The Code and the Main Working of the Project

The code and algorithms were executed on command prompt only and the whole code was created on the app VS-Code. I have used the libraries like dlib and face\_utils to make an algorithm and make the project work in real time. We also use the web cam of our device in order to get the real time effect of the project. The code captures video frames from the webcam, detects faces within the frames, and analyzes eye blink patterns to determine the user's drowsiness state. Let's go through the code and its key components:

Starting with the necessary libraries required for the code execution:

Libraries Imported:

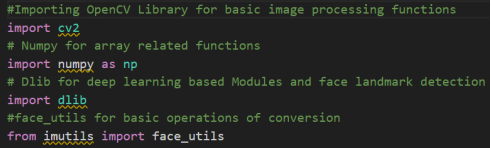
cv2: OpenCV library for image processing and video capture.

numpy: Used for numerical operations (e.g., computing distances).

dlib: Library for face detection and facial landmark detection.

face\_utils: Part of the imutils library, used for converting

Dlib's facial landmarks to NumPy arrays.



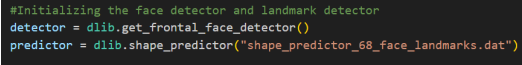
Initializing Webcam Capture:

The cv2.VideoCapture(0) line initializes the camera to capture video from the default webcam



Face and Landmark Detectors:

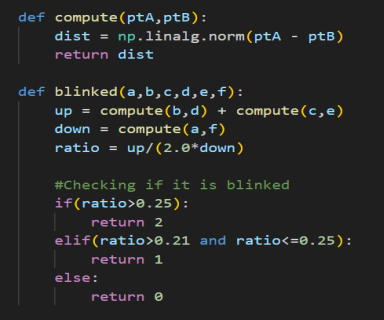
detector = dlib.get\_frontal\_face\_detector(): Initializes the face detector from Dlib. predictor = dlib.shape\_predictor("shape\_predictor\_68\_face\_landmarks.dat"): Initializes the shape predictor from Dlib using a pre-trained model file (shape\_predictor\_68\_face\_landmarks.dat) to detect 68 facial landmarks.



Blink Detection Functions:

compute(ptA, ptB): A helper function that computes the Euclidean distance between two points.

blinked(a, b, c, d, e, f): A function that calculates an eye aspect ratio (EAR) to determine if a blink has occurred.



Drowsiness Detection System Main Loop:

The script enters an infinite loop to continuously read frames from the webcam. Each frame is converted to grayscale using cv2.cvtColor and then passed to the face detector.

For each detected face, a rectangle is drawn around it using cv2.rectangle. The facial landmarks are detected using the shape predictor and converted to a NumPy array for easier manipulation.

Eye blink detection is performed using the landmarks around the eyes and the blinked function.

Depending on the results of the blink detection (whether the user is sleeping, drowsy, or active), the script tracks how long each state persists.

If the user is sleeping or drowsy for a certain amount of time, an alert (status) is displayed on the frame.

Facial landmarks are visualized on the frame as small circles using cv2.circle. The frame with all annotations (status text, face rectangles, and landmarks) is displayed using cv2.imshow



A computer screen shot of a program

Description automatically generated

Final Result : Here we are going to execute the whole code by going to the specific directory where the project is saved and just type “python drowsiness detection.py”. And hence the project will get executed and we will be able to see the results, The result will show three situations where the person is active, drowsy or sleeping. Here we can see the whole project working in full fledged mode and the project actually working.



A person smiling for the camera

Description automatically generated

A person with his eyes closed

Description automatically generated