**Importing Libraries**

import cv2

import numpy as np

import dlib

from imutils import face\_utils

import imutils

import playsound

import threading

import time

**Video Capture Initialization**

def ply\_snd():

playsound.playsound("mixkit-classic-alarm-995.wav")

time.sleep(30)

**Face and Landmark Detector Initialization**

detector = dlib.get\_frontal\_face\_detector()

predictor = dlib.shape\_predictor("C:\\Users\\rakul\\Downloads\\archive\\shape\_predictor\_68\_face\_landmarks.dat")

**Helper Functions**

def compute(ptA, ptB):

dist = np.linalg.norm(ptA - ptB)

return dist

def blink(a, th):

up = compute(a[1], a[5]) + compute(a[2], a[4])

down = compute(a[0], a[3])

ratio = up / (2.0 \* down)

if ratio > abs(th \* 0.75):

return 2 # Open eyes

elif abs(th \* 0.5) < ratio <= abs(th \* 0.75):

return 1 # Half-closed eyes

else:

return 0 # Closed eyes

**Calibration Loop**

i = 0

d = 0

for i in range(0, 10):

ret, frame = cap.read()

...

d += (l + r) / 2.0

d = d / 10

**Main Loop for Drowsiness Detection**

while True:

ret, frame = cap.read()

...

left\_blink = blink(landmarks[36:42], d)

right\_blink = blink(landmarks[42:48], d)

...

if status == "Sleeping!!!" or status == "Drowsy!!":

thread = threading.Thread(target=ply\_snd)

thread.daemon = True

thread.start()

**Drawing and Displaying Results**

cv2.putText(frame, status, (100, 100), cv2.FONT\_HERSHEY\_SIMPLEX, 1.2, color, 3)

for n in range(0, 68):

(x, y) = landmarks[n]

cv2.circle(frame, (x, y), 1, (255, 255, 255), -1)

**End Condition**

if cv2.waitKey(1) == ord('q'):

break