Drowsiness Detection system

Overview of the project:

- ➔ WHAT?
  - ◆ Computer vision application to detect if the person is sleeping or not.
  - ◆ The data of the people who are sleeping will be stored.
  - ◆ The source of the data is IP Camera, Webcam, Online video.
  - ◆ It is a Web application which can be accessed over local network/WiFi.
  - ◆ An IP camera is a wireless camera. The purpose is to send footage wirelessly to the local.wifi network and later processed via application (also connected to the same local/wifi network).

- ➔ WHY?
  - ◆ Applications can be done at
    - **Automotive:** Driver monitoring to prevent fatigue-related accidents.
    - **Workplace Safety:** Monitor machine operators and night-shift workers.
    - **Transportation:** Track truckers, pilots, and train operators for alertness.
    - **Healthcare:** Support sleep studies and patient monitoring.
    - **Consumer Electronics:** Integrate with smartphones, laptops, and gaming devices.
    - **Public Safety:** Ensure alertness in surveillance and emergency response teams.
  - ◆ It showcases Python programming skills, web development skills, ML knowledge.

- ➔ HOW?
  - ◆ Backend
    - Connection with IP Camera: OpenCV
    - Face detection: dlib
    - Eyes detection: numpy, scipy.spatial.distance
    - Save data: OpenCV
  - ◆ Frontend
    - Web application: streamlit

- **Stages of project:**
  - Overview and OpenCV
  - Eye detection
  - Frontend

- Create virtual environment
  - Go to C drive – New folder (name it my projects)-- New folder(name it facemasks). Then copy the folder location.
  - Then go to the Scripts folder, go to Python 311 or 312, then type cmd in the address bar.
  - Then type ***python -m venv <Paste the folder location>*** and hit Enter.
  - Then go to the Scripts folder of the facemask and type cmd in the address bar.
  - The type activate and the virtual environment will be activated.

- Install openCV
  - On the command screen, type ***pip install opencv*** and hit enter.

- OPen IDLE power shell:

- Go to Start and open IDLE Powershell. A new window will open.
- Then click on File>New>Name the file as backend and save it the location under C:\myprojects\facemask\Scripts.
- Then the coding begins. You can find the coding in the backend.py file.

Face detection

```
import cv2
import dlib
import numpy as np

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

vid = cv2.VideoCapture(0)
while vid.isOpened():
    flag, frame = vid.read()
    if not flag:
        break

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

    for face in faces:
        landmarks = predictor(gray, face)
        left_eye = np.array([[landmarks.part(i).x, landmarks.part(i).y] for i in range(36, 42)])
        right_eye = np.array([[landmarks.part(i).x, landmarks.part(i).y] for i in range(42, 48)])
        cv2.polylines(frame, [left_eye], True, (0, 255, 0), 1)
        cv2.polylines(frame, [right_eye], True, (0, 255, 0), 1)

    cv2.imshow("Drowsiness Detection", frame)

    if cv2.waitKey(5) & 0xFF == ord('x'):
        break
vid.release()
cv2.destroyAllWindows()
```

Drowsiness detection on webcam

```
import cv2
import dlib
import numpy as np
from scipy.spatial import distance as dist

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

def calculate_ear(eye):
    A = dist.euclidean(eye[1], eye[5])  # Vertical distance
    B = dist.euclidean(eye[2], eye[4])  # Vertical distance
    C = dist.euclidean(eye[0], eye[3])  # Horizontal distance
    ear = (A + B) / (2.0 * C)
    return ear

vid = cv2.VideoCapture(0)
while vid.isOpened():
    flag, frame = vid.read()
    if not flag:
        break

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

    for face in faces:
        landmarks = predictor(gray, face)
        left_eye = np.array([[landmarks.part(i).x, landmarks.part(i).y] for i in range(36, 42)])
        right_eye = np.array([[landmarks.part(i).x, landmarks.part(i).y] for i in range(42, 48)])

        left_ear = calculate_ear(left_eye)
        right_ear = calculate_ear(right_eye)
        avg_ear = (left_ear + right_ear) / 2.0

        eye_closure_percentage = (1 - avg_ear / 0.3) * 100
        cv2.polylines(frame, [left_eye], True, (0, 255, 0), 1)
        cv2.polylines(frame, [right_eye], True, (0, 255, 0), 1)

        cv2.putText(frame, f"{eye_closure_percentage:.1f}%",
                    (left_eye[0][0], left_eye[0][1] - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255), 2)

        if eye_closure_percentage > 25:
            cv2.putText(frame, "DROWSINESS ALERT!",
                        (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0, 0, 255), 3)
```

```
    cv2.imshow("Drowsiness Detection", frame)
    if cv2.waitKey(5) & 0xFF == ord('x'):
        break

vid.release()
cv2.destroyAllWindows()
```

Drowsiness detection on IP camera

```
import cv2
import dlib
import numpy as np
from scipy.spatial import distance as dist

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

def calculate_ear(eye):
    A = dist.euclidean(eye[1], eye[5])  # Vertical distance
    B = dist.euclidean(eye[2], eye[4])  # Vertical distance
    C = dist.euclidean(eye[0], eye[3])  # Horizontal distance
    ear = (A + B) / (2.0 * C)
    return ear

vid = cv2.VideoCapture(“IP camera URL/video”)
while vid.isOpened():
    flag, frame = vid.read()
    if not flag:
        break

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

    for face in faces:
        landmarks = predictor(gray, face)
        left_eye = np.array([[landmarks.part(i).x, landmarks.part(i).y] for i in range(36, 42)])
        right_eye = np.array([[landmarks.part(i).x, landmarks.part(i).y] for i in range(42, 48)])
        left_ear = calculate_ear(left_eye)
        right_ear = calculate_ear(right_eye)
        avg_ear = (left_ear + right_ear) / 2.0

        eye_closure_percentage = (1 - avg_ear / 0.3) * 100
        cv2.polylines(frame, [left_eye], True, (0, 255, 0), 1)
        cv2.polylines(frame, [right_eye], True, (0, 255, 0), 1)

        cv2.putText(frame, f"{eye_closure_percentage:.1f}%",
                    (left_eye[0][0], left_eye[0][1] - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255), 2)

        if eye_closure_percentage > 25:
            cv2.putText(frame, "DROWSINESS ALERT!",
                        (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0, 0, 255), 3)

    cv2.imshow("Drowsiness Detection", frame)
```

```
    if cv2.waitKey(5) & 0xFF == ord('x'):
        break

vid.release()
cv2.destroyAllWindows()
```

Streamlit program

```
import streamlit as st

st.title("Drowsiness Detection System")
choice=st.sidebar.selectbox("My Menu",("Home", "IP Camera", "Webcam", "Video"))

if (choice=="Home"):
    st.image("C:\myprojects\sentiment\Scripts\homepage.png")
    st.header("Welcome to Drowsiness Detection System")
    st.write("Our Drowsiness Detection System is an AI-powered application designed to monitor eye closure levels in real-time using a webcam. It detects signs of drowsiness by analyzing facial landmarks and calculating the Eye Aspect Ratio (EAR).")

    st.subheader("How It Works?")
    st.write("✔ The system detects the face and eyes using a deep learning-based model.")
    st.write("✔ It calculates the percentage of eye closure based on eye landmarks.")
    st.write("✔ If the eye closure percentage exceeds 50%, a drowsiness alert appears on the screen.")

    st.subheader("Key Features")
    st.write("✔ Real-time Monitoring 🖥️ – Detects drowsiness instantly.")
    st.write("✔ Visual Alert ⚠️ – Displays a red warning message when drowsy.")
    st.write("✔ Non-Intrusive 📷 – Works using a simple webcam.")
    st.write("✔ Customizable Thresholds 🎛️ – Can be fine-tuned for different users.")

    st.write("This system is ideal for drivers, machine operators, students, and professionals who need to stay alert while working!")

    st.write("Would you like to add an alarm sound feature for better alerts? 🔊🚀")
```

Complete Drowsiness detection System Code

```
import streamlit as st
import cv2
import dlib
import numpy as np
import tempfile
from scipy.spatial import distance as dist

st.title("Drowsiness Detection System")
choice=st.sidebar.selectbox("My Menu",("Home", "IP Camera", "Webcam", "Video"))

if (choice=="Home"):
    st.image("C:\myprojects\drowsiness1\Scripts\drowsiness.jpg")
    st.header("Welcome to Drowsiness Detection System")
    st.write("Our Drowsiness Detection System is an AI-powered application designed to monitor eye closure levels in real-time using a webcam. It detects signs of drowsiness by analyzing facial landmarks and calculating the Eye Aspect Ratio (EAR).")

    st.subheader("How It Works?")
    st.write("✔ The system detects the face and eyes using a deep learning-based model.")
    st.write("✔ It calculates the percentage of eye closure based on eye landmarks.")
    st.write("✔ If the eye closure percentage exceeds 50%, a drowsiness alert appears on the screen.")

    st.subheader("Key Features")
    st.write("✔ Real-time Monitoring 🖥️ – Detects drowsiness instantly.")
    st.write("✔ Visual Alert ⚠️ – Displays a red warning message when drowsy.")
    st.write("✔ Non-Intrusive 📷 – Works using a simple webcam.")
    st.write("✔ Customizable Thresholds 🎛️ – Can be fine-tuned for different users.")

    st.write("This system is ideal for drivers, machine operators, students, and professionals who need to stay alert while working!")

    st.write("Would you like to add an alarm sound feature for better alerts? 🔊🚀")


elif (choice=="Video"):
    file=st.file_uploader("Upload Video", type=["mp4", "avi", "mov"])
    window=st.empty()

    if file is not None:
        tfile=tempfile.NamedTemporaryFile()
        tfile.write(file.read())
```

```
        vid=cv2.VideoCapture(tfile.name)
        detector = dlib.get_frontal_face_detector()
        predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")

        def calculate_ear(eye):
            A = dist.euclidean(eye[1], eye[5])  # Vertical distance
            B = dist.euclidean(eye[2], eye[4])  # Vertical distance
            C = dist.euclidean(eye[0], eye[3])  # Horizontal distance
            ear = (A + B) / (2.0 * C)
            return ear

        while vid.isOpened():
            flag, frame = vid.read()
            if not flag:
                break

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

            for face in faces:
                landmarks = predictor(gray, face)

                left_eye = np.array([[landmarks.part(i).x, landmarks.part(i).y] for i in range(36, 42)])
                right_eye = np.array([[landmarks.part(i).x, landmarks.part(i).y] for i in range(42, 48)])
                left_ear = calculate_ear(left_eye)
                right_ear = calculate_ear(right_eye)
                avg_ear = (left_ear + right_ear) / 2.0

                eye_closure_percentage = (1 - avg_ear / 0.3) * 100

                cv2.polylines(frame, [left_eye], True, (0, 255, 0), 1)
                cv2.polylines(frame, [right_eye], True, (0, 255, 0), 1)

                cv2.putText(frame, f"{eye_closure_percentage:.1f}%",
                        (left_eye[0][0], left_eye[0][1] - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255), 2)

                if eye_closure_percentage > 40:
                    cv2.putText(frame, "DROWSINESS ALERT!",
                            (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0, 0, 255), 3)

            cv2.imshow("Drowsiness Detection", frame)

            if cv2.waitKey(5) & 0xFF == ord('x'):
                break

        vid.release()
        cv2.destroyAllWindows()


    elif (choice=="Webcam"):
        detector = dlib.get_frontal_face_detector()
        predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
```

```
    def calculate_ear(eye):
        A = dist.euclidean(eye[1], eye[5])  # Vertical distance
        B = dist.euclidean(eye[2], eye[4])  # Vertical distance
        C = dist.euclidean(eye[0], eye[3])  # Horizontal distance
        ear = (A + B) / (2.0 * C)
        return ear

    vid = cv2.VideoCapture(0)

    while vid.isOpened():
        flag, frame = vid.read()
        if not flag:
            break

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

        for face in faces:
            landmarks = predictor(gray, face)
            left_eye = np.array([[landmarks.part(i).x, landmarks.part(i).y] for i in range(36, 42)])
            right_eye = np.array([[landmarks.part(i).x, landmarks.part(i).y] for i in range(42, 48)])
            left_ear = calculate_ear(left_eye)
            right_ear = calculate_ear(right_eye)
            avg_ear = (left_ear + right_ear) / 2.0

            eye_closure_percentage = (1 - avg_ear / 0.3) * 100

            cv2.polylines(frame, [left_eye], True, (0, 255, 0), 1)
            cv2.polylines(frame, [right_eye], True, (0, 255, 0), 1)

            cv2.putText(frame, f"{eye_closure_percentage:.1f}%",
                        (left_eye[0][0], left_eye[0][1] - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255), 2)

            if eye_closure_percentage > 40:
                cv2.putText(frame, "DROWSINESS ALERT!",
                            (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0, 0, 255), 3)

        cv2.imshow("Drowsiness Detection", frame)

        if cv2.waitKey(5) & 0xFF == ord('x'):
            break

    vid.release()
    cv2.destroyAllWindows()

elif (choice=="IP Camera"):
    k=st.text_input("Enter Camera URL")
    window=st.empty()
    if k:
        print("Enter Camera URL:",k)
        vid=cv2.VideoCapture(str(k)+"/video")

        detector = dlib.get_frontal_face_detector()
```

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

def calculate_ear(eye):
    A = dist.euclidean(eye[1], eye[5])  # Vertical distance
    B = dist.euclidean(eye[2], eye[4])  # Vertical distance
    C = dist.euclidean(eye[0], eye[3])  # Horizontal distance
    ear = (A + B) / (2.0 * C)
    return ear

while vid.isOpened():
    flag, frame = vid.read()
    if not flag:
        break

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

    for face in faces:
        landmarks = predictor(gray, face)
        left_eye = np.array([[landmarks.part(i).x, landmarks.part(i).y] for i in range(36, 42)])
        right_eye = np.array([[landmarks.part(i).x, landmarks.part(i).y] for i in range(42, 48)])
        left_ear = calculate_ear(left_eye)
        right_ear = calculate_ear(right_eye)
        avg_ear = (left_ear + right_ear) / 2.0

        eye_closure_percentage = (1 - avg_ear / 0.3) * 100

        cv2.polylines(frame, [left_eye], True, (0, 255, 0), 1)
        cv2.polylines(frame, [right_eye], True, (0, 255, 0), 1)

        cv2.putText(frame, f"{eye_closure_percentage:.1f}%",
                    (left_eye[0][0], left_eye[0][1] - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255), 2)

        if eye_closure_percentage > 40:
            cv2.putText(frame, "DROWSINESS ALERT!",
                        (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0, 0, 255), 3)

    cv2.imshow("Drowsiness Detection", frame)

    if cv2.waitKey(200) & 0xFF == ord('x'):
        break

vid.release()
cv2.destroyAllWindows()
```