BACKEND CODE

```
# from flask import Flask, render_template, Response
# import cv2
# import mediapipe as mp
# import numpy as np
# from scipy.spatial import distance
# from pygame import mixer

# app = Flask(_name_)

# # Initialize pygame mixer
# mixer.init()
# mixer.music.load("music.wav")

# # Function to calculate Eye Aspect Ratio (EAR)
# def eye_aspect_ratio(eye):
#     A = distance.euclidean(eye[5], eye[5])
#     B = distance.euclidean(eye[2], eye[4])
#     C = distance.euclidean(eye[0], eye[3])
#     ear = (A + B) / (2.0 * C)
#     return ear
```

```
# # Threshold and frame check parameters
# thresh = 0.25
# frame_check = 30

# # Initialize Mediapipe Face Mesh
# mp_face_mesh = mp.solutions.face_mesh
# face_mesh =
mp_face_mesh.FaceMesh(max_num_faces=1,
min_detection_confidence=0.5,
min_tracking_confidence=0.5)

# def generate_frames():
#     cap = cv2.VideoCapture(0)
#     flag = 0

#     while True:
#         ret, frame = cap.read()
#         if not ret:
#             break

#         rgb_frame = cv2.cvtColor(frame,
cv2.COLOR_BGR2RGB)
#         results = face_mesh.process(rgb_frame)
```

```
#       if results.multi_face_landmarks:
#           for face_landmarks in results.multi_face_landmarks:
#               landmarks = face_landmarks.landmark

#               # Get coordinates of left and right eyes
#               left_eye = [landmarks[i] for i in [33, 160, 158, 133, 153, 144]]
#               right_eye = [landmarks[i] for i in [362, 385, 387, 263, 373, 380]]

#               # Convert to numpy arrays
#               left_eye = np.array([(p.x * frame.shape[1], p.y * frame.shape[0]) for p in left_eye], dtype=np.int32)
#               right_eye = np.array([(p.x * frame.shape[1], p.y * frame.shape[0]) for p in right_eye], dtype=np.int32)

#               # Calculate EAR for both eyes
#               leftEAR = eye_aspect_ratio(left_eye)
#               rightEAR = eye_aspect_ratio(right_eye)
#               ear = (leftEAR + rightEAR) / 2.0
```

```
#             # Draw contours around the eyes
#             cv2.drawContours(frame, [left_eye], -1, (0, 255,
0), 1)
#             cv2.drawContours(frame, [right_eye], -1, (0, 255,
0), 1)


#             # Check if EAR is below the threshold
#             if ear < thresh:
#                 flag += 1
#                 if flag >= frame_check:
#                     cv2.putText(frame, "ALERT", (10, 30),
#                                 cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,
0, 255), 2)
#                     cv2.putText(frame, "ALERT", (10, 325),
#                                 cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,
0, 255), 2)
#                     mixer.music.play()
#             else:
#                 flag = 0


#         # Encode the frame as JPEG
#         _, buffer = cv2.imencode('.jpg', frame)
#         frame = buffer.tobytes()
```

```
#         # Yield the frame
#         yield (b'--frame\r\n'
#                b'Content-Type: image/jpeg\r\n\r\n' + frame +
b'\r\n')

#     cap.release()

# @app.route('/')
# def index():
#     return render_template('index.html')

# @app.route('/video_feed')
# def video_feed():
#     return Response(generate_frames(),
mimetype='multipart/x-mixed-replace; boundary=frame')

# if _name_ == '_main_':
#     app.run(debug=True)



from flask import Flask, render_template, Response
```

```
import cv2
import mediapipe as mp
print("MediaPipe Version:", mp._version_)
import numpy as np
from scipy.spatial import distance
from pygame import mixer

app = Flask(_name_)

# Initialize pygame mixer
mixer.init()
mixer.music.load("music.wav")

# Function to calculate Eye Aspect Ratio (EAR)
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])
    # EAR formula
    ear = (A + B) / (2.0 * C)
```

```
    return ear

# EAR threshold for detecting a blink
thresh = 0.25
eye_closed = False  # To track if the eyes are currently closed

# Initialize Mediapipe Face Mesh
mp_face_mesh = mp.solutions.face_mesh
face_mesh = mp_face_mesh.FaceMesh(
    max_num_faces=1,
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5
)

def generate_frames():
    global eye_closed
    cap = cv2.VideoCapture(0)

    while True:
        ret, frame = cap.read()
        if not ret:
```

```
            break

        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        results = face_mesh.process(rgb_frame)

        if results.multi_face_landmarks:
            for face_landmarks in results.multi_face_landmarks:
                landmarks = face_landmarks.landmark

                # Get coordinates of left and right eyes
                left_eye = [landmarks[i] for i in [33, 160, 158, 133,
153, 144]]
                right_eye = [landmarks[i] for i in [362, 385, 387,
263, 373, 380]]

                # Convert to numpy arrays
                left_eye = np.array([(int(p.x * frame.shape[1]),
int(p.y * frame.shape[0])) for p in left_eye])
                right_eye = np.array([(int(p.x * frame.shape[1]),
int(p.y * frame.shape[0])) for p in right_eye])

                # Calculate EAR for both eyes
                leftEAR = eye_aspect_ratio(left_eye)
```

```
            rightEAR = eye_aspect_ratio(right_eye)
            ear = (leftEAR + rightEAR) / 2.0

            # Draw contours around the eyes
            cv2.drawContours(frame, [left_eye], -1, (0, 255, 0),
1)
            cv2.drawContours(frame, [right_eye], -1, (0, 255,
0), 1)

            # Check if EAR is below the threshold (eyes closed)
            if ear < thresh:
               if not eye_closed:
                  eye_closed = True
                  # Play sound continuously
                  mixer.music.play(-1)  # -1 means the sound will
loop indefinitely
                  cv2.putText(frame, "Eyes Closed!", (10, 30),
                              cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0,
255), 2)
            else:
               if eye_closed:
                  eye_closed = False
                  # Stop the sound
```

```
                mixer.music.stop()
                cv2.putText(frame, "Eyes Open", (10, 30),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,
255, 0), 2)
            else:
                cv2.putText(frame, "Eyes Open", (10, 30),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,
255, 0), 2)
        else:
            # If no face is detected, ensure sound is stopped
            if eye_closed:
                eye_closed = False
                mixer.music.stop()
            cv2.putText(frame, "No Face Detected", (10, 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255,
255), 2)

        # Encode the frame as JPEG
        _, buffer = cv2.imencode('.jpg', frame)…
```

FRONTEND CODE

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Drowsiness Detection</title>
   <style>
      /* Styling for the body of the page, including background and text alignment */
      body {
         font-family: 'Arial', sans-serif;
         text-align: center;
         background: linear-gradient(135deg, #007bff, #0056b3);
         color: #fff;
         padding: 20px;
         margin: 0;
         min-height: 100vh;
         display: flex;
         flex-direction: column;
         justify-content: center;
```

```
    align-items: center;
}

/* Styling for the main title */
h1 {
    margin-bottom: 20px;
    font-size: 2.5rem;
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3);
}

/* Styling for buttons, including size, color, and hover effects */
button {
    padding: 15px 30px;
    font-size: 1.2rem;
    cursor: pointer;
    border: none;
    border-radius: 25px;
    margin: 10px;
    transition: all 0.3s ease-in-out;
}
```

```
    /* Button color for the start button */
    button#start-btn {
        background-color: #28a745;
        color: white;
    }

    /* Button color for the stop button */
    button#stop-btn {
        background-color: #dc3545;
        color: white;
        display: none; /* Initially hidden */
    }

    /* Hover effect for buttons */
    button:hover {
        transform: scale(1.1);
        box-shadow: 0px 8px 15px rgba(0, 0, 0, 0.2);
    }

    /* Styling for the video container, including size and appearance */
    #video-container {
```

```
            margin-top: 20px;
            width: 90%;
            max-width: 600px;
            display: none; /* Initially hidden */
            background-color: rgba(255, 255, 255, 0.1);
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.3);
        }

        /* Styling for the image (video feed) inside the container */
        img {
            width: 100%;
            height: auto;
            border-radius: 10px;
            border: 2px solid white;
        }
    </style>
</head>
<body>
    <!-- Main header of the page -->
```

```
    <h1>Drowsiness Detection</h1>
    <!-- Start camera button with an onclick event to trigger the camera start -->
    <button id="start-btn" onclick="startCamera()">Start Camera</button>

    <!-- Stop camera button, initially hidden, with an onclick event to stop the camera -->
    <button id="stop-btn" onclick="stopCamera()">Stop Camera</button>

    <!-- Video container that will display the camera feed -->
    <div id="video-container">
        <h2>Camera Feed</h2>
        <!-- Image element to display the video feed -->
        <img src="/video_feed" alt="Camera Feed" onerror="handleError()">
    </div>

    <script>
        /* Function to start the camera feed and show the video container */
        function startCamera() {
```

```
        document.getElementById('video-
container').style.display = 'block';
        document.getElementById('start-btn').style.display =
'none';
        document.getElementById('stop-btn').style.display =
'inline-block';
    }

    /* Function to stop the camera feed and hide the video
container */
    function stopCamera() {
        document.getElementById('video-
container').style.display = 'none';
        document.getElementById('stop-btn').style.display =
'none';
        document.getElementById('start-btn').style.display =
'inline-block';
    }

    /* Error handling function for the camera feed */
    function handleError() {
        alert("Unable to load camera feed. Please check your
camera or server settings.");
    }
```

```
    </script>
</body>
</html>
```