In [13]:
```
import datetime
import cv2
from pygame import mixer
import tensorflow as tf
from tensorflow.keras.models import load_model
import numpy as np
import sqlite3
```

In [14]:
```
class Time:

    def __init__(self, day, name, time):
        self.day=day
        self.name = name
        self.time = time
```

In [15]:
```
def update_clock(system_t):
    os = datetime.datetime.now()
    system_t = os.strftime("%H:%M:%S")
    # system_d = os.strftime('%A')
    return system_t
```

In [16]:
```
def update_day(system_d):
    day= datetime.datetime.now()
    system_d = day.strftime('%A')
    return system_d
```

In [17]:
```
def name(system_t):

    basename = "mylog"
#     suffix = datetime.datetime.now().strftime("%H%M%S")
    filename = "_".join([basename, update_clock(system_t)])
    return filename
```

```
In [23]:

con = sqlite3.connect('Time99.db')
c = con.cursor()

# c.execute("""CREATE TABLE timestamp(day text,name text, time integer)""")
def insert_timestamp(n):
    con = sqlite3.connect('Time99.db')
    c = con.cursor()
    c.execute("INSERT INTO timestamp (day, name, time) VALUES (?,?,?)", (n.day, r
    con.commit()
    con.close()


c.execute("SELECT * FROM timestamp ")
# print(c.fetchall())
for i in range(30):
    print("\n")
    print(c.fetchone())
con.close()
```

```


('Thursday', 'Driver Unknown', '03:36:15')


('Thursday', 'Driver Unknown', '03:36:16')


('Thursday', 'Driver Unknown', '03:36:16')


('Thursday', 'Driver Unknown', '03:36:17')


('Thursday', 'Driver Unknown', '03:36:18')


('Thursday', 'Driver Unknown', '03:36:19')


('Thursday', 'Driver Unknown', '03:36:23')


('Thursday', 'Driver Unknown', '03:36:24')


('Thursday', 'Driver Unknown', '03:36:24')


('Thursday', 'Driver Unknown', '03:36:25')


('Thursday', 'Driver Unknown', '03:36:26')


('Thursday', 'Driver Unknown', '03:36:29')
```

```
('Thursday', 'Driver Unknown', '03:36:30')

('Thursday', 'Driver Unknown', '03:36:31')

('Thursday', 'Driver Unknown', '03:36:31')

('Thursday', 'Driver Unknown', '03:36:32')

None

None

None

None

None

None

None

None

None

None

None

None

None

None
```

In [19]:

```
mixer.init()
sound = mixer.Sound('C:/Users/visha/OneDrive/Desktop/ES/mini project/mixkit-vinta
faceCascade = cv2.CascadeClassifier('C:/Users/visha/OneDrive/Desktop/ES/mini proj
eyeCascade = cv2.CascadeClassifier('C:/Users/visha/OneDrive/Desktop/ES/mini proje
model = load_model(r'D:\eyes\models\model.h5')
count=0
ostime = datetime.datetime.now()

system_time=ostime.strftime('%H:%M:%S')
system_day=ostime.strftime('%A')
```

In [ ]:

```
In [20]:

cap = cv2.VideoCapture(0)
while(1):

    ret, img = cap.read()
    if ret:
        frame = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = faceCascade.detectMultiScale(frame, 1.1, 5)
        if len(faces) > 0:
            for (x, y, w, h) in faces:
                cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
            frame_tmp = img[faces[0][1]:faces[0][1] + faces[0][3], faces[0][0]:fa
            frame = frame[faces[0][1]:faces[0][1] + faces[0][3], faces[0][0]:face
            eyes = eyeCascade.detectMultiScale(frame, scaleFactor=1.1, minNeighbo
            for (ex, ey, ew, eh) in eyes:

                eye = frame_tmp[ey:ey + eh, ex:ex + ew]
                eye = cv2.resize(eye, (80, 80))
                eye = eye / 255
                eye = eye.reshape(80, 80, 3)
                eye = np.expand_dims(eye, axis=0)

                prediction = model.predict(eye)

            if prediction[0][0] < 0.30:

                count+=1
                if(count==4):
                    # Saved closed eye image in folder
#                     nameo=name(system_time);
#                     print(nameo)
                    cv2.imwrite(r"C:\\Users\\visha\\OneDrive\\Desktop\\New folder
                    count = 0

                    print('WARNING: eyes closed')
                    a = Time(update_day(system_day), 'Driver Unknown', update_clo
                    # created database for saving timestamps
                    insert_timestamp(a)
                    print(update_clock(system_time))

                    cv2.putText(frame_tmp, "WARNING!!!", (10, 30), cv2.FONT_HERSH
                    cv2.putText(frame_tmp, update_clock(system_time),(00,200),cv2

                    sound.play(0, 5, 0)

            else:
                count = 0
                cv2.putText(frame_tmp, "ALL GOOD", (10, 30), cv2.FONT_HERSHEY_SIM
                cv2.putText(frame_tmp, update_clock(system_time), (00, 200), cv2.


            frame_tmp = cv2.resize(frame_tmp, (400, 400), interpolation=cv2.INTER

            cv2.imshow('Face Recognition', frame_tmp)
        waitkey = cv2.waitKey(1)
        if cv2.waitKey(33) & 0xFF == ord('q'):
            cap.release()
            cv2.destroyAllWindows()
            break
```

```
1/1 [==============================] - 3s 3s/step
1/1 [==============================] - 0s 65ms/step
1/1 [==============================] - 0s 68ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 64ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 64ms/step
1/1 [==============================] - 0s 66ms/step
1/1 [==============================] - 0s 68ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 67ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 66ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 74ms/step
1/1 [==============================] - 0s 68ms/step
1/1 [==============================] - 0s 69ms/step
```

```
In [ ]:
```