

main.py:

```

import cv2

import dlib

import numpy as np

import time

# --- SETTINGS ---

EYE_AR_THRESH = 0.23 # Threshold for eye closure

MOUTH_AR_THRESH = 0.5 # Threshold for yawning

SLEEP_TIME_LIMIT = 1.5 # Seconds before Sleep Alert

GREEN_BUFFER = [] # Buffer for heart rate

HEART_RATE = 0

# Parameters for Blink Analysis

BLINK_COUNT = 0

BLINK_START_TIME = time.time()

ALCOHOL_WARNING = False # Low blink rate

STRESS_WARNING = False # High blink rate

is_eye_closed = False

# Initialize dlib

detector = dlib.get_frontal_face_detector()

try:

predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")

except:

print("Error: .dat model file not found!")

exit()

def get_ear(eye_points):

```

```

v1 = np.linalg.norm(eye_points[1] - eye_points[5])
v2 = np.linalg.norm(eye_points[2] - eye_points[4])
h = np.linalg.norm(eye_points[0] - eye_points[3])
return (v1 + v2) / (2.0 * h)

cap = cv2.VideoCapture(0)
eye_closed_timer = None

while cap.isOpened():
    success, frame = cap.read()

    if not success: break

    h, w, _ = frame.shape

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    rects = detector(gray, 0)

    status = "STATUS: OK"

    color = (0, 255, 0)

    health_msg = ""

    msg_color = (255, 255, 255)

    for rect in rects:

        shape = predictor(gray, rect)

        pts = np.array([[p.x, p.y] for p in shape.parts()])

        # 1. EYE ANALYSIS & BLINK COUNTER

        ear = (get_ear(pts[36:42]) + get_ear(pts[42:48])) / 2.0

        if ear < EYE_AR_THRESH:

            if not is_eye_closed:

                is_eye_closed = True

                BLINK_COUNT += 1

            if eye_closed_timer is None:

```

```

eye_closed_timer = time.time()

if time.time() - eye_closed_timer >= SLEEP_TIME_LIMIT:

    status = "!!! DANGER: SLEEP !!!"

    color = (0, 0, 255)

else:

    is_eye_closed = False

    eye_closed_timer = None

# 2. ADVANCED BLINK RATE ANALYSIS (every 30 seconds)

current_time = time.time()

if current_time - BLINK_START_TIME > 30:

    # Low blink rate (<4) -> Intoxication

    ALCOHOL_WARNING = True if BLINK_COUNT < 4 else False

    # High blink rate (>15) -> Stress / Fatigue

    STRESS_WARNING = True if BLINK_COUNT > 15 else False

    BLINK_COUNT = 0

    BLINK_START_TIME = current_time

    # Logic for health messages

    if ALCOHOL_WARNING:

        health_msg = "WARNING: POSSIBLE INTOXICATION"

        msg_color = (255, 0, 255) # Magenta

    elif STRESS_WARNING:

        health_msg = "WARNING: HIGH STRESS / EYE FATIGUE"

        msg_color = (0, 255, 255) # Yellow/Cyan

# 3. DISTRACTION & YAWNING (Simplified)

nose_x = pts[30][0]

if abs(nose_x - rect.center().x) > 20:

```

```

status = "WARNING: LOOK AT THE ROAD!"

color = (0, 165, 255)

mar = np.linalg.norm(pts[62] - pts[66]) / np.linalg.norm(pts[60] - pts[64])

if mar > MOUTH_AR_THRESH and status == "STATUS: OK":

    status = "YAWNING DETECTED"

    color = (255, 255, 0)

# 4. HEART RATE (rPPG)

roi_y1, roi_y2 = max(0, rect.top() - 25), max(0, rect.top() - 5)

roi_x1, roi_x2 = rect.center().x - 10, rect.center().x + 10

forehead_roi = frame[roi_y1:roi_y2, roi_x1:roi_x2]

if forehead_roi.size > 0:

    GREEN_BUFFER.append(np.mean(forehead_roi[:, :, 1]))

if len(GREEN_BUFFER) > 150:

    GREEN_BUFFER.pop(0)

data = np.array(GREEN_BUFFER)

data = (data - np.mean(data)) / (np.std(data) + 1e-6)

fft = np.abs(np.fft.rfft(data))

freqs = np.fft.rfftfreq(len(data), 1 / 30)

idx = np.where((freqs >= 0.8) & (freqs <= 2.2))[0]

if len(idx) > 0:

    HEART_RATE = freqs[idx[np.argmax(fft[idx])]] * 60

# VISUALIZATION

cv2.rectangle(frame, (rect.left(), rect.top()), (rect.right(), rect.bottom()), color, 2)

cv2.putText(frame, status, (20, 45), cv2.FONT_HERSHEY_SIMPLEX, 0.9, color, 3)

cv2.putText(frame, f"Pulse: {int(HEART_RATE)} BPM", (20, 80),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 0), 2)

if health_msg:

```

```
cv2.putText(frame, health_msg, (20, 115), cv2.FONT_HERSHEY_SIMPLEX, 0.6,  
msg_color, 2)  
  
cv2.imshow('Driver Monitoring System v4.0', frame)  
  
if cv2.waitKey(1) & 0xFF == ord('q'): break  
  
cap.release()  
  
cv2.destroyAllWindows()
```

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод.
------------	--------------	------

