Data.py > ...

```python
import tkinter as tk
from scipy.spatial import distance
from imutils import face_utils
from pygame import mixer
import imutils
import dlib
import cv2
from PIL import Image, ImageTk
from twilio.rest import Client

# Initialize pygame mixer to play alert sound
mixer.init()
mixer.music.load('music.wav')  # Load the alert sound file

# Twilio configuration for sending SMS alerts
TWILIO_ACCOUNT_SID = 'AC7be913fbae807e1b7b029c4b770332fc'
TWILIO_AUTH_TOKEN = '4cc838b0d80cab65f72a50ebf0ce1a0f'
TWILIO_PHONE_NUMBER = '+14847121221'
TO_PHONE_NUMBER = '+916378966072'
client = Client(TWILIO_ACCOUNT_SID, TWILIO_AUTH_TOKEN)

# Function to calculate Eye Aspect Ratio (EAR), a metric for eye openness
def eye_aspect_ratio(eye):
    # Vertical distances
    A = distance.euclidean(eye[1], eye[5])
    B = distance.euclidean(eye[2], eye[4])
    # Horizontal distance
    C = distance.euclidean(eye[0], eye[3])
    # Compute EAR
    ear = (A + B) / (2.0 * C)
    return ear

# Function to send SMS alert and update GUI
def send_sms():
    message = client.messages.create(
        body="Alert: Drowsiness Detected!",
        from_=TWILIO_PHONE_NUMBER,
        to=TO_PHONE_NUMBER
    )
```

Data.py > ...

```
34  def send_sms():
39      )
40      print(f"SMS sent: {message.sid}")
41      sms_status_label.config(text="Alert SMS sent!", fg="blue")  # Update SMS status label
42
43  # Drowsiness detection parameters
44  thresh = 0.25  # EAR threshold for drowsiness detection
45  frame_check = 20  # Consecutive frames required to trigger drowsiness alert
46  flag = 0  # Frame counter for continuous drowsiness detection
47
48  # Initialize face detector and facial landmarks predictor using Dlib
49  detect = dlib.get_frontal_face_detector()
50  predict = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
51
52  # Eye landmark indices in the 68-point facial landmarks model
53  (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS["left_eye"]
54  (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS["right_eye"]
55
56  # Set up webcam feed
57  cap = cv2.VideoCapture(0)
58
59  # Tkinter GUI setup
60  root = tk.Tk()
61  root.title("Drowsiness Detection System")
62  root.geometry("800x600")
63  root.config(bg="#282828")
64
65  # Video feed display in GUI
66  video_frame = tk.Label(root, bg="#282828")
67  video_frame.pack(pady=20)
68
69  # Status label for drowsiness monitoring
70  status_label = tk.Label(root, text="Status: Monitoring...", font=("Helvetica", 16, "bold"), fg="green", bg="#282828")
71  status_label.pack(pady=10)
72
73  # SMS alert status label
74  sms_status_label = tk.Label(root, text="", font=("Helvetica", 14), fg="blue", bg="#282828")
75  sms_status_label.pack(pady=10)
76
```

```python
sms_status_label.pack(pady=10)

# Drowsiness detection function to update video feed and check for alert conditions
def update_frame():
    global flag

    # Capture video frame
    frame = cap.read()
    frame = imutils.resize(frame, width=640)
    frame = cv2.flip(frame, 1)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = detect(gray, 0)

    for face in faces:
        # Detect facial landmarks
        shape = predict(gray, face)
        shape = face_utils.shape_to_np(shape)

        # Calculate EAR for both eyes
        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]
        leftEAR = eye_aspect_ratio(leftEye)
        rightEAR = eye_aspect_ratio(rightEye)
        ear = (leftEAR + rightEAR) / 2.0  # Average EAR

        # Check if EAR falls below threshold to detect drowsiness
        if ear < thresh:
            flag += 1
            # Trigger alert if drowsiness detected for sustained period
            if flag >= frame_check:
                status_label.config(text="ALERT! Drowsiness Detected!", fg="red")
                sms_status_label.config(text="Alert SMS sent!", fg="blue")
                if not mixer.music.get_busy():  # Play alert sound if not already playing
                    mixer.music.play()
                send_sms()  # Send SMS alert
        else:
            flag = 0  # Reset drowsiness counter
            status_label.config(text="Status: Monitoring...", fg="green")
            sms_status_label.config(text="")  # Clear SMS status after alert condition resolves
```

Data.py > ...

```python
def update_frame():
            sms_status_label.config(text="")  # Clear SMS status after alert condition resolves

        # Draw contours around eyes for visual feedback
        leftEyeHull = cv2.convexHull(leftEye)
        rightEyeHull = cv2.convexHull(rightEye)
        cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
        cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)

    # Convert frame for display in Tkinter
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    img = Image.fromarray(frame_rgb)
    img_tk = ImageTk.PhotoImage(image=img)

    # Update GUI video frame
    video_frame.img_tk = img_tk
    video_frame.config(image=img_tk)
    video_frame.after(10, update_frame)  # Schedule next frame update

# Function to stop detection and release resources
def stop_detection():
    cap.release()
    root.quit()

# Start video feed and drowsiness monitoring
update_frame()

# Stop button to end monitoring
stop_button = tk.Button(root, text="Stop Detection", font=("Helvetica", 14), fg="white", bg="#e74c3c", command=stop_detection)
stop_button.pack(pady=20)

# Run Tkinter main loop
root.mainloop()
cap.release()
cv2.destroyAllWindows()
```