FaceRecoApi > __init__.py

```python
# -*- coding: utf-8 -*-
__version__ = '1.0'

from .api import detect_face, face_landmarks, face_encode, compare_faces,
mail_unknown_faces
```

FaceRecoApi > api.py

```python
# import necessary packages
# import necessary packages
import os
import dlib
import numpy as np
import json
import face_recognition_models as frm

# custom exceptions
class NoKnownFaceException(Exception):
    pass

# load necessary face_recognition models

# face detector
face_detector = dlib.get_frontal_face_detector()
face_landmark_model = frm.pose_predictor_model_location()

# 68-point face landmark detector
face_landmark_predictor = dlib.shape_predictor(face_landmark_model)

# 128D face encoder
face_encoder_model = frm.face_recognition_model_location()
face_encoder = dlib.face_recognition_model_v1(face_encoder_model)

# detect faces in an image
def detect_face(frame):
    faces = face_detector(frame, 1)
    return faces


# detect facial landmarks in a detected face
def face_landmarks(frame, faces):
    landmarks = []
    for face in faces:
        face_landmarks = face_landmark_predictor(frame, face)
        landmarks.append(face_landmarks)

    return landmarks
```

```python
# encode the face
def face_encode(frame, landmarks):
    encoded_faces = []
    for landmark in landmarks:
        encoded_face = face_encoder.compute_face_descriptor(frame, landmark, 1)
        encoded_faces.append(encoded_face)

    return encoded_faces

# compare known faces and detected and encoded faces
def compare_faces(detected_faces, encoded_faces):

    """
        :param: detected_faces and encoded_faces
        :return: matched_faces and unmatched_faces

        - known_encoded_faces is a dictionary of known_faces.json file
        - this function will process the encoded faces from known encodings
        - dictionary and list names are self explanatory
    """

    # deserialize data from json file
    known_encoded_faces = {}
    current_directory = os.path.dirname(os.path.abspath('__file__'))
    path = os.path.join(current_directory,'assets', 'known_faces.json')
    with open(path, "r") as json_read:
        known_encoded_faces = json.load(json_read)

    # dictionary and list which will contain matched and unmatched faces
    matched_faces = {}
    unmatched_faces = []
    # a list containing encodings of unmatched faces
    unmatched_encodings = []

    # list which will contain euclidean distance between
    # a detected face and known encoded faces
    compare_distance = []

    # tolerance - how much distance measure will be tolerated
    tolerance = 0.53

    index = 0
    for encoded_face in encoded_faces:
        temp = 0
        fetched_name = ""
        if len(known_encoded_faces) > 0:
            for face in known_encoded_faces:
                fetched_name = face.get('Name')
                encoding = face.get('Encoding')
                compare_distance = np.linalg.norm(encoded_face - encoding, axis=0)

                if compare_distance <= tolerance:
                    temp = 1
                    break
            if temp == 1:       # matched faces found
                matched_faces[fetched_name] = detected_faces[index]
            else:
                unmatched_faces.append(detected_faces[index])
                unmatched_encodings.append(encoded_face)
        index += 1

    return matched_faces, unmatched_faces, unmatched_encodings
```

FaceRecoApi > process_unknown_face.py

```
# mail unknown faces once to a specified email

# this separate program is written separately
# avoid circular import issues

import os
import time
import numpy as np
import json
import cv2 as cv
from faces import get_json_path, add_face

import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage

# Example usage
sender_email = "facereco3@gmail.com"
sender_password = "elvcnipsgokhjvuo"
receiver_email = "shyamendrahazra@gmail.com"
subject = "Sending unknown faces via email"
message = "Unknown faces found in webcam."
def mail_unknown_faces(frame):
    print("Inside mail unknown")
    current_directory = os.path.dirname(os.path.abspath('__file__'))
    output_filename = os.path.join(current_directory, 'assets'
                              ,'Unknown_Faces', 'image_')
    output_filename = output_filename + str(time.time()) + ".jpg"

    if len(frame) > 0:
        _, image_data = cv.imencode('.jpg', frame)
        with open(output_filename, "wb") as image_file:
            image_file.write(image_data)

        with open(output_filename, "rb") as image_file:
            image = image_file.read()

        # Create a multipart message
        msg = MIMEMultipart()
        msg["From"] = sender_email
        msg["To"] = receiver_email
        msg["Subject"] = subject

        # Add the message body
        msg.attach(MIMEText(message, "plain"))

        # Load and attach the image
        img = MIMEImage(image)
        img.add_header("Content-Disposition", "attachment",
filename="unknown.jpg")
        msg.attach(img)

        # Establish an SMTP connection
        with smtplib.SMTP("smtp.gmail.com", 587) as server:
            server.starttls()
            print("Logging in : (" + sender_email + ")...")
            server.login(sender_email, sender_password)
            print("Logged in to server.")
            print("Sending mail to: " + receiver_email + "...")
            server.sendmail(sender_email, receiver_email, msg.as_string())
            print("Mail sent successfully.")


def process_unknown_faces(frame, unmatched_faces, unmatched_encodings):
    """
    :param frame: the image from where unknown faces will be found
    :param unmatched_faces: the dlib rectangle coordinates of
        unmatched/unknown faces
    :param unmatched_encodings: enconding of the unmatched/unknown faces
    """
    # check if the unknown face is already known
    path = get_json_path('u')
    with open(path, "r") as json_read:
        unknown_face_data = json.load(json_read)
    for face in unmatched_encodings:
        for face_data in unknown_face_data:
            for _, encoding in face_data.items():
                compare_distance = np.linalg.norm(encoding - face, axis=0)
                if compare_distance <= 0.6:
                    return

        # add the unknown faces in the unknown_faces.json file
        face = face.tolist()
        add_face(face, 'u')

        # mail the unknown faces
        for face in unmatched_faces:
            cv.rectangle(frame,
                         (face.left(), face.top()),
                         (face.right(), face.bottom()),
                         (0, 255, 0), 2)
            # print(type(frame))
            mail_unknown_faces(frame)
```