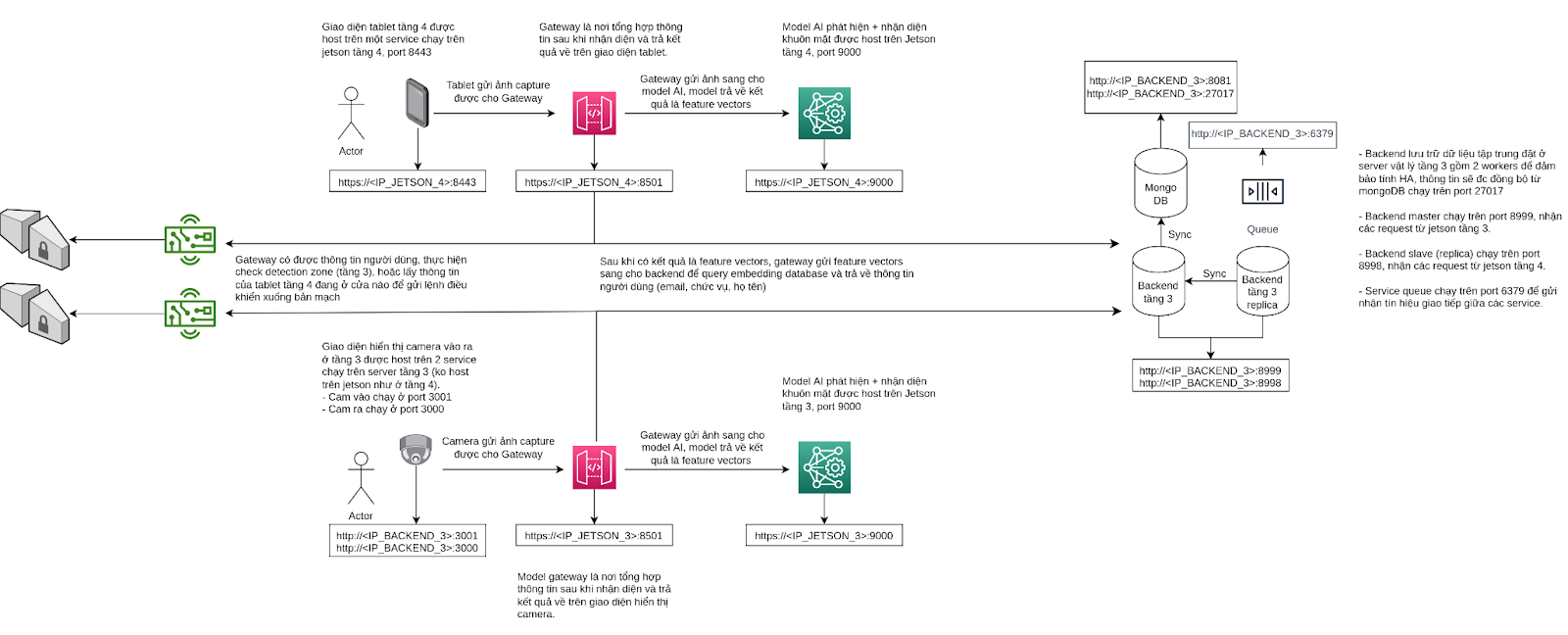
## 1. Thiết kế hệ thống:

<https://drive.google.com/file/d/1pbNiN-5voEZ_XknSrbw754IraXuz6Rjg/view?usp=sharing>



\* Note:

- Về cơ bản, các service trên jetson tầng 4 tương tự như jetson tầng 3, chỉ khác là jetson tầng 4 có host thêm service camera tablet.

## 2. Hướng dẫn triển khai

### 2.1. Requirements:

- Yêu cầu trên tất cả các node đều phải cài docker, docker-compose, có kết nối ssh.

- Mặc định jetpack cho AIBox Jetson đã có docker, chỉ cần cài thêm docker-compose như sau:

| wget -O docker-compose <https://github.com/docker/compose/releases/download/v2.4.1/docker-compose-linux-aarch64>  sudo mv docker-compose /usr/local/bin/docker-compose  sudo chmod +x /usr/local/bin/docker-compose  sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose --force  docker-compose -v |
| --- |

- Backend cài docker theo hướng dẫn: <https://docs.docker.com/engine/install/ubuntu/>

Sau đó cài docker-compose:

| wget -O docker-compose <https://github.com/docker/compose/releases/download/v2.4.1/docker-compose-linux-x86_64>  sudo mv docker-compose /usr/local/bin/docker-compose  sudo chmod +x /usr/local/bin/docker-compose  sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose --force  docker-compose -v |
| --- |

### 2.2. Source code:

Đầu tiên ssh vào server backend, clone source code về:

git clone <https://github.com/hoangph3/Face-KYC>

Chuyển sang branch deploy

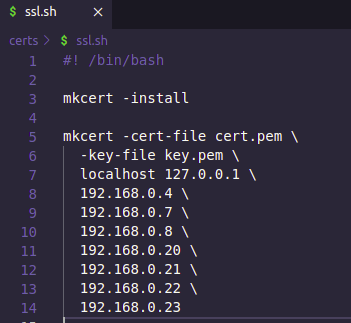
cd Face-KYC

git checkout deploy

Tạo chứng chỉ ssl cho cluster (Backend, AIBox, Tablet):

cd certs

nano ssl.sh

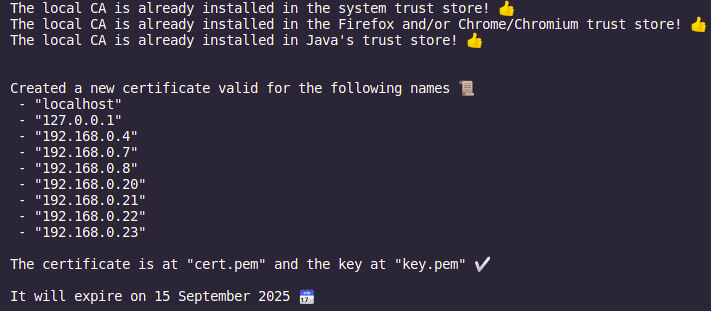


Ở đây backend có địa chỉ 192.168.0.4, AIBox có địa chỉ là 192.168.0.7 và 192.168.0.8 và các tablet có địa chỉ là 4 ip còn lại (192.168.0.20 -> 192.168.0.23)

Tạo cert ssl:

bash ssh.sh

Thông báo tạo thành công như sau:



Nếu báo **mkcert command not found** thì cần cài apt mkcert, chạy từng dòng lệnh sau:

| sudo apt install libnss3-tools  sudo wget <https://github.com/FiloSottile/mkcert/releases/download/v1.4.3/mkcert-v1.4.3-linux-amd64>  sudo mv mkcert-v1.4.3-linux-amd64 mkcert  sudo chmod +x mkcert  sudo cp mkcert /usr/local/bin/ |
| --- |

Sau đó chạy lại lệnh: bash ssh.sh

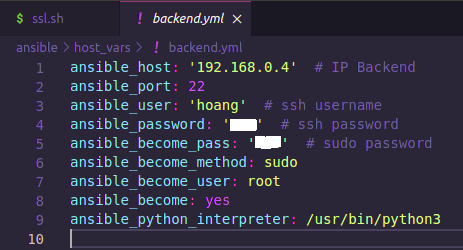
### 2.3. Cấu hình ssh:

Điền ip, username và password ssh của các node vào file trong folder: ansible/host\_vars:



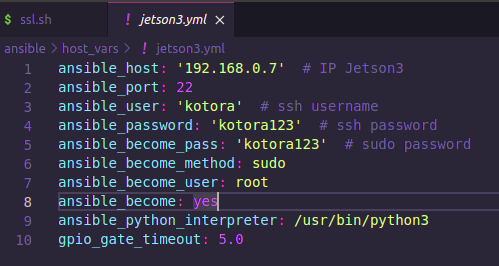
Với backend:

nano ansible/host\_vars/backend.yml

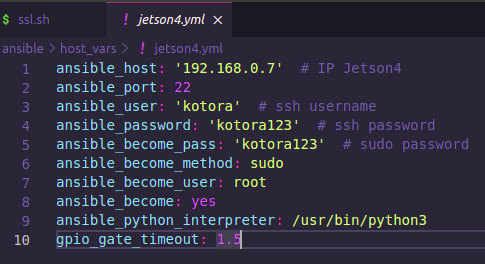


Với AIBox:

nano ansible/host\_vars/jetson3.yml



nano ansible/host\_vars/jetson4.yml



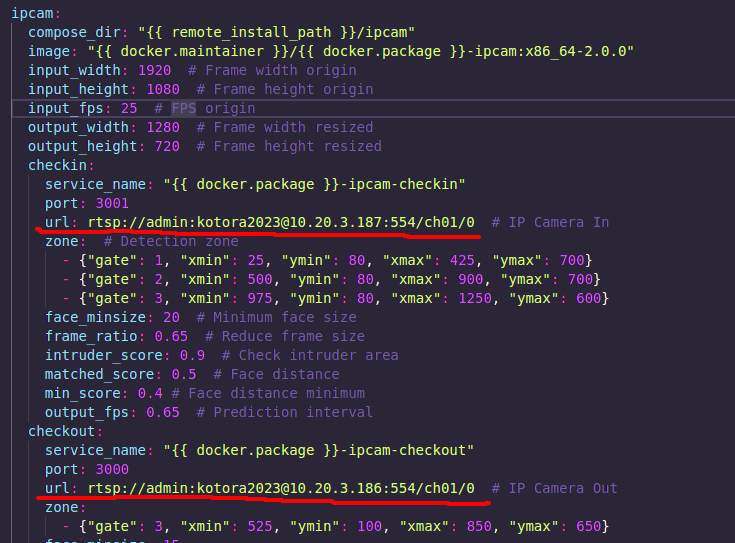
Nếu test chỉ có 1 con Jetson thì để thông tin jetson3 và jetson4 là giống nhau

### 2.4. Cấu hình các service:

Cấu hình các service trong file: ansible/group\_vars/all.yml

nano ansible/group\_vars/all.yml

Sửa địa chỉ camera vào/ra:



Một số tham số có thể phải cần chỉnh tùy thuộc vào tình hình thực tế:

- **zone** là khu vực detection zone với các tọa độ min, max, cổng số bao nhiêu (gate).

- **face\_minsize** là kích thước khuôn mặt tối thiểu để nhận diện (ở đây height và width đều phải lớn hơn minsize)

- **frame\_ratio** là tỉ lệ resize ảnh, ví dụ 0,5 là resize về ảnh chỉ còn bằng một nửa, ratio càng cao thì ảnh càng nét nhưng request sẽ càng chậm, ratio càng nhỏ thì request càng nhanh nhưng độ chính xác thấp do ảnh mờ.

- **intruder\_score** là tỉ lệ phần giao giữa khu vực detection và bounding box của khuôn mặt, ví dụ 0.5 là một nửa khuôn mặt nằm trong khu vực detection là coi như đã nằm trong khu vực detection, còn 1.0 là khuôn mặt phải nằm hoàn toàn trong khu vực detection zone.

- **matched\_score** là khoảng cách lúc search khuôn mặt, khoảng cách càng nhỏ thì càng chặt, giống hệt thì matched\_score = 0.

- **min\_score** là khoảng cách nhỏ nhất lúc search khuôn mặt (< matched\_score).

- **output\_fps** là tốc độ predict, ví dụ fps = 1 là 1 giây predict 1 lần.

### 2.5. Deloy all:

Chạy lệnh:

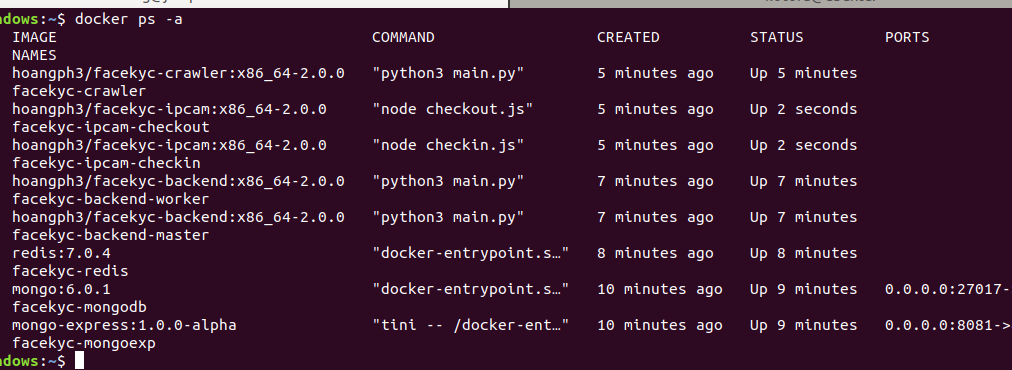
chmod +x run.sh

./run.sh install.yml

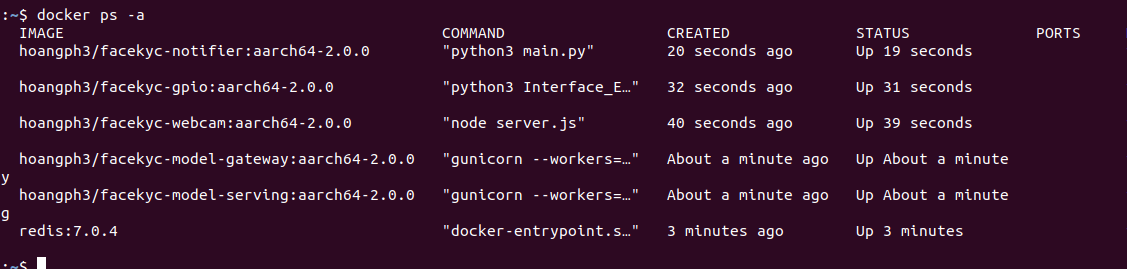
### 2.6. Kiểm tra sau cài đặt:

ssh vào các node, thực hiện chech các service xem đã run lên chưa:

Trên backend: docker ps -a



Trên jetson: docker ps -a



Check log:

docker logs -f <container\_name>

### 2.7. Testing:

Truy cập camIP vào tại địa chỉ: <http://192.168.0.4:3001>

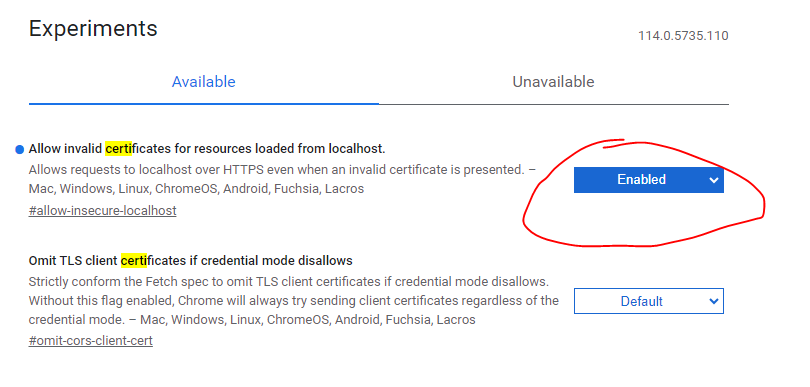
Truy cập camIP ra tại địa chỉ: <http://192.168.0.4:3000>

(\*Note: 192.168.0.4 là địa chỉ backend)

Truy cập webcam tablet từ địa chỉ: <http://192.168.0.7:8443/camera?cam_id=1>

(\*Note: 192.168.0.7 là địa chỉ jetson, query cam\_id được set theo cửa tương ứng, ví dụ cam\_id=2 là cửa 2, cam\_id=3 là cửa 3, …)

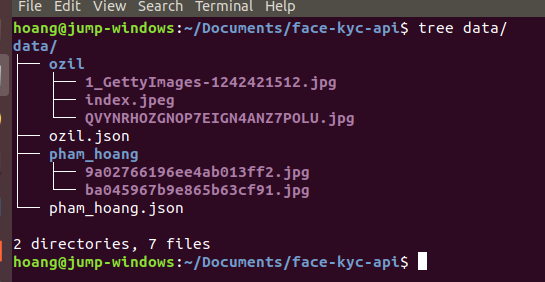
Trên tablet, vào url: chrome://flags/



Cho phép bypass invalid SSL bằng cách Enable: Allow invalid certificates for resources loaded from localhost.

Đăng ký truy cập:

Kiểm tra thư mục data, tạo folder cho mỗi user, copy ảnh vào bên trong folder của mỗi user.



Chú ý tên folder phải trùng với tên email trong file telehouse\_contact.csv (ko có @xxx phía sau), nếu trong file telehouse\_contact.csv chưa có user đó thì phải thêm vào. Ví dụ folder ảnh PhamHoang sẽ có email là PhamHoang@xxx.com, LocHB123 sẽ có email là LocHB123@xxx.com.vn

Chạy script để đăng ký:

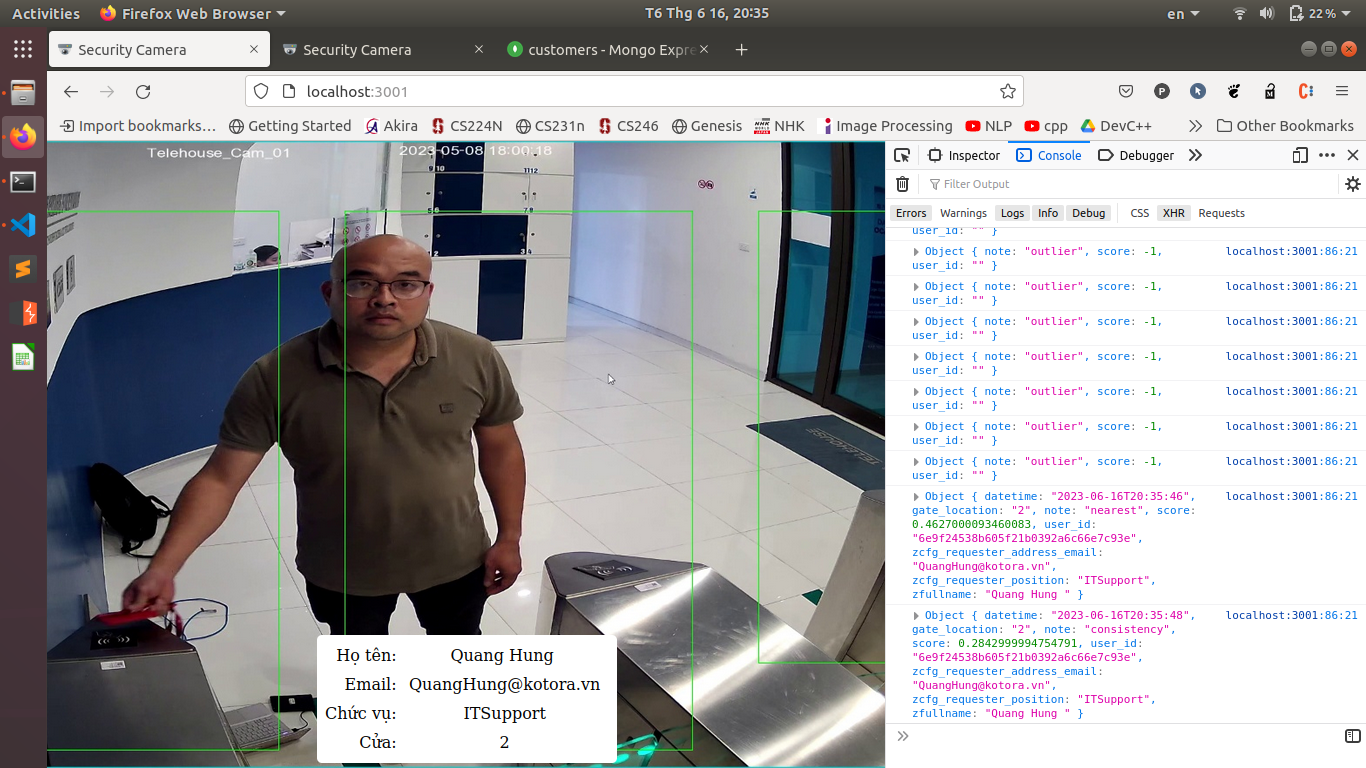
python3 register.py

Mỗi lần chạy script đăng ký thì sẽ đăng ký 1 user trong file telehouse\_contact.csv và lưu vào file registered\_users.txt, nếu đăng ký thất bại mà muốn đăng ký lại cho user nào thì phải xóa user đó trong file registered\_users.txt (chi tiết đã comment trong code).

Kết quả (test với tablet):

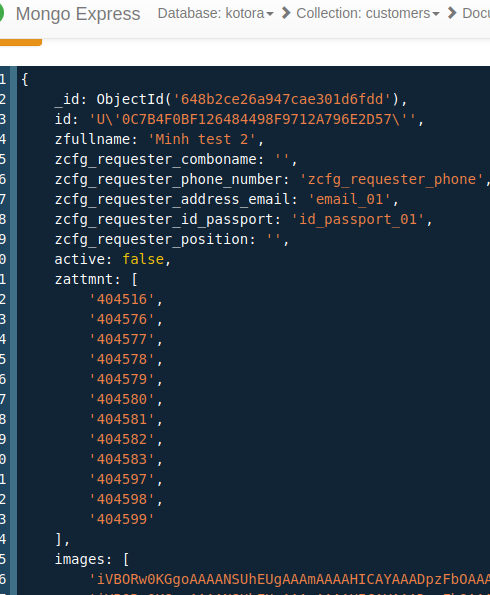


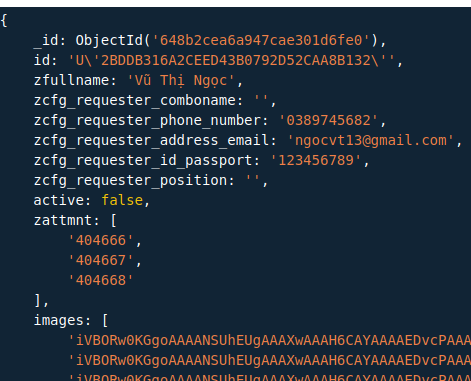
Kết quả (test với Cam tại địa chỉ: <http://192.168.0.4:3001> hoặc <http://192.168.0.4:3000>) trong đó 192.168.0.4 là địa chỉ backend)



Test khả năng đồng bộ dữ liệu từ DC sang Kotora (truy cập vào UI mongo tại địa chỉ: <http://192.168.0.4:8081>) trong đó 192.168.0.4 là địa chỉ backend.

Sau đó truy cập collection customers ta có:





### 2.8. Gỡ cài đặt:

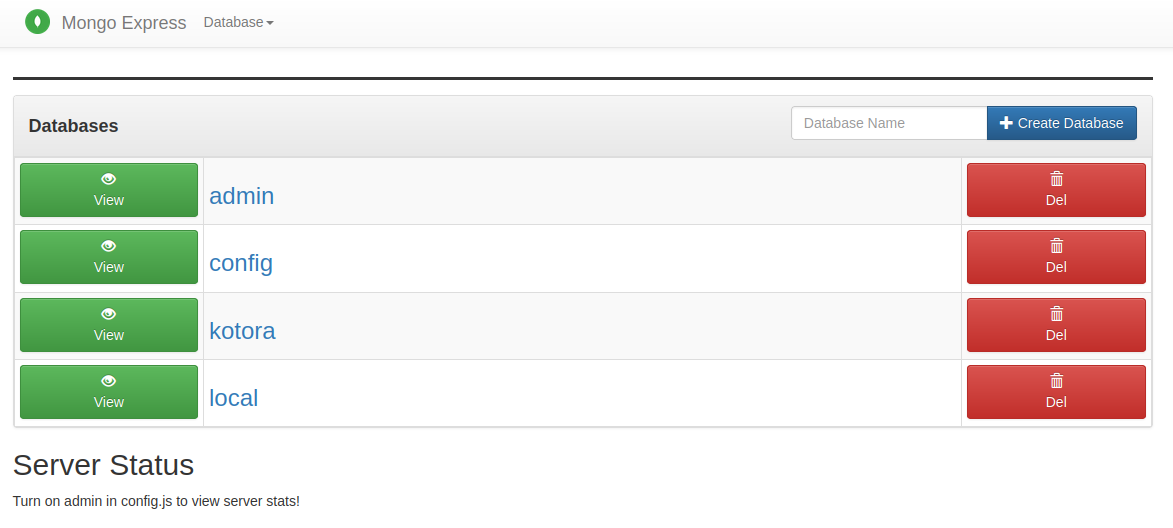
Để gỡ cài đặt, chạy lệnh:

./run.sh uninstall.yml

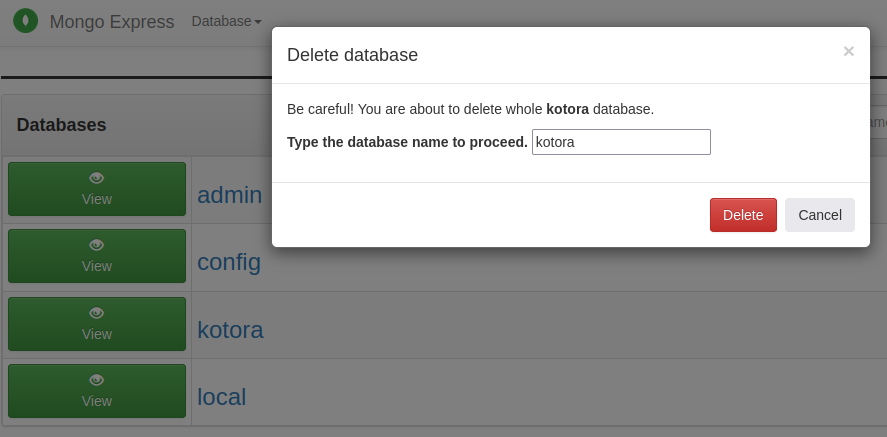
## 3. Hướng dẫn sử dụng

### 3.1. Hướng dẫn xóa toàn bộ DB Kotora (để test full luồng)

- Vào url: <http://192.168.0.4:8081> trong đó 192.168.0.4 là địa chỉ ip của backend:



- Ấn nút Del để xóa DB Kotora.

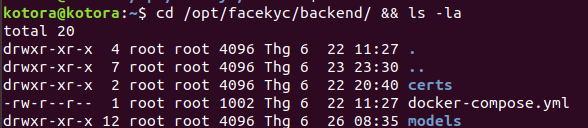


- Confirm để thực hiện xóa toàn bộ DB.

Sau khi xóa xong sẽ xóa toàn bộ dữ liệu của người dùng, tuy nhiên model features vectors vẫn chưa bị xóa. Để xóa model features vectors, ta làm các bước sau:

- Thực hiện ssh vào server backend: **ssh kotora@192.168.0.4** (192.168.0.4 là địa chỉ backend).

- Truy cập vào folder: **cd /opt/facekyc/backend && ls -la**



- Thư mục models là thư mục lưu trữ các models feature vectors, xem các model hiện tại: **cd models/ && ls**



- Mỗi model sẽ được lưu version với định dạng và timestamp.

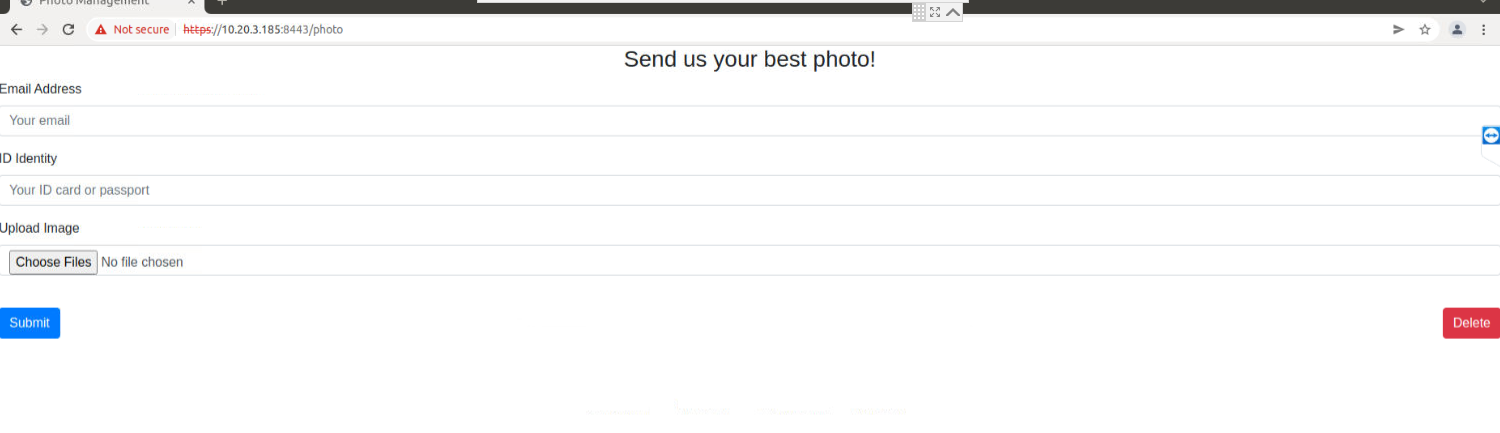
- Để xóa models ta thực hiện lệnh: **sudo rm -rf \***

- Sau khi xóa xong, thực hiện restart các container: **docker restart facekyc-backend-worker facekyc-backend-master**

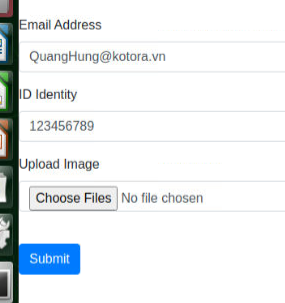
- Sau khi restart container, test lại sẽ ko nhận mặt là ok.

### 3.2. Hướng dẫn upload ảnh thủ công:

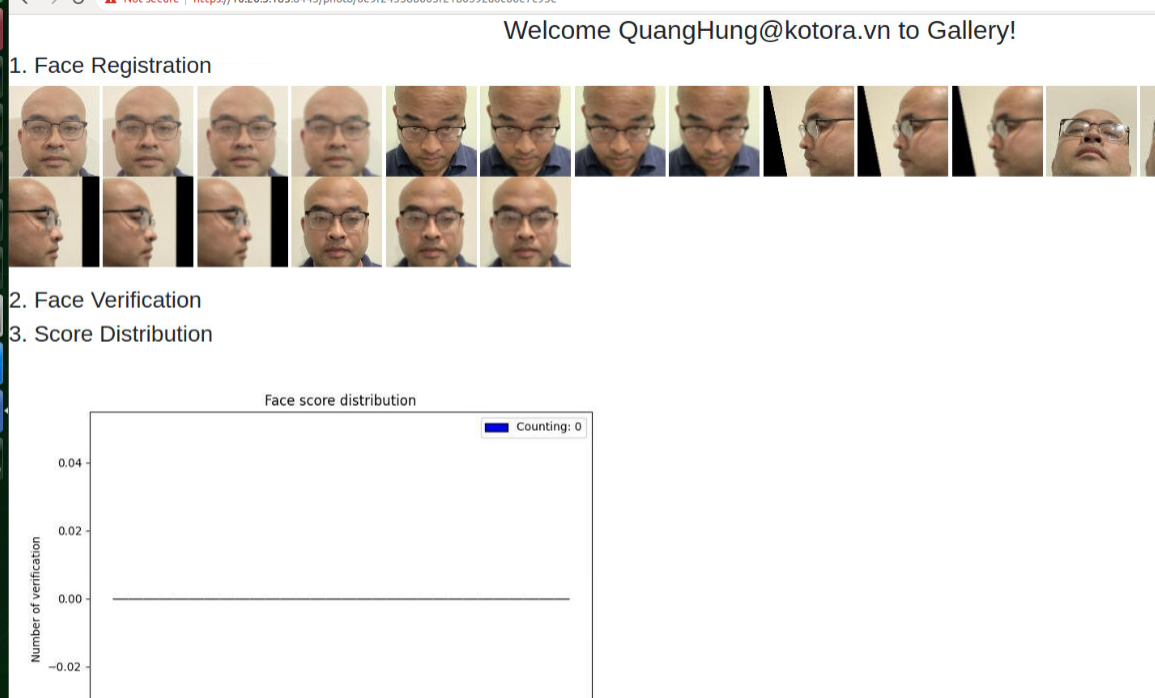
- Truy cập vào url: <https://192.168.0.7:8443/photo> trong đó 192.168.0.7 là địa chỉ của jetson tầng 4:



- Trường hợp muốn xem ảnh đăng ký + ảnh nhận diện, nhập thông tin email và cccd (tương tự như web EKYC) để xem thông tin, phần upload file ko cần chọn file nào, nhấn Submit, ví dụ:

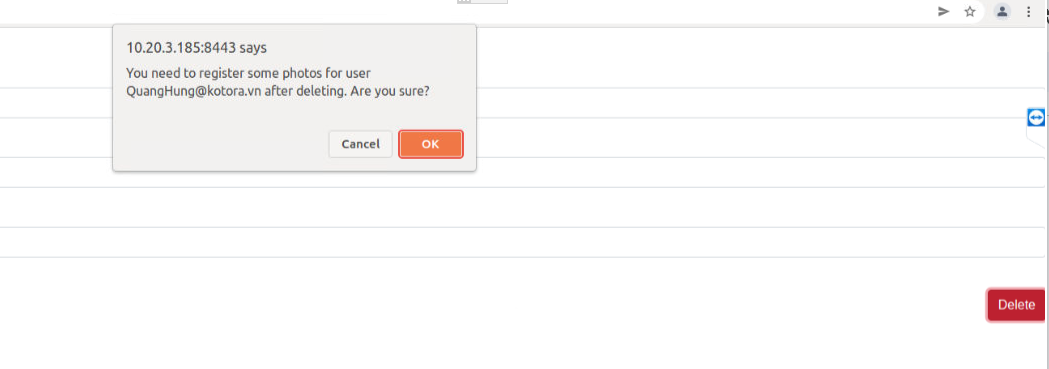


- Sau khi submit ta xem dc các ảnh khuôn mặt đã đăng ký. Phần khuôn mặt nhận diện (verification) sẽ lấy trong 1 ngày gần nhất (có thể config 3 ngày, 7 ngày, …) Nếu trong khoảng thời gian config mà ko truy cập hệ thống thì sẽ ko hiển thị gì. Phần score distribution là phân bố score lúc nhận diện để xem sự thay đổi score như thế nào.



- Để upload thêm ảnh, thì ở phần upload files ta chọn các file ảnh cần upload sau đó nhấn Submit.

- Trường hợp upload ảnh nhầm, cần xóa hết ảnh để update lại thì sau khi điền thông tin (email + CCCD) ta nhấn Delete.



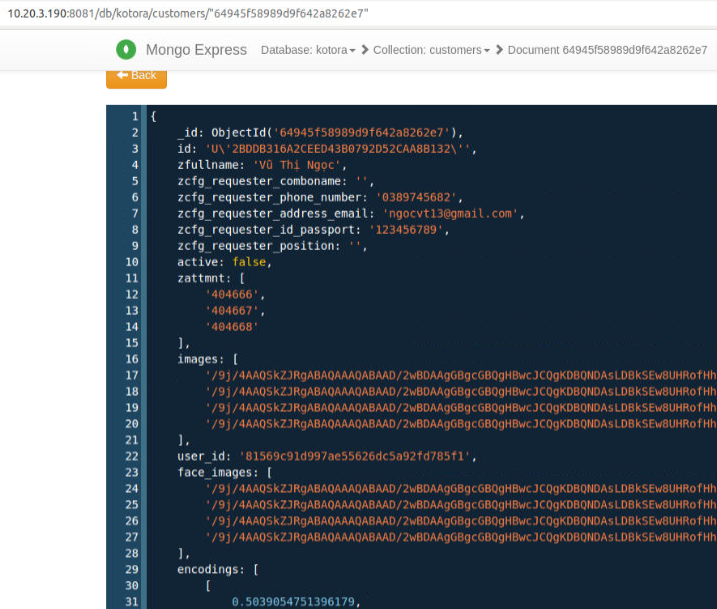
- Confirm trước khi xóa và nhấn OK để xóa.

- Sau khi xóa sẽ xóa sạch toàn bộ ảnh của user đó, chuyển trạng thái user về deactive nhưng ko xóa đi thông tin, sau đó ta cần upload ảnh lại như bình thường.

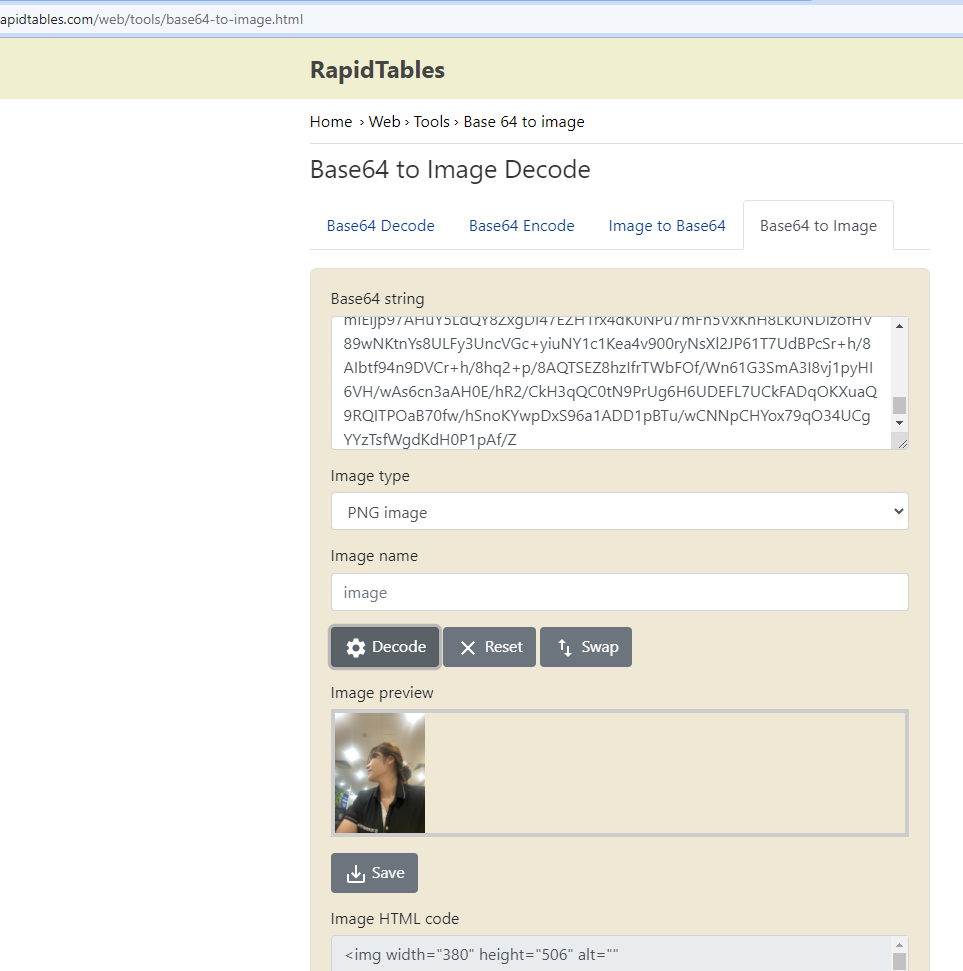
### 3.3. Kiểm tra khả năng đồng bộ dữ liệu từ CASDM:

Sau khi thực hiện bước 3.1, xóa toàn bộ DB Kotora, ta sẽ kiểm tra khả năng đồng bộ dữ liệu (sau 5 phút - config hiện tại) như sau:

Truy cập vào UI mongo tại địa chỉ: <http://192.168.0.4:8081>) trong đó 192.168.0.4 là địa chỉ backend. Sau đó truy cập db kotora, collection customers, ta sẽ thấy một số bản ghi như sau:

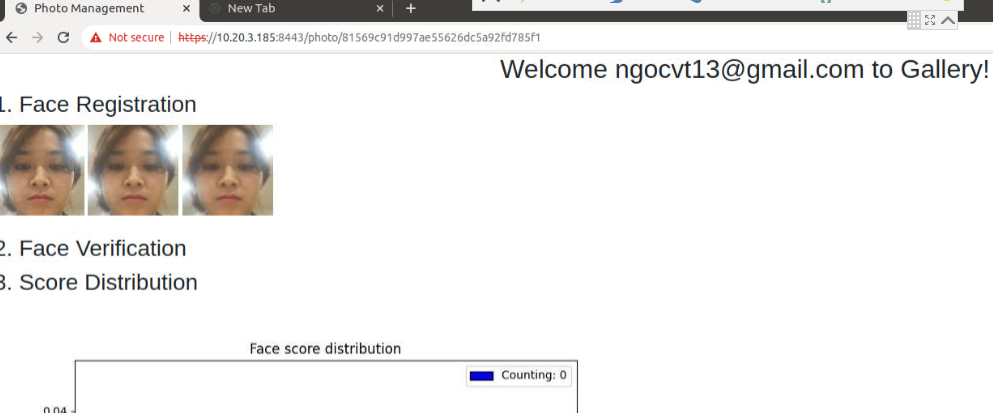


Ở trường images có list các base64 string là ảnh đc đồng bộ từ CASDM sang, user có email là [ngocvt13@gmail.com](mailto:ngocvt13@gmail.com) và cccd là 123456789, ta vào trang: <https://www.rapidtables.com/web/tools/base64-to-image.html> để decode ảnh, xem ảnh này có giống với ảnh giống như lúc đăng ký trên WebEKYC hay WebForm Telehouse hay ko:

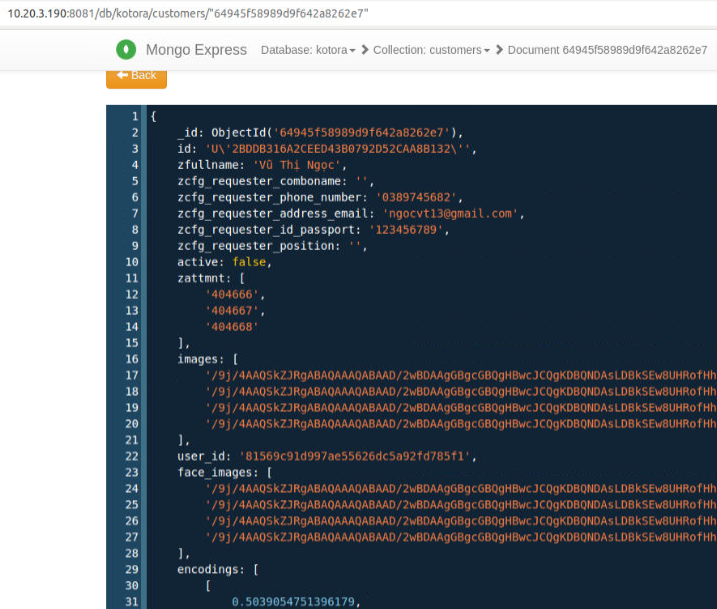


-> Ở đây đã đồng bộ đc ảnh của user

- Test xem các khuôn mặt đăng ký có ổn ko bằng cách thực hiện như bước 3.2, nhập email là [ngocvt13@gmail.com](mailto:ngocvt13@gmail.com) và cccd là 123456789 sau đó submit:



\* Lưu ý:



-> Trong các bản ghi đăng ký sẽ có trường active để check status của user, nếu trạng thái user là active = true thì mới nhận diện được, active = false sẽ không nhận diện được.

Để trạng thái active = true thì email phải có domain là telehouse.vn hoặc thời gian CR lúc đăng ký trên webform Telehouse vẫn còn valid (chưa hết hiệu lực). Nên trong quá trình test đồng bộ, lúc đăng ký webform cần tạo CR đảm bảo thời gian có hiệu lực hoặc email user có domain là telehouse.vn.

## 4. Danh sách API <đang cập nhật>

### 4.1. API đăng ký:

- API đăng ký phục vụ cho việc đăng ký user mới truy cập DC.

* Endpoint (backend): [https://localhost:8999/api/user/pattern](http://localhost:8999/api/user/pattern)
* Method: POST
* Content-Type: application/json
* Body:
* Response:



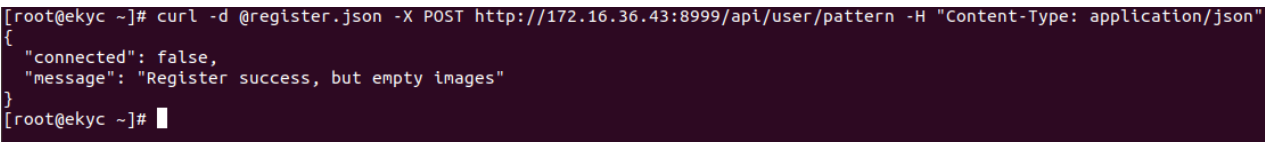
Các trường quan trọng bao gồm: “images”, “zcfg\_requester\_address\_email”, “zcfg\_requester\_id\_passpord” là các trường giúp định danh user. Trong đó trường “images” là một list các ảnh, mỗi ảnh có dạng string base64.

- Test case 1:

* Body:



* Response:



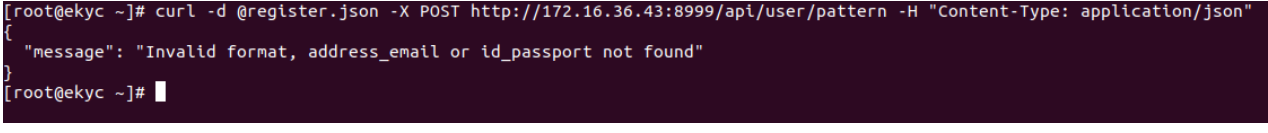
Trong đó: connected là thông báo trạng thái kết nối từ Backend đến đến AI Box, do chưa có kết nối nên báo false, message đăng ký thành công nhưng trường images không chứa ảnh.

- Test case 2:

* Body:



* Response:



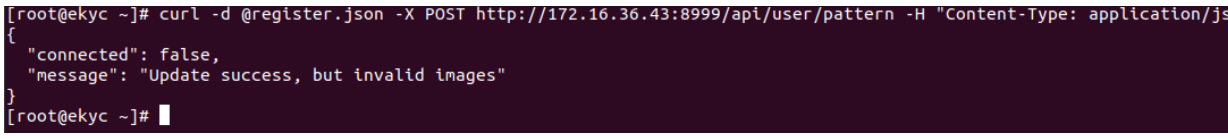
Đăng ký không thành công do thiếu trường thông tin.

- Test case 3:

* Body:



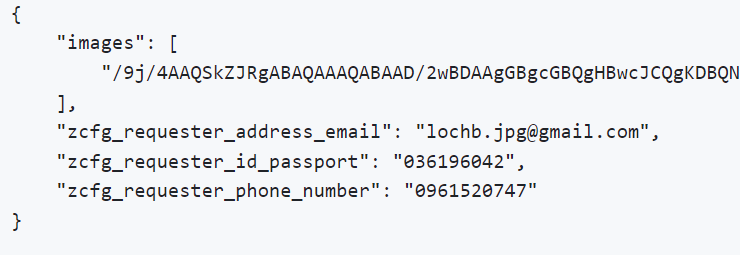
* Response:



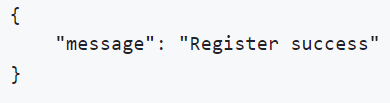
Khi đã tồn tại user, thực hiện update thông tin, nếu request có gửi thêm ảnh thì backend thực hiện thêm ảnh của người đó vào database. Vì vậy có thể đăng ký thông tin trước, đăng ký ảnh sau.

- Test case 4:

* Body:



* Response:

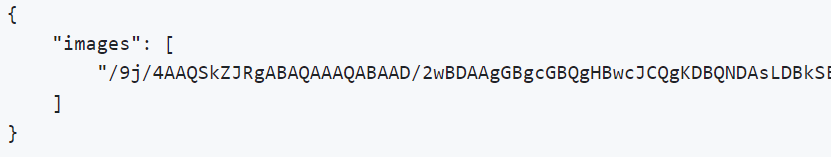


Kiểm tra trên giao diện database ở địa chỉ: <http://localhost:8081> ta thấy thông tin user, ảnh khuôn mặt sau khi được xử lý (face images), và features đặc trưng của khuôn mặt (encodings):



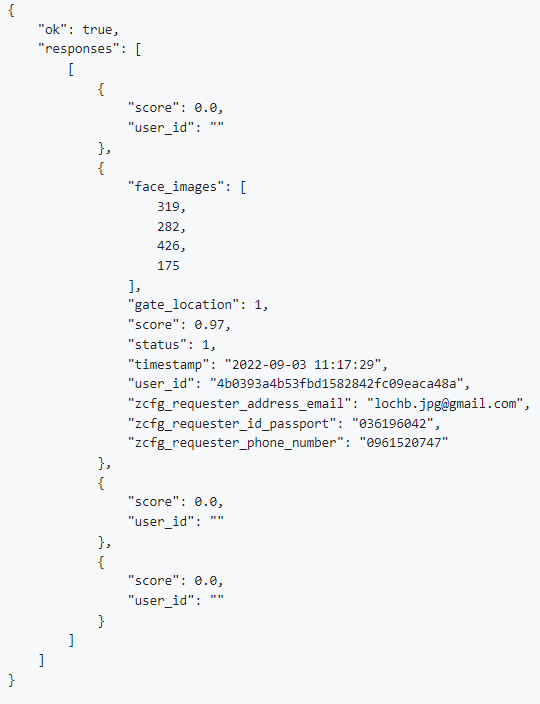
### 4.2. API nhận diện:

* Endpoint (AIBox): [https://localhost:8501/api/user/pattern](http://localhost:8999/api/user/pattern)
* Method: PUT
* Content-Type: application/json
* Body:



Trong đó trường “images” là list các ảnh base64.

* Response:



Response trả về thông tin user nhận diện được trong khung hình, khu vực user đang đứng là cửa nào (gate location), thời gian nhận diện được (timestamp).

Trong khung hình có thể có nhiều người nên response trả về là 1 list, nếu user nào không nhận diện được sẽ trả về score = 0 và user\_id = rỗng.

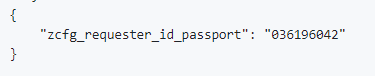
### 4.3. API giám sát vào ra:

* Endpoint (backend): <https://localhost:8999/api/user/monitor>
* Method: POST, GET
* Content-Type: application/json

- Lịch sử truy cập DC được lưu vào database mỗi lần nhận diện được.

Ví dụ muốn xem lịch sử truy cập của một user theo id passport:

* Body:



* Response:



Response trả về lịch sử đăng ký (register\_logs) cũng như lịch sử vào ra DC (verify logs). Từ log này sẽ có thể lấy ra ảnh khuôn mặt lúc nhận diện được, phục vụ cho công tác phân tích dữ liệu, gán nhãn dữ liệu để retrain và cải thiện mô hình.