] = time
        known_all_data[index]["flag"] = 0
        print(data["name"] + " Çıkış yapmıştır")

        trigger = False
```

3. Results:

A. Efficiency

- It took 12 seconds for an average Ted university student/staff to enter the school using the turnstile system. With this project, this time can be reduced to 6 seconds and even lower according to the customer's request. The obligation to carry cards for students and staff has been abolished.

B. Security

- One of the biggest security vulnerabilities experienced in the turnstile system was observed as someone who is not a staff member/student at the school entering the school with a card either found or given upon request. With the Tedu pass system, this problem has been eliminated. With the web interface, the security personnel will be able to observe the number of staff/students in the school live. Instant tracking was provided with the log documents kept.

C. Accuracy

- Since convolutional neural networks are not included in the project, training data is not available. In addition, during the tests of the project, it was manually tested using the data of 50 people and an average facial recognition accuracy rate of 72% was calculated.

4. References:

- https://dba.stackexchange.com/questions/23124/whats-better-faster-mysqlor-filesystem
- https://stackoverflow.com/questions/14794928/locale-support-in-database
- https://www.ibm.com/docs/en/informix-servers/14.10?topic=environmentdatabase-locale
- https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
- https://machinelearningmastery.com/face-recognition-using-principalcomponent-analysis/
- https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.861.6059&rep=rep1&type=pdf
- https://www.researchgate.net/publication/4283561_FacePerf_Benchmarks_for_Face_Recognition_Algorithms
- https://ieeexplore.ieee.org/document/937541
- https://www.researchgate.net/figure/Performances-of-Euclidean-Distanceand-Neural-Networks_tbl1_221583042

- https://www.researchgate.net/publication/286353466_Facial_Expression_Recognition_Using_Euclidean_Distance_Method