

## **RAPPORT TP BIOMÉTRIE**

**Projet : authentification d'une personne à l'aide de la  
reconnaissance vocal**

### **MASTER 2 BIO-INFORMATIQUE**

**REALISER PAR :**

**Djennaoui Raouf**

## 1. Conception :

Notre application est divisée en deux parties :

- **L'administrateur :**

Qui a pour rôle d'enrichir le data set en collectant  $n$  échantillon vocal de durée  $t$ , et de créer le model Associer à la voix de chaque individu et d'attribuer le droit d'accès au système (soit l'utilisateur a le droit d'accéder ou non)

- **L'utilisateur :**

Il va fournir un échantillon vocal d'une durée  $t$  et de voir s'il a accès au système ou pas.

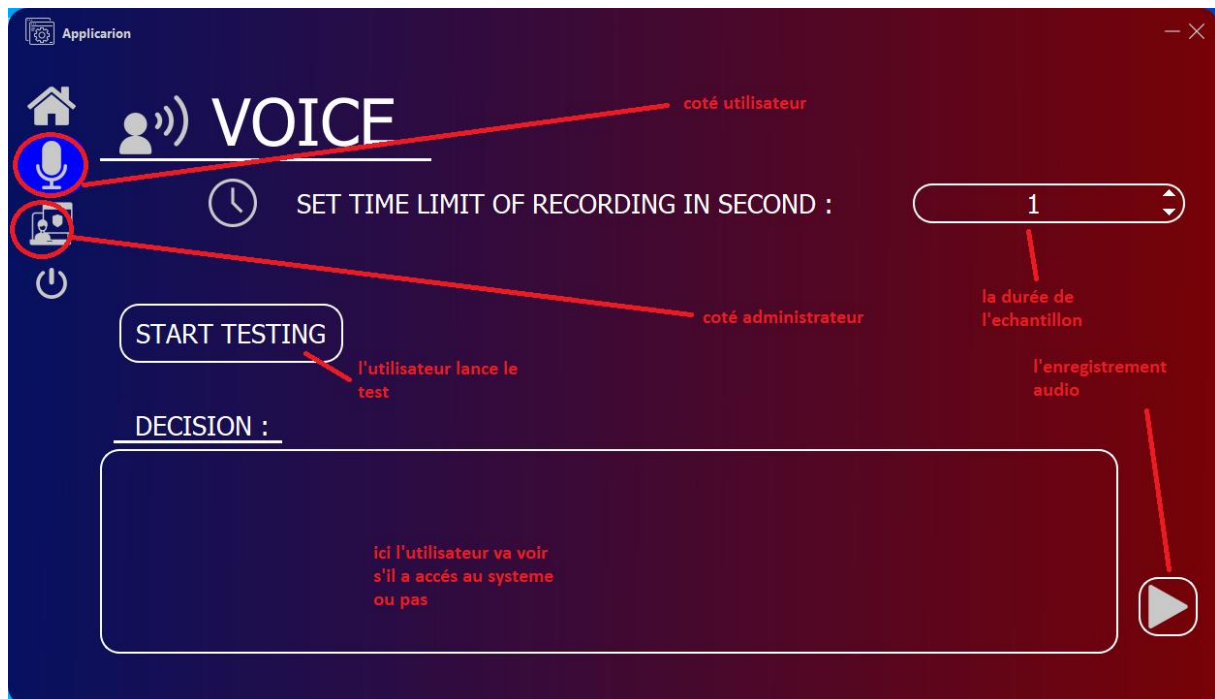


Figure1 : la partie utilisateur

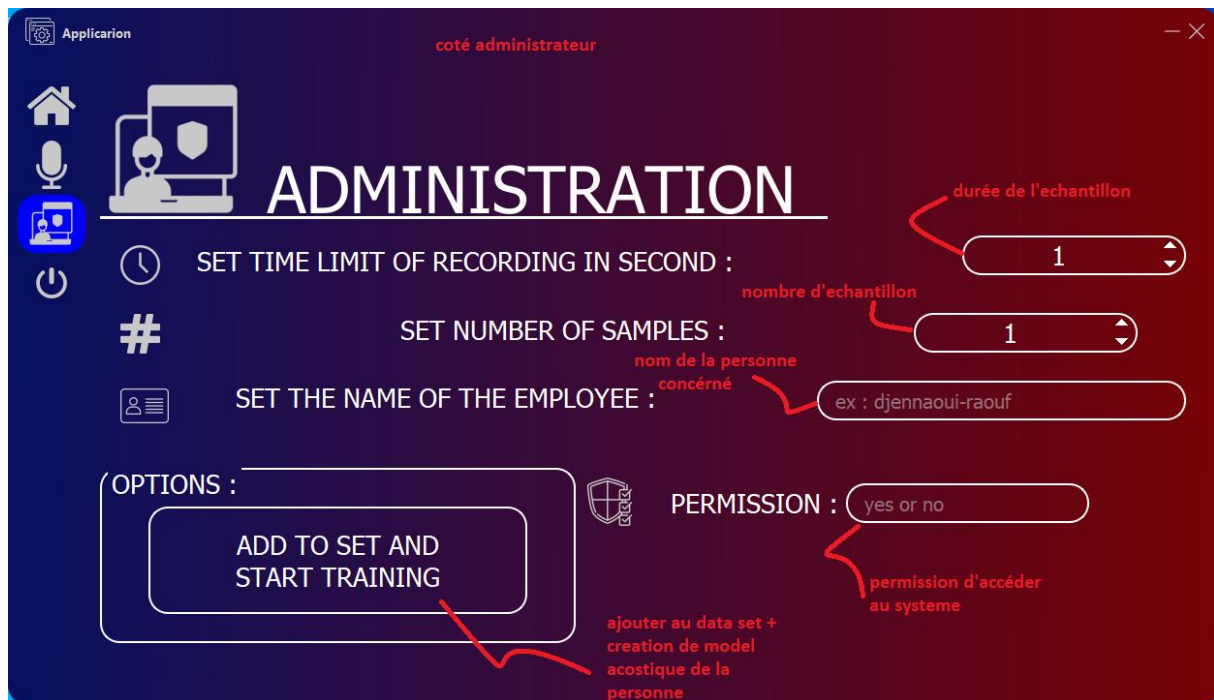


Figure2 : la partie administrateur

## 2. Comment utiliser l'application :

2.1 L'administrateur répertorie la voix d'un utilisateur

2.2 L'utilisateur test s'il est autoriser ou pas

(Plus d'information le jours de la démonstration)

## 3. La procédure :

3.1 Enregistrement de l'échantillon audio à l'aide de Py audio

Soit pour le training (générer le model) soit pour le test

Tout dépend de la valeur de la variable flag

(Code voire la figure ci-dessous)

Les packages utilisés :

```
import os
import wave
import time
import pickle
import pyaudio
import numpy as np
from sklearn import preprocessing
from scipy.io.wavfile import read
import python_speech_features as mfcc
from sklearn.mixture import GaussianMixture
from sklearn import preprocessing
from python_speech_features import mfcc
```

```

from python_speech_features import delta
import re
import warnings
import pymongo

def record(user_name, nbr_samples, time_limit, flag):
    #creation du data set par l administrateur ou teste
    FORMAT = pyaudio.paInt16
    CHANNELS = 2
    RATE = 44100
    CHUNK = 1024
    RECORD_SECONDS = time_limit
    if flag=='train': # enregistrement pour enrichir le data set (repertorier
de nouvelle personnes )
        #nbr_samples : nombre d'échantillon enregistré
        #chaque personne fourni un certain nombre d'échantillon
        l=list()
        for count in range(nbr_samples):
            audio = pyaudio.PyAudio()
            # commencer l'enregistrement
            stream = audio.open(format=FORMAT, channels=CHANNELS, rate=RATE, in
put=True, frames_per_buffer=CHUNK)
            frames = []
            for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
                data = stream.read(CHUNK)
                frames.append(data)
            # fin d'enregistrement
            stream.stop_stream()
            stream.close()
            audio.terminate()

            FILENAME = user_name + str(count) + ".wav"
            # le fichier wav généré va être déplacé dans le repertoire data_se
t

            WAVE_FILE=os.path.join("data_set", FILENAME)

            l.append(FILENAME)
            waveFile = wave.open(WAVE_FILE, 'wb')
            waveFile.setnchannels(CHANNELS)
            waveFile.setsampwidth(audio.get_sample_size(FORMAT))
            waveFile.setframerate(RATE)
            waveFile.writeframes(b''.join(frames))
            waveFile.close()
            # ajouter le fichier de sortie dans la liste des trained files
            # inserer dans MongoDB username et la liste des fichiers audio
            add_to_train_set(user_name, l)

        if flag=='test': # enregistrer la voix pour le test

```

```

audio = pyaudio.PyAudio()
# commencer l'enregistrement
stream = audio.open(format=FORMAT, channels=CHANNELS, rate=RATE, input=
True, frames_per_buffer=CHUNK)
frames = []
for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
    data = stream.read(CHUNK)
    frames.append(data)
# fin d'enregistrement

stream.stop_stream()
stream.close()
audio.terminate()
FILENAME="test.wav"
WAVE_FILE=os.path.join("test_folder",FILENAME)
waveFile = wave.open(WAVE_FILE, 'wb')
waveFile.setnchannels(CHANNELS)
waveFile.setsampwidth(audio.get_sample_size(FORMAT))
waveFile.setframerate(RATE)
waveFile.writeframes(b''.join(frames))
waveFile.close()

```

**Figure 3 : enregistrement audio**

**Remarque :**

User\_name : le nom du locuteur

Nbr\_samples : le nombre d'échantillon audio que le locuteur va fournir

Time\_limit : la durée de l'enregistrement

Flag : soit pour faire un training soit pour faire le test

L'appel se fait :

```

record('nom-prenom', 5, 5, 'train')
record(None, None, 5, 'test')

```

### 3.2 Extraction des features (fonctionnalités) à partir des échantillons audios

En utilisant le package suivant :

```
from python_speech_features import mfcc
from python_speech_features import delta
```

pour faire une identification correcte du locuteur en utilisant le model de mixture gaussien (GMM : **G**aussian **M**ixture **M**odel).

Cette étape consiste à extraire le **C**oefficient **C**epstral de **F**réquence de **M**el (**MFCC**) qui cartographie le signal sur une échelle de mel non linéaire et fournit un vecteur de fonctionnalités

(le code dans la figure ci-dessous)

```
# extraction des features
def extract_features(audio, rate):
    #Extraire les caractéristiques vocales, y compris le coefficient cepstral mel fréquence (MFCC)
    #à partir d'un audio utilisant le module python_speech_features, effectuer Cepstral Mean
    #normalisation (CMS) et le combiner avec les deltas MFCC et le double MFCC
    #Deltas

    mfcc_feature = mfcc(audio,rate, 0.025, 0.01,20,nfft = 1200, appendEnergy = True)
    mfcc_feature = preprocessing.scale(mfcc_feature)
    deltas = delta(mfcc_feature, 2)
    double_deltas = delta(deltas, 2)
    combined = np.hstack((mfcc_feature, deltas, double_deltas))
    return combined
```

Figure 4 : extraction des fonctionnalités

### 3.3 Création du model d'apprentissage en utilisant le model de mixture gaussienne

Nous avons utilisé **MFCC** et **GMM** pour identifier le locuteur. **GMM** va former le model en se basant sur les fonctionnalités extraites précédemment par la fonction suivante :

```
extract_features(audio, rate)
```

Nous avons utilisé le package suivant :

```
from sklearn.mixture import GaussianMixture
from sklearn import preprocessing
```

#### (Le code pour la création du model)

```
def start_training(user_n, permission):

    src = "data_set/"
    dest = "trained_folder/"
    count = 1
    features = np.asarray(())
    pattern = r'[0-9]'

    file_train = get_samples(user_n)
    nbr_samples = len(file_train)

    for path in file_train:

        path = path.strip()
        sr, audio = read(src+path)
        v = extract_features(audio, sr)

        if features.size == 0:
            features = v
        else:
            features = np.vstack((features, v))
        #creation du model pour chaque groupe d echantillon fourni par la pers
        # chaque personne possede un model
        if count == nbr_samples:
            gmm = GaussianMixture(n_components = nbr_samples+1, max_iter = 200
, covariance_type='diag', n_init = 3)
            gmm.fit(features)
            name = re.sub(pattern, '', path.split(".")[0])
            picklefile = name+".gmm"
```

```

    pickle.dump(gmm,open(dest + picklefile,'wb'))
    features = np.asarray(())
    count = 0

    count = count + 1

add_user_to_bdd(user_n,permission)

```

Figure 5 : création du model GMM

### Résulta de la création :








Nom	Modifié le	Type	Taille
 Frozen-Fire.gmm	04/03/2021 16:19	Fichier GMM	16 Ko
 Eric-Hedekar.gmm	04/03/2021 16:24	Fichier GMM	15 Ko
 djennaoui-raouf.gmm	04/03/2021 16:57	Fichier GMM	13 Ko
 David-Bender.gmm	04/03/2021 16:29	Fichier GMM	16 Ko
 Daniel-Heath.gmm	04/03/2021 16:32	Fichier GMM	15 Ko
 Craig-Williamson.gmm	04/03/2021 16:49	Fichier GMM	18 Ko
 Colin-Beckingham.gmm	04/03/2021 16:36	Fichier GMM	15 Ko

Figure 6 : model de chaque locuteur

### Remarque :

Nous avons utilisé une base de données NOSQL MongoDB et le package **Pymongo**

User\_n : nom du locuteur

Permission : (yes ou no) si le locuteur a accès ou non au système



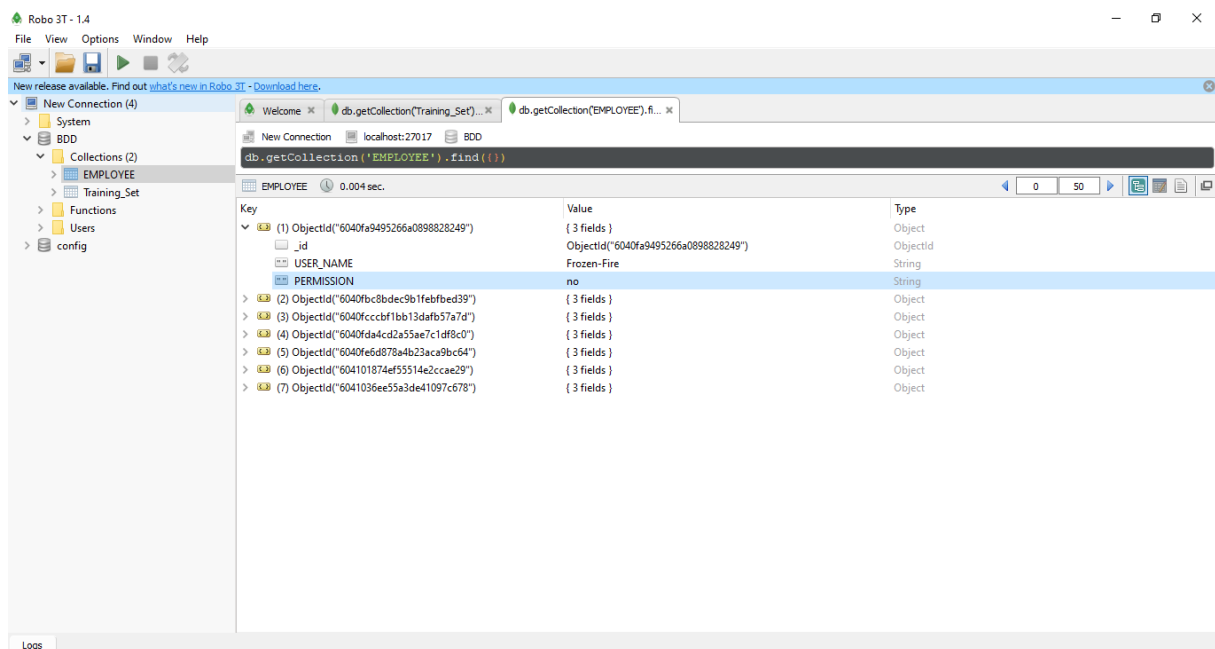


Figure 7 : MongoDB collection des locuteurs et leur permission

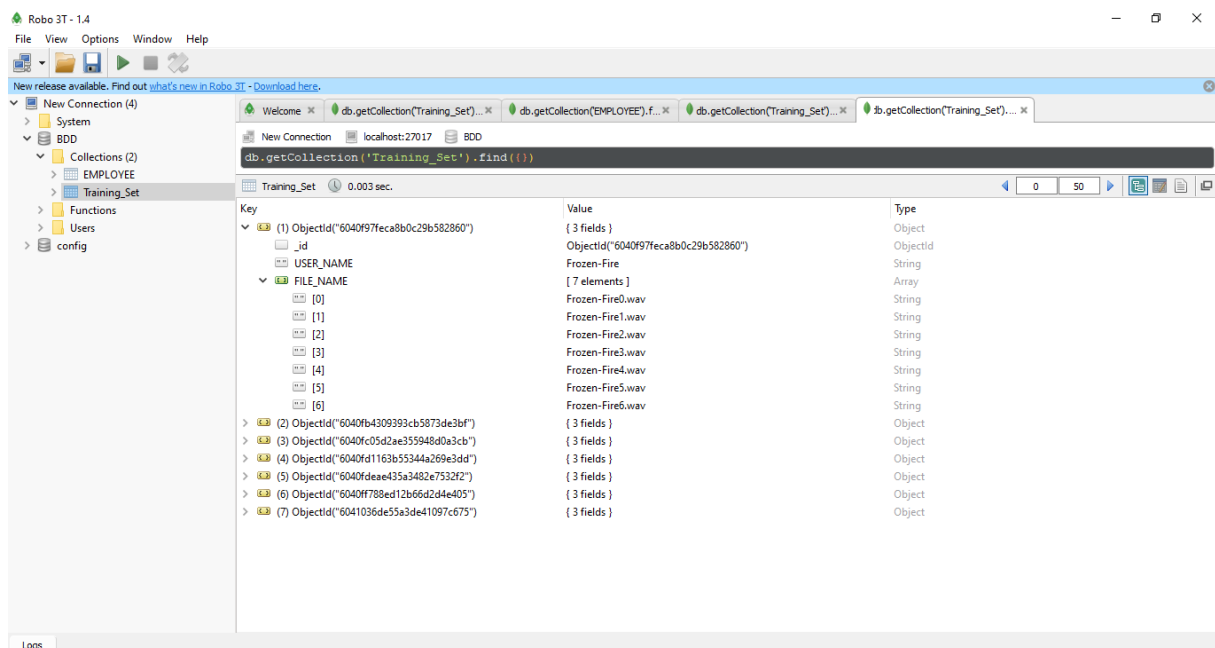


Figure 8 : MongoDB collection des locuteurs et leur fichier audio (DATA-SET)

```
def add_user_to_bdd(user_name,permission): # inserer user_name et permission (
yes or no ) dans la base mongodb
    myclient = pymongo.MongoClient("mongodb://localhost:27017/")
    mydb = myclient["BDD"]
    mycol = mydb["EMPLOYEE"]
    mydict = { "USER_NAME": str(user_name), "PERMISSION": str(permission) }
    mycol.insert_one(mydict)
```

Figure 9 : ajouter le locuteur et sa permission dans la collection EMPLOYEE

```
def add_to_train_set(user_n, audio_file):
    myclient = pymongo.MongoClient("mongodb://localhost:27017/")
    mydb = myclient["BDD"]
    mycol = mydb["Training_Set"]
    train_dict = { "USER_NAME": user_n, "FILE_NAME": audio_file }
    x = mycol.insert_one(train_dict)
```

Figure 10 : ajouter le locuteur et les fichiers audio dans le data set

```
def get_samples(user_n):
    myclient = pymongo.MongoClient("mongodb://localhost:27017/")
    mydb = myclient["BDD"]
    mycol = mydb["Training_Set"]
    for t in mycol.find({"USER_NAME": str(user_n)}, {"_id": 0, "FILE_NAME": 1 }):
        samples=t['FILE_NAME']
    return samples
```

Figure 9 : extraire les échantillons audios pour la création du model GMM

### 3.4 Phase de tests ou phase d'identification du locuteur :

Dans cette phase nous allons calculer le score de similarité du nouveau locuteur (fonctionnalités) avec tous les model créés précédemment

Le score maximal désigne le locuteur comme trouvé (identifié)  
**(Le code ci-dessous)**

```
def start_testing():
    allowed=True
    res=None
    src = "test_folder/"
    model = "trained_folder/"
    test_file = "test.wav"
    user_name=None
    gmm_files = [os.path.join(model,fname) for fname in os.listdir(model) if f
name.endswith('.gmm')]

    models = [pickle.load(open(fname,'rb')) for fname in gmm_files]
    speakers = [fname.split("\\")[-
1].split(".gmm")[0] for fname in gmm_files]

    # tester les fichiers audio
    sr, audio = read(src+test_file)
    v = extract_features(audio, sr)
    l = np.zeros(len(models))

    for i in range(len(models)):
        gmm = models[i] # verification avec chaque model
        scores = np.array(gmm.score(v)) #calcule des scores
        l[i] = scores.sum()

    winner = np.argmax(l) #prendre le meilleur score donc le plus simillair
    res=speakers[winner]
    res=res.split("/)[-1]
    answer=get_user_perm(res)
    if answer==None:
        answer='no'
    return(answer, res) # retourne la permission
```

Figure 11 : code d'identification du locuteur

```
def get_user_perm(user_n): #recuperer la permission de user_n
    allowed=None
```

```

myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["BDD"]
mycol = mydb["EMPLOYEE"]
for t in mycol.find({"USER_NAME":str(user_n)},{ "_id": 0,"USER_NAME": 1, "
PERMISSION": 1 }):
    allowed=t['PERMISSION']
return allowed

```

Figure 12 : code de récupération de la permission du locuteur

### Exemple :

> (6) ObjectId("604101874ef55514e2ccae29")	{ 3 fields }	Object
▼ (7) ObjectId("6041036ee55a3de41097c678")	{ 3 fields }	Object
id	ObjectId("6041036ee55a3de41097c678")	ObjectId
USER_NAME	djennaoui-raouf	String
PERMISSION	yes	String

Figure 13 : ici djennaoui-raouf est autorisé

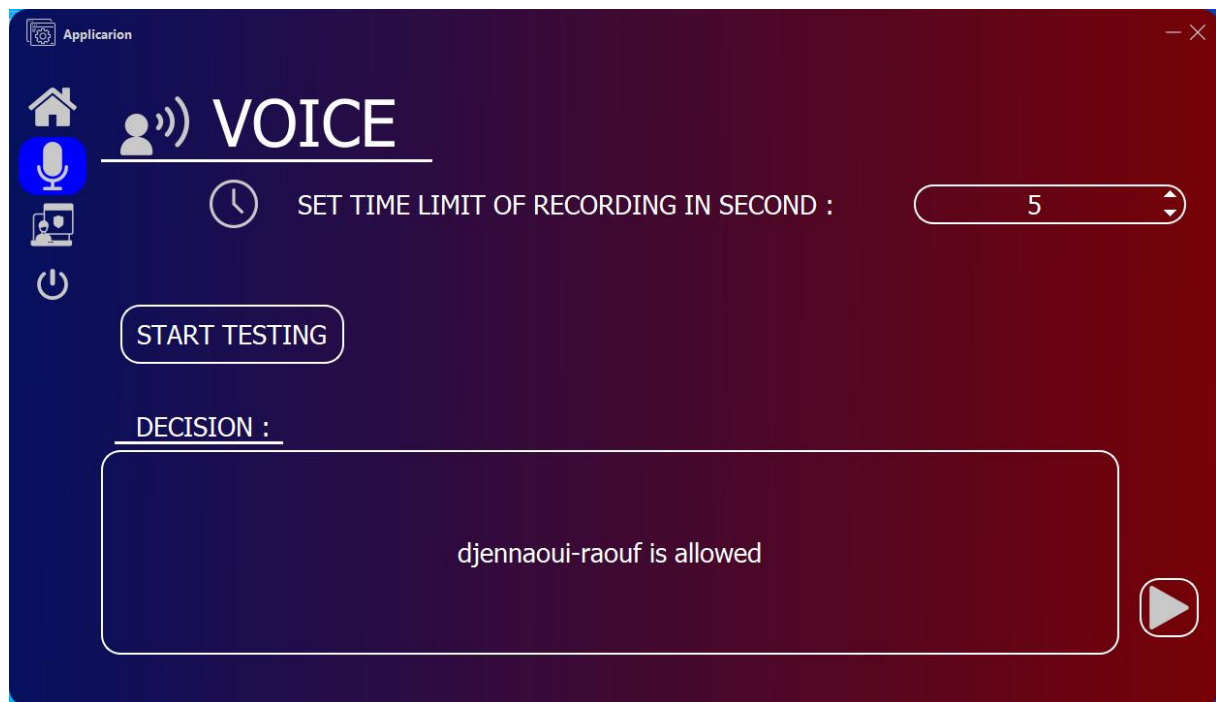


Figure 14 : ici djennaoui-raouf a été identifié via sa voix et affichage de la permission

#### 4. Lancement de l'application :

Exécuter le fichier Application.bat contenant la commande  
suivante :

Python loading.py

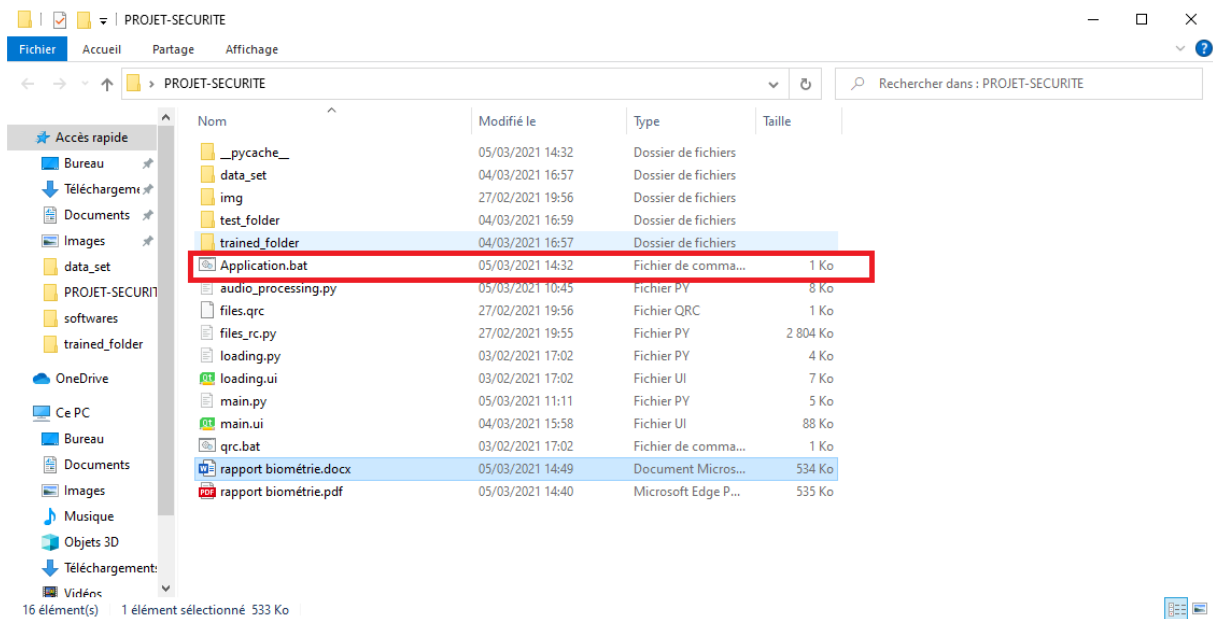


Figure 15 : lancer l'application