

Data.py > ...

```
1  import tkinter as tk
2  from scipy.spatial import distance
3  from imutils import face_utils
4  from pygame import mixer
5  import imutils
6  import dlib
7  import cv2
8  from PIL import Image, ImageTk
9  from twilio.rest import Client
10
11 # Initialize pygame mixer to play alert sound
12 mixer.init()
13 mixer.music.load('music.wav') # Load the alert sound file
14
15 # Twilio configuration for sending SMS alerts
16 TWILIO_ACCOUNT_SID = 'AC7be913fbae807e1b7b029c4b770332fc'
17 TWILIO_AUTH_TOKEN = '4cc838b0d80cab65f72a50ebf0ce1a0f'
18 TWILIO_PHONE_NUMBER = '+14847121221'
19 TO_PHONE_NUMBER = '+916378966072'
20 client = Client(TWILIO_ACCOUNT_SID, TWILIO_AUTH_TOKEN)
21
22 # Function to calculate Eye Aspect Ratio (EAR), a metric for eye openness
23 def eye_aspect_ratio(eye):
24     # Vertical distances
25     A = distance.euclidean(eye[1], eye[5])
26     B = distance.euclidean(eye[2], eye[4])
27     # Horizontal distance
28     C = distance.euclidean(eye[0], eye[3])
29     # Compute EAR
30     ear = (A + B) / (2.0 * C)
31     return ear
32
33 # Function to send SMS alert and update GUI
34 def send_sms():
35     message = client.messages.create(
36         body="Alert: Drowsiness Detected!",
37         from_=TWILIO_PHONE_NUMBER,
38         to=TO_PHONE_NUMBER
39     )
40     # Update GUI with message ID
41     message_id_label.config(text=f"Message ID: {message.sid}")
```

```
1  # Import necessary libraries
2  import cv2
3  import dlib
4  import imutils
5  import numpy as np
6  import sys
7  import time
8  import tkinter as tk
9  from PIL import Image, ImageTk
10 from twilio.rest import Client
11
12 # Initialize Twilio client
13 TWILIO_ACCOUNT_SID = 'AC7be913fbae807e1b7b029c4b770332fc'
14 TWILIO_AUTH_TOKEN = '4cc838b0d80cab65f72a50ebf0ce1a0f'
15 TWILIO_PHONE_NUMBER = '+14847121221'
16 TO_PHONE_NUMBER = '+916378966072'
17 client = Client(TWILIO_ACCOUNT_SID, TWILIO_AUTH_TOKEN)
18
19 # Initialize OpenCV and Dlib
20 detector = dlib.get_frontal_face_detector()
21 predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
22
23 # Initialize Tkinter GUI
24 root = tk.Tk()
25 root.title("Drowsiness Detection")
26 root.geometry("600x400")
27
28 # Create labels for video path, message ID, and EAR
29 video_path_label = tk.Label(root, text="Video Path: ")
30 message_id_label = tk.Label(root, text="Message ID: ")
31 ear_label = tk.Label(root, text="EAR: ")
32
33 # Create a video file path
34 video_path = "video.mp4"
35
36 # Create a video object
37 video = cv2.VideoCapture(video_path)
38
39 # Create a window for the video
40 window = tk.Toplevel(root)
41 window.title("Video Feed")
42 window.geometry("600x400")
43
44 # Create a canvas for the video
45 canvas = tk.Canvas(window, width=600, height=400)
46 canvas.pack()
47
48 # Create a video frame
49 frame = cv2.cvtColor(video.read(), cv2.COLOR_BGR2RGB)
50
51 # Create a video image
52 image = ImageTk.PhotoImage(Image.fromarray(frame))
53
54 # Create a video label
55 video_label = tk.Label(window, image=image)
56 video_label.pack()
57
58 # Create a video object
59 video = cv2.VideoCapture(video_path)
60
61 # Create a video object
62 video = cv2.VideoCapture(video_path)
63
64 # Create a video object
65 video = cv2.VideoCapture(video_path)
66
67 # Create a video object
68 video = cv2.VideoCapture(video_path)
69
70 # Create a video object
71 video = cv2.VideoCapture(video_path)
72
73 # Create a video object
74 video = cv2.VideoCapture(video_path)
75
76 # Create a video object
77 video = cv2.VideoCapture(video_path)
78
79 # Create a video object
80 video = cv2.VideoCapture(video_path)
81
82 # Create a video object
83 video = cv2.VideoCapture(video_path)
84
85 # Create a video object
86 video = cv2.VideoCapture(video_path)
87
88 # Create a video object
89 video = cv2.VideoCapture(video_path)
90
91 # Create a video object
92 video = cv2.VideoCapture(video_path)
93
94 # Create a video object
95 video = cv2.VideoCapture(video_path)
96
97 # Create a video object
98 video = cv2.VideoCapture(video_path)
99
100 # Create a video object
101 video = cv2.VideoCapture(video_path)
```

```
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
```



```
75 sms_status_label.pack(pady=10)
76
77 # Drowsiness detection function to update video feed and check for alert conditions
78 def update_frame():
79     global flag
80
81     # Capture video frame
82     frame = cap.read()
83     frame = imutils.resize(frame, width=640)
84     frame = cv2.flip(frame, 1)
85     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
86     faces = detect(gray, 0)
87
88     for face in faces:
89         # Detect facial landmarks
90         shape = predict(gray, face)
91         shape = face_utils.shape_to_np(shape)
92
93         # Calculate EAR for both eyes
94         leftEye = shape[lStart:lEnd]
95         rightEye = shape[rStart:rEnd]
96         leftEAR = eye_aspect_ratio(leftEye)
97         rightEAR = eye_aspect_ratio(rightEye)
98         ear = (leftEAR + rightEAR) / 2.0 # Average EAR
99
100     # Check if EAR falls below threshold to detect drowsiness
101     if ear < thresh:
102         flag += 1
103         # Trigger alert if drowsiness detected for sustained period
104         if flag >= frame_check:
105             status_label.config(text="ALERT! Drowsiness Detected!", fg="red")
106             sms_status_label.config(text="Alert SMS sent!", fg="blue")
107             if not mixer.music.get_busy(): # Play alert sound if not already playing
108                 mixer.music.play()
109             send_sms() # Send SMS alert
110     else:
111         flag = 0 # Reset drowsiness counter
112         status_label.config(text="Status: Monitoring...", fg="green")
113         sms_status_label.config(text="") # Clear SMS status after alert condition resolves
```



```
78 def update_frame():
112     status_label.config(text=status_monitoring, fg="green",
113     sms_status_label.config(text="") # Clear SMS status after alert condition resolves
114
115     # Draw contours around eyes for visual feedback
116     leftEyeHull = cv2.convexHull(leftEye)
117     rightEyeHull = cv2.convexHull(rightEye)
118     cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
119     cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
120
121     # Convert frame for display in Tkinter
122     frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
123     img = Image.fromarray(frame_rgb)
124     img_tk = ImageTk.PhotoImage(image=img)
125
126     # Update GUI video frame
127     video_frame.img_tk = img_tk
128     video_frame.config(image=img_tk)
129     video_frame.after(10, update_frame) # Schedule next frame update
130
131 # Function to stop detection and release resources
132 def stop_detection():
133     cap.release()
134     root.quit()
135
136 # Start video feed and drowsiness monitoring
137 update_frame()
138
139 # Stop button to end monitoring
140 stop_button = tk.Button(root, text="Stop Detection", font=("Helvetica", 14), fg="white", bg="#e74c3c", command=stop_detection)
141 stop_button.pack(pady=20)
142
143 # Run Tkinter main loop
144 root.mainloop()
145 cap.release()
146 cv2.destroyAllWindows()
```