In [1]:

```
import numpy as np
import os
import cv2
import time
import face_recognition
import face_recognition as fr
import glob
import logging
```

In [2]:

```
def get_face_embeddings_from_image(image, convert_to_rgb=False):
    """
    Take a raw image and run both the face detection and face embedding model on it
    """
    # Convert from BGR to RGB if needed
    if convert_to_rgb:
        image = image[:, :, ::-1]
    # run the face detection model to find face locations
    face_locations = face_recognition.face_locations(image)
    # run the embedding model to get face embeddings for the supplied locations
    face_encodings = face_recognition.face_encodings(image, face_locations)
    return face_locations, face_encodings
```

In [3]:

```
def setup_database():

    """
    Load reference images and create a database of their face encodings
    """
    database = {}
    for folders in glob.glob(os.path.join(IMAGES_PATH, '*')):
        for filename in glob.glob(os.path.join(folders, '*.jpg')):
            # load image
            image_rgb = face_recognition.load_image_file(filename)
            # use the name in the filename as the identity key
            identity = os.path.splitext(os.path.basename(filename))[0]
            # get the face encoding and link it to the identity
            locations, encodings = get_face_embeddings_from_image(image_rgb)
            database[identity] = encodings[0]

    return database
```

In [8]:

```
IMAGES_PATH='E:\\DATA2\\'
```

In [11]:

```
database = setup_database()
len(database)
```

Out[11]:

```
50
```

In [6]:

```
known_face_names=[]
for folders in os.listdir(IMAGES_PATH):
    for images in os.listdir(IMAGES_PATH + folders):
        known_face_names.append(folders)
len(known_face_names)
```

Out[6]:

316

In [7]:

```
face_locations = []
face_encodings = []
face_names = []
process_this_frame = True

video_capture = cv2.VideoCapture(0)

while True:
    # Grab a single frame of video
    ret, frame = video_capture.read()
    # 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 face_recognition us
es)
    rgb_small_frame = small_frame[:, :, ::-1]
    # Only process every other frame of video to save time

    known_face_encodings=list(database.values())
    if process_this_frame:
        # 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.api.compare_faces(known_face_encodings, face_encoding, toler
ance =0.6)
            name = "Unknown"
            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]
            face_names.append(name)
    process_this_frame = not process_this_frame
    # 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
        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)
        font = cv2.FONT_HERSHEY_DUPLEX
        cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)
    # Display the resulting image
    cv2.imshow('Video', frame)
    # Hit 'q' on the keyboard to quit!
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
# Release handle to the webcam
# video_capture.release()
video_capture.release()
cv2.destroyAllWindows()
```

In [ ]:

```
video_capture.release()
```

In [ ]:

```

```