VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“Jnana Sangama”, Belagavi-590 014

A Mini - Project Report

On

“DROWSINESS DETECTION”

Submitted in partial fulfillment of the requirements for the **MINI PROJECT (BCD586)** course of the 5th semester

Bachelor of Engineering

In

Computer Science & Engineering (DATA SCIENCE)

Submitted by

Ms. Amrutha B V	**Ms. Ananya B K**
(4AI22CD003)	(4AI22CD004)
Ms. Bhumika K	**Ms. Hitha B R**
(4AI22CD009)	(4AI22CD026)

Under the guidance of

Mrs. PALLAVI C S B.E., M.Tech.

Assistant Professor

Department of CS&E (DATA SCIENCE)

Adichunchanagiri Institute of Technology

CHIKKAMAGALURU - 577102

2024-25

ADHICHUNCHANAGIRI INSTITUTE OF TECHNOLOGY

Jyothinagar, Chikkamagaluru-577102

DEPARTMENT OF CS&E (DATA SCIENCE)

CERTIFICATE

This is to certify that the Mini project work entitled **"DROWSINESS DETECTION"** is a bonafied work carried out by **Ms. Amrutha B V (4AI22CD003), Ms. Ananya B K (4AI22CD004), Ms. Bhumika K (4AI22CD009), Ms. Hitha B R (4AI22CD026)** in partial fulfillment for the **Mini Project (BCD586)** course of 5th semester Bachelor of Engineering in **Computer Science and Engineering (Data Science)** of the Visvesvaraya Technological University, Belagavi during the academic year **2024-2025**. It is certified that all corrections and suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The Mini project report has been approved as it satisfies the academic requirements in respect of Project Work prescribed for the said Degree.

Signature of the Guide	**Signature of Coordinator**
Mrs.Pallavi C S B.E., M.Tech	**Mrs. Shilpa K V.** B.E., M.Tech
Assistant Professor	**Assistant Professor**

Signature of the HOD
Dr. Adarsh M J B.E., M.Tech., Ph.D
Associate Professor and Head

ABSTRACT

Drowsiness detection in online classes is an emerging field aimed at addressing the challenge of reduced student engagement and productivity in remote learning environments. With the shift to virtual education, prolonged screen time and lack of direct supervision often lead to fatigue and inattentiveness among students. This study focuses on implementing a real-time drowsiness detection system that monitors facial features and behavior patterns to identify signs of tiredness, such as closed eyes.

The proposed system employs advanced computer vision techniques and machine learning algorithms to analyze video feed from students' webcams. Features like eye aspect ratio (EAR) and facial landmarks are extracted to detect micro-expressions indicative of drowsiness. When signs of drowsiness are detected, the system alerts the student to restore attentiveness, ensuring a more interactive and focused learning experience.

This research highlights the potential of integrating artificial intelligence into virtual education to enhance student performance. By promoting sustained attention and minimizing learning disruptions, the system addresses key challenges in online learning. The approach also ensures privacy, as the analysis occurs locally without storing personal data.

ACKNOWLEDGEMENTS

We express our humble pranamas to his holiness **Divine Soul Parama Poojya Jagadguru Padmabushana Sri Sri Sri Dr.Balagangadharanatha Maha Swamiji** and **Parama Poojya Jagadguru Sri Sri Sri Dr. Nirmalanandanatha Maha Swamiji Pontiff, Sri Adichunchanagiri Maha Samsthana Matt and Sri Sri Gunanatha Swamiji,** Chikkamagaluru branch, Sringeri who have showered their blessings on us.

The completion of any project involves the efforts of many people. We have been lucky enough to have received a lot of help and support from all quarters during the making of this project, so with gratitude, we take this opportunity to acknowledge all those whose guidance and encouragement helped us emerge successful.

We express our gratitude to **Dr. C K Subbaraya**, Director, Adichunchanagiri Institute of Technology.

We express our sincere thanks to our beloved principal**, Dr. C T Jayadeva** for having supported us in our academic endeavors.

We are thankful to the resourceful guidance, timely assistance and graceful gesture of our guide **Mrs. Pallavi C S,** Asst. Professor, Department of CS&E (DATA SCIENCE), who has helped us in every aspect of our project work.

We thank our project coordinator **Mrs. Shilpa K V** Asst. Professor, Department of CS&E (DATA SCIENCE), for her lively correspondence and assistance in carrying on with this project

We are also indebted to **Dr. Adarsh M J,** HOD of CS&E (DATA SCIENCE) Department, for the facilities and support extended towards us.

We would be very pleased to express our heart full thanks to all the teaching and non-teaching staff of CS&E (DATA SCIENCE) Department and our friends who have rendered their help, motivation and support.

Amrutha B V

Ananya B K

Bhumika K

Hitha B R

CONTENTS

List of Tables

List of Snapshots

Chapter 1

Introduction

1.1 Background

- **Context**: With the rise of online education, student's engagement and attention are increasingly difficult to monitor, leading to decreased focus and drowsiness. This poses a challenge to educators trying to maintain effective learning environments remotely.
- **Problem**: Drowsiness during online classes often goes undetected, affecting student performance and retention as instructors cannot physically monitor students' attention or well-being.
- **Opportunity**: Technologies like AI and computer vision offer a chance to develop automated systems for detecting drowsiness, enabling real-time interventions and improving student engagement and learning outcomes.

1.2 Problem Statement

- **Overview of the Problem**: Drowsiness during online classes reduces student engagement and learning outcomes, as it is difficult for instructors to monitor attentiveness remotely. This project aims to create an automated system to detect drowsiness and improve student focus.
- **Specific Issues**:
  - **Inability to Monitor Behavior:** Instructors can't observe students' physical cues like yawning or head drooping.
  - **Subtle Fatigue Symptoms**: Early signs of drowsiness are often too faint to be detected manually.
  - **Increased Distractions:** Home environments and long screen time contribute to higher chances of student fatigue.
  - **Decreased Retention**: Drowsy students struggle to retain key information, affecting academic performance.

1.3 Objective of the System

- The primary objective of the **drowsiness detection** system in online classes is to monitor student's alertness levels and ensure active participation during lessons.
- **Key Goals**:
  - **Monitor Eye Movement**: Track the position and movement of the eyes to determine if they are closing or exhibiting signs of drowsiness.
  - **Measure Eye Closure**: Calculate the eye aspect ratio (EAR) or similar metrics to detect prolonged eye closure, indicating potential drowsiness.
  - **Alert System**: Trigger alerts or notifications when drowsiness is detected to help prevent accidents or ensure timely breaks.
  - **Real-time Processing**: Implement real-time image processing to monitor and analyse the driver's or user's state continuously.

1.4 Significance of the System

- **Improves Student Focus**: Detects drowsiness to ensure students remain attentive during classes.
- **Enhances Learning Outcomes**: Minimizes distractions and ensures better information retention.
- **Real-time Monitoring**: Provides immediate feedback to both students and instructors about attention levels.
- **Supports Personalized Learning**: Adjusts teaching methods based on student engagement and alertness.
- **Increases Course Effectiveness**: Helps maintain a high level of interaction and participation in online classes.

1.5 Scope of the Project

- **In Scope**:
    - **Facial Expression and Eye Movement Tracking:** Detecting drowsiness through students' facial expressions and eye movements during online classes.
    - **Real-Time Notifications**: Sending alerts to both students and instructors when drowsiness is detected.

- **Out of Scope**:
    - **Health Diagnosis**: The system will not diagnose or monitor health conditions beyond drowsiness.
    - **Offline Usage:** The system is not designed for offline or in-person classrooms.
    - **Advanced Sensor Integration:** No use of non-visual sensors like EEG or heart rate monitors for detecting drowsiness.

1.6 Methodology

- **Face and Eye Detection:** The code uses MediaPipe’s Face Mesh to detect facial landmarks and extract the eye coordinates (left and right eyes) from the webcam feed.

- **Eye Aspect Ratio (EAR) Calculation:** It calculates the Eye Aspect Ratio (EAR) to monitor eye closure by measuring distances between specific eye landmarks.

- **Drowsiness Detection:** If the average EAR falls below a defined threshold for a set number of frames, an alert is triggered.

- **Sound Alert:** When drowsiness is detected, a sound (from music.wav) plays as an alert to warn the user.

1.7 Target Audience

- **Students:** Helps students stay alert during long online classes by detecting drowsiness and prompting them to refocus.
- **Teachers/Instructors:** Enables teachers to monitor student engagement and potentially reduce distractions during virtual lessons.
- **Educational Institutions**: Provides schools and universities with a tool to improve online learning experiences and maintain student attentiveness.

1.8 Overview of the Report

- This report is structured into several chapters that detail the development and design of the **Drowsiness Detection**. The following chapters include:
  - **Chapter 2: System Design** – Describes the architecture and design of the system.
  - **Chapter 3: Implementation** – Discusses the system's development and the technologies used.
  - **Chapter 4: Testing and Validation** – Details the testing process and results.
  - **Chapter 5: Results and Discussions** – Presents and results obtained and discusses the limitations
  - **Chapter 6: Conclusion and Future enhancement -** Summarizes the project and suggests future improvements.

Chapter 2

System Design

2.1 System Architecture

- **Video Capture Module:**
  - Captures live video from the webcam using OpenCV (cv2.VideoCapture).
- **Face Detection & Landmark Identification:**
  - Uses MediaPipe (mp.solutions.face_mesh) to detect face landmarks in the captured frames.
- **Eye Aspect Ratio (EAR) Calculation:**
  - Extracts key eye landmarks (left and right eyes).
  - Calculates the Eye Aspect Ratio (EAR) to determine if the eyes are closed.
- **EAR Thresholding & Alert:**
  - Compares EAR values with a predefined threshold (thresh).
  - If EAR is below the threshold for a set number of frames, it triggers an alert.
- **Alert System:**
  - If an alert is triggered, a warning message ("ALERT!") is displayed on the screen.
  - Plays an audio alert using Pygame's mixer.
- **UI & Visualization:**
  - Draws contours around the eyes for visualization of detected features.
  - Displays the processed video feed in a window (cv2.imshow).

2.2 Module Design

- The system is divided into functional modules, each handling a specific task.

2.2.1 Webcam Capture Module

- Responsibility: Capture and preprocess video frames.
- Function: capture_frame()
- Input: Webcam feed
- Output: Preprocessed frame (RGB).

2.2.2 Landmark Detection Module

- Responsibility: Detect facial and eye landmarks using MediaPipe.
- Function: detect_landmarks()
- Input: Preprocessed frame
- Output: Eye landmarks (left & right).

2.2.3 EAR Calculation Module

- Responsibility: Calculate Eye Aspect Ratio (EAR)
- Function: eye_aspect_ratio()
- Input: Eye landmarks
- Output: EAR value for each eye

2.2.4 Drowsiness Detection Module

- Responsibility: Detect drowsiness based on EAR threshold.
- Function: check_drowsiness()
- Input: EAR, flag, threshold
- Output: Drowsiness status (True/False)

2.2.5 Alert System Module

- Responsibility: Display visual and audio alerts
- Function: show_alert(), play_alert_sound()
- Input: Drowsiness status
- Output: Visual alert, audio alert

2.2.6 Main Loop Module

- Responsibility: Coordinate all modules
- Function: run_system()
- Input: Webcam feed, user input
- Output: Processed frame with alerts

2.3 User Interface (UI) Design

- **Video Feed**: Displays real-time webcam video with eye contours.
- **Eye Contours**: Green contours around detected eyes.
- **Alert Text**: Red "ALERT!" text appears when drowsiness is detected.
- **Sound Alert**: Plays an audio alert when drowsiness is detected.

2.4 Technology Stack

- **Frontend**:
  - **OpenCV:**
    Purpose: Displays real-time video and overlays visual alerts on the screen (e.g., text, contours around eyes).
  - **Pygame (mixer):**
    Purpose: Plays alert sound when drowsiness is detected.
  - **MediaPipe:**
    Purpose: Used for facial landmark detection, particularly for tracking eye landmarks to calculate the Eye Aspect Ratio (EAR).

- **Backend**:
  - **Python:**
    Purpose: The core programming language to implement all system functionalities Node.js or Python with Express or Django for business logic and server-side processing.
  - **OpenCV:**
    Purpose: Captures and processes video frames, handles image manipulations.

- **MediaPipe:**
  Purpose:Provide machine learning-based facial and eye landmark detection.
- **NumPy:**
  Purpose:Handles numerical computations for landmark scaling and EAR calculation.
- **SciPy (distance):**
  Purpose:Calculates the Euclidean distances between landmarks to compute EAR.
- **Pygame:**
  Purpose: Plays audio alerts when drowsiness is detected.

- **Database**:
  No Database Used:
  - The code does not use any database.
  - It operates entirely in real-time by capturing video frames from a webcam.

Chapter 3

Implementation

3.1 Backend Implementation

```
# 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)
        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)
```

Here's a breakdown of the main components:

1. Imports

- Flask, render_template, Response: Used for creating a web application and serving a video feed to a web page.
- cv2 (OpenCV): Handles video capture and image processing.
-mediapipe as mp: Provides the **Face Mesh* solution for detecting facial landmarks.
- numpy: Used for array manipulation when processing eye landmarks.
- scipy.spatial.distance: Used to calculate Euclidean distances, required for the EAR formula.
- pygame.mixer: Manages audio playback to provide an alert sound.

2. Audio Alert Setup

```
python
mixer.init()
mixer.music.load("music.wav")
```

- Initializes *pygame.mixer* to play a sound file (music.wav) as an alert when eyes are detected as closed.

3. Eye Aspect Ratio (EAR) Function

```
python
def eye_aspect_ratio(eye):
    A = distance.euclidean(eye[1], eye[5])  # Vertical distances
    B = distance.euclidean(eye[2], eye[4])
    C = distance.euclidean(eye[0], eye[3])  # Horizontal distance
    ear = (A + B) / (2.0 * C)
    return ear
```

- EAR measures the ratio of vertical and horizontal distances between key eye landmarks.
- A low EAR indicates the eye is closed.

4. Face Mesh Initialization

- Initializes the *Face Mesh* model to detect facial landmarks, focusing on the eyes.
- Limits detection to one face at a time for simplicity.

5. Frame Processing and Eye Detection

The generate_frames function processes video frames in real-time and determines if the eyes are open or closed:

```
python
cap = cv2.VideoCapture(0)
```

- Captures video input from the default webcam.

Eye Coordinates Extraction:

```
python
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]]
```

- Specific face mesh landmarks correspond to eye positions. These are extracted for EAR calculation.

EAR Calculation and Alert:

- If the EAR is below the **threshold (thresh), the eyes are detected as closed, and an alert sound is played continuously.
- If eyes are open, the sound stops.

Drawing Eye Contours and Status:

```
python
cv2.drawContours(frame, [left_eye], -1, (0, 255, 0), 1)
cv2.putText(frame, "Eyes Closed!", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, ...)
```

- Draws green contours around the eyes and overlays status text ("Eyes Closed" or "Eyes Open").

6. Flask Application

Routes:

- @app.route('/'): Serves the main HTML page (index.html), which contains a video feed.
- @app.route('/video_feed'): Streams the processed video frames as a **multipart/x-mixed-replace** response, which is suitable for live video in a browser.

Running the App:

```
python
if __name__ == '__main__':
    app.run(debug=True)
```

- Launches the Flask application in debug mode.

3.2Frontend Implementation

```
<!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);
      }

      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>
    <h1>Drowsiness Detection</h1>
    <button id="start-btn" onclick="startCamera()">Start Camera</button>
    <button id="stop-btn" onclick="stopCamera()">Stop Camera</button>
    <div id="video-container">
        <h2>Camera Feed</h2>
        <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>
```

Explanation:

This HTML document creates a simple webpage for a Drowsiness Detection system with a user interface to toggle a camera feed on and off. Below is a detailed explanation of the components:

HTML Structure

1. DOCTYPE and <html> tags:
   - Specifies the HTML5 document type.
   - The lang="en" attribute defines the language of the content as English.

2. Head Section (<head>):
   - Includes metadata (<meta> tags) like character encoding (UTF-8) and viewport settings for responsive design.
   - The <title> sets the webpage's title as "Drowsiness Detection".
   - A <style> block is included for defining the page's CSS styling.

3. Body Section (<body>):
   - The main content of the webpage includes:
     - A header (<h1>) displaying the title "Drowsiness Detection".
       - "Start Camera": Initially visible and triggers the startCamera function when clicked.
       - "Stop Camera": Initially hidden and triggers the stopCamera function when clicked.
     - A container (<div id="video-container">) for the camera feed, which is initially hidden. Inside this container:
       - A subheading (<h2>) labeled "Camera Feed."
       - An <img> element displaying the video feed from the source /video_feed. If there's an error loading the feed, the handleError function is called.

CSS Styling

The styles are defined in the <style> block within the <head> section:

1. Global Styles:
   - A body style defines the font, background gradient, text alignment, padding, and a flexbox layout for centering content vertically and horizontally.
   - Adds consistent styling across all buttons and a hover effect (scale and shadow).

2. Specific Element Styles:
   - h1: Larger font size, margin, and a subtle text shadow for emphasis.
   - Buttons:
   - #start-btn: Green background (#28a745) for the "Start Camera" button.
   - #stop-btn: Red background (#dc3545) for the "Stop Camera" button, initially hidden using display: none.

- #video-container: A responsive, styled container for the camera feed with rounded corners and a translucent background.

JavaScript Functionality
Defined in the <script> block within the <body> section:

1. startCamera() Function:
   - Displays the #video-container by setting its style.display to 'block'.
   - Hides the "Start Camera" button (#start-btn) and displays the "Stop Camera" button (#stop-btn).

2. stopCamera() Function:
   - Hides the #video-container and "Stop Camera" button.
   - Displays the "Start Camera" button.

3. handleError() Function:
   - Triggered if the image (camera feed) fails to load, e.g., if /video_feed is unreachable.
   - Shows an alert to the user indicating the issue.

Use Case

This webpage could be used for a **Drowsiness Detection System** where the camera captures the feed, and server-side logic analyzes it (e.g., detecting eye closure). The /video_feed URL would typically serve the video stream, provided by a server using a framework like Flask or Django in Python.

Chapter 4

Testing

4.1 Testing Objectives

- **Functionality Validation**: Ensure the EAR calculation detects drowsiness accurately.
- **Threshold Accuracy**: Verify the threshold (thresh=0.25) and frame check (frame_check=30) trigger alerts correctly.
- **Landmark Detection**: Confirm accurate identification and tracking of eye landmarks using MediaPipe.
- **Alert Mechanism:** Test the audio and visual alert system for proper activation when drowsiness is detected.
- **Frame Processing**: Validate real-time video frame resizing, color conversion, and landmark processing.
- **Performance:** Check the program's performance and responsiveness during live video capture.
- **Error Handling**: Ensure the program gracefully handles edge cases like no face detected or interrupted video feed.

4.2 Testing Environment

1.Hardware:

- A computer with a wedcam for real-time video capture.
- Speaker or headphones for audio alerts.

2.Software:

- Python 3.7 or above.
- Required libraries:OpenCV, MediaPipe, NumPy, SciPy, Pygame, imutils.

3. System Specifications:

- Processor: Minimum dual-core CPU for smooth video processing.
- RAM: At least 4 GB for running real-time application.

4. Audio File:

A valid music.wav file placed in the same directory as the code.

6. Camera Position:

Webcam positioned to capture a clear view of the user's face.

7. Python IDE:

Any IDE or terminal that supports Python script execution (e.g.,VS Code).

4.3 Test Cases

Below are sample test cases for various components:

Table 4.1: Test Cases

Test Case ID	Test Scenario	Test Steps	Expected Result	Status
TC-001	Verify EAR calculation	Pass predefined eye landmarks to eye_aspect_ration()	EAR is calculated accurately	Pass
TC-002	Detect face and eyes	Run the code with a clear fontal face in the camera frame.	Eyes are correctly detected,and landmarks are visible	Pass
TC-003	Trigger drowsiness alert	Simulate a closed eye condition on for more than 30 frames	Visual and audio alerts are triggered.	Pass
TC-004	Handle open eyes	Look straight at the camera with open eyes.	No alert triggered;flag counter resets.	Pass
TC-005	Test alert reset mechanism	Blink for less than 30 consecutive frames.	Alert does not trigger;flag counter resets after eyes open.	Pass
TC-006	Handle multiple faces	Have multiple people in the frame.	Only one closest face is tackled and processed.	Pass
TC-007	Handle no face in the frame	Move out of the camera’s view.	No crash,program continues running.	Pass
TC-008	Test video resizing and display	Ensure the frame is resized to 450 px width.	Resized frame is displayed correctly without distortion.	Pass
TC-009	Handle multiple consecutive alerts.	Simulate drowsiness(closed eyes)multiple times with short gaps in between.	Alerts are triggered each time the condition is met without overlap.	Pass

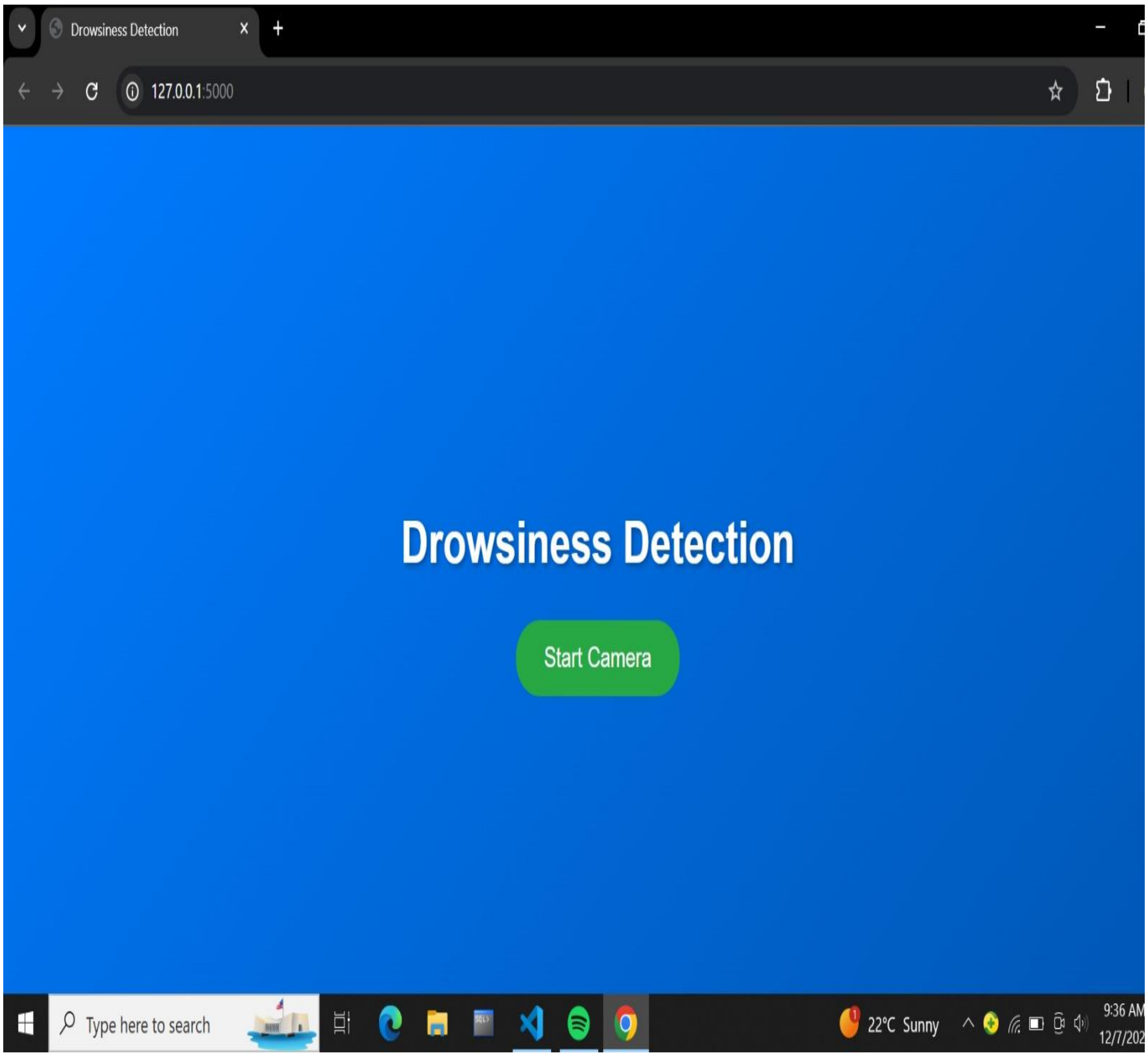
Chapter 5

Results and Discussion

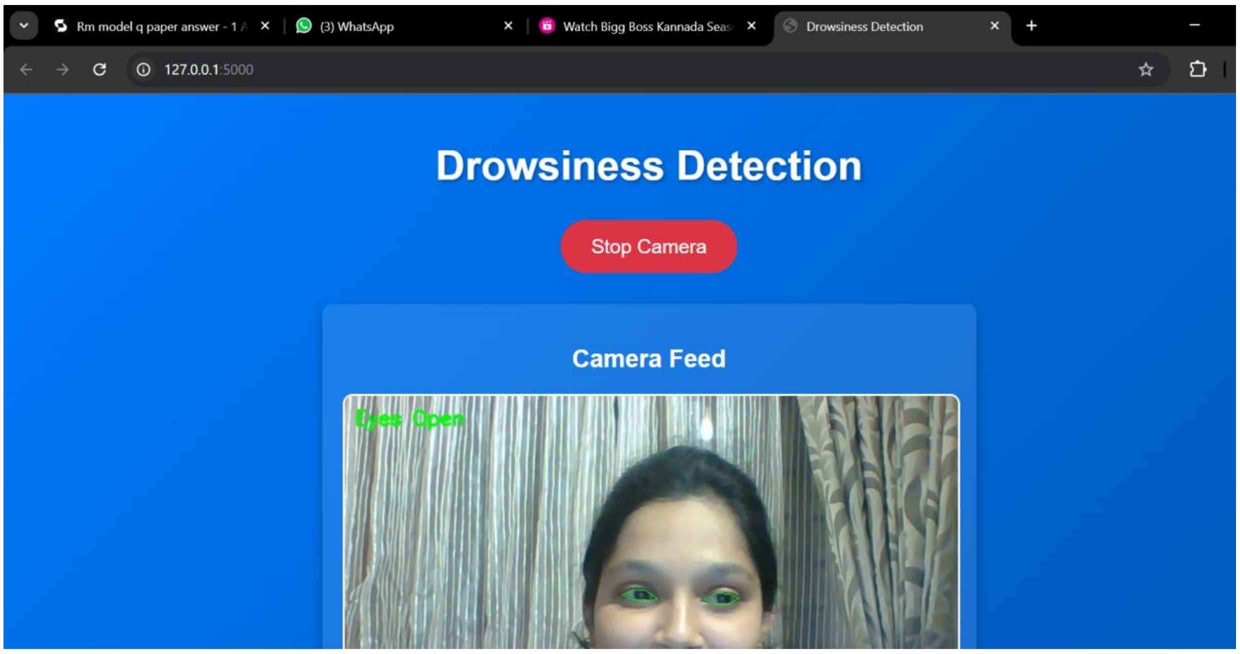
The code detects eye closure in real-time using Mediapipe's Face Mesh and raises an audio alert if the eyes remain closed. It streams the processed video feed with visual an notations through a Flask-based web interface.

5.1 Results

Snapshots of the Project with description:

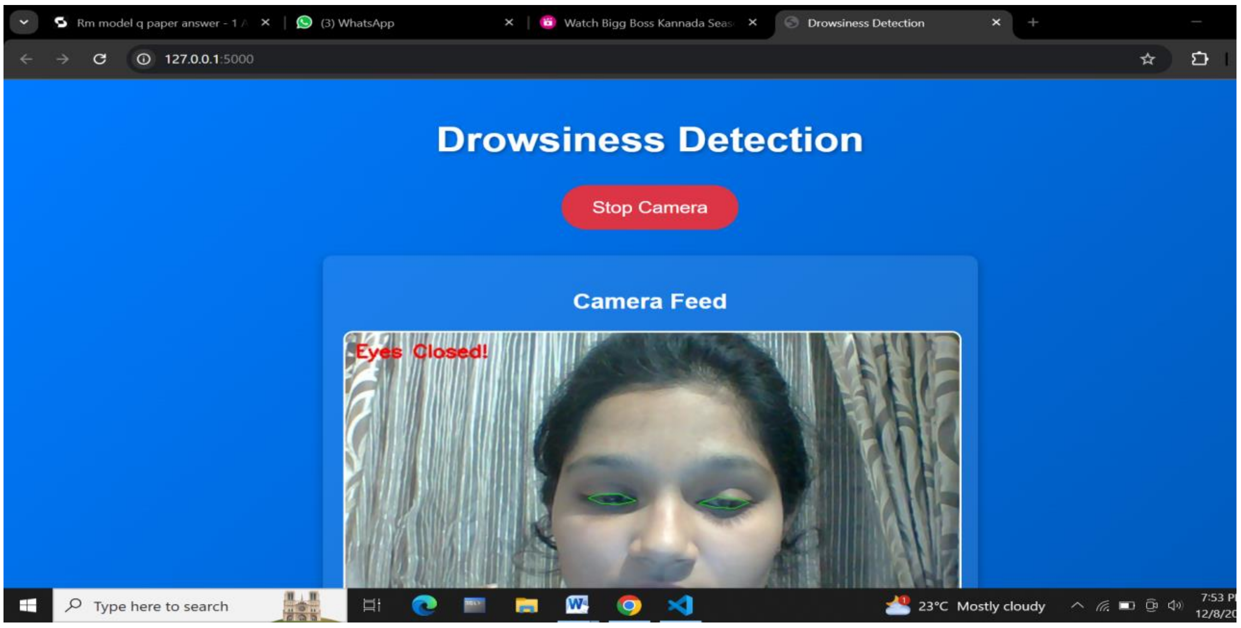


Snapshot 5.1.1: It is a web page for a “Drowsiness Detection System” with a user interface is used to toggle a camera feed on and off.



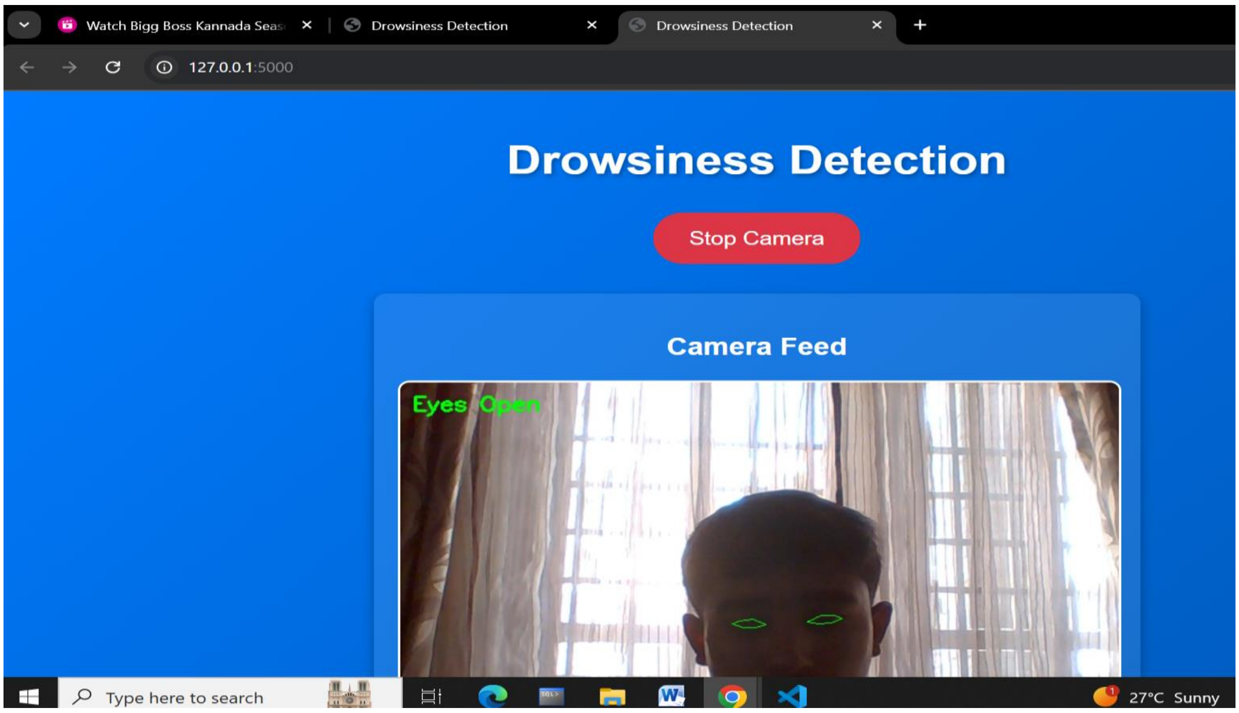
Snapshot 5.1.2: Frame for test1 user

"The snapshot captures a user1 with their eyes open, indicating they are alert. As a result, no alert message is displayed, confirming that the individual is not drowsy."



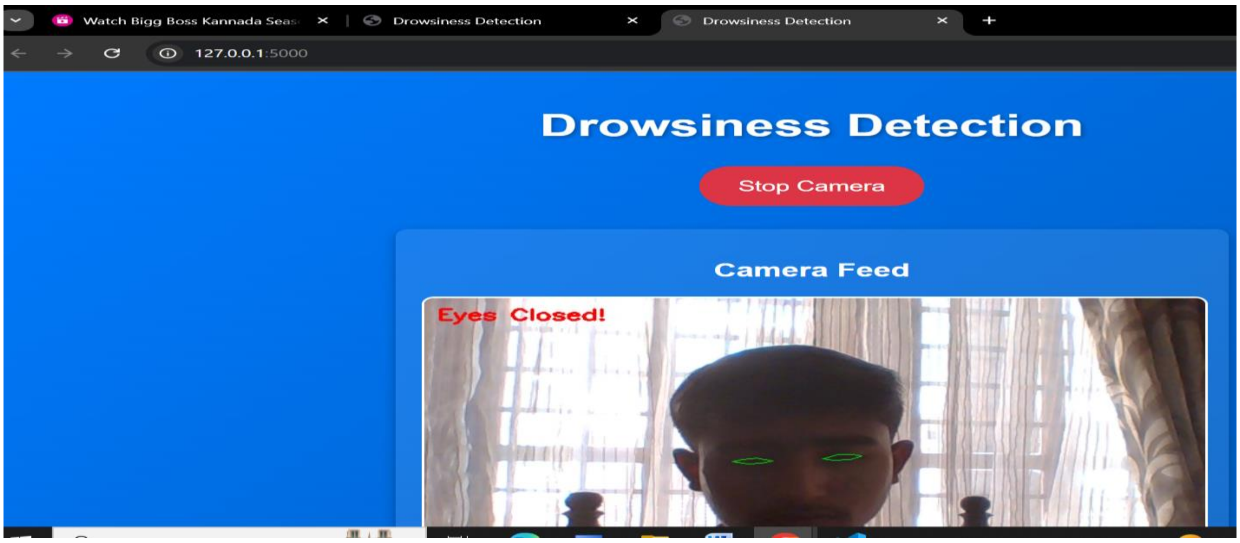
Snapshot 5.1.3: user 1 result

“The snapshot shows a user1 with their eye closed for over two seconds, triggering the alert message.This indicates the detection of potential drowsiness, prompting the necessary alert”.



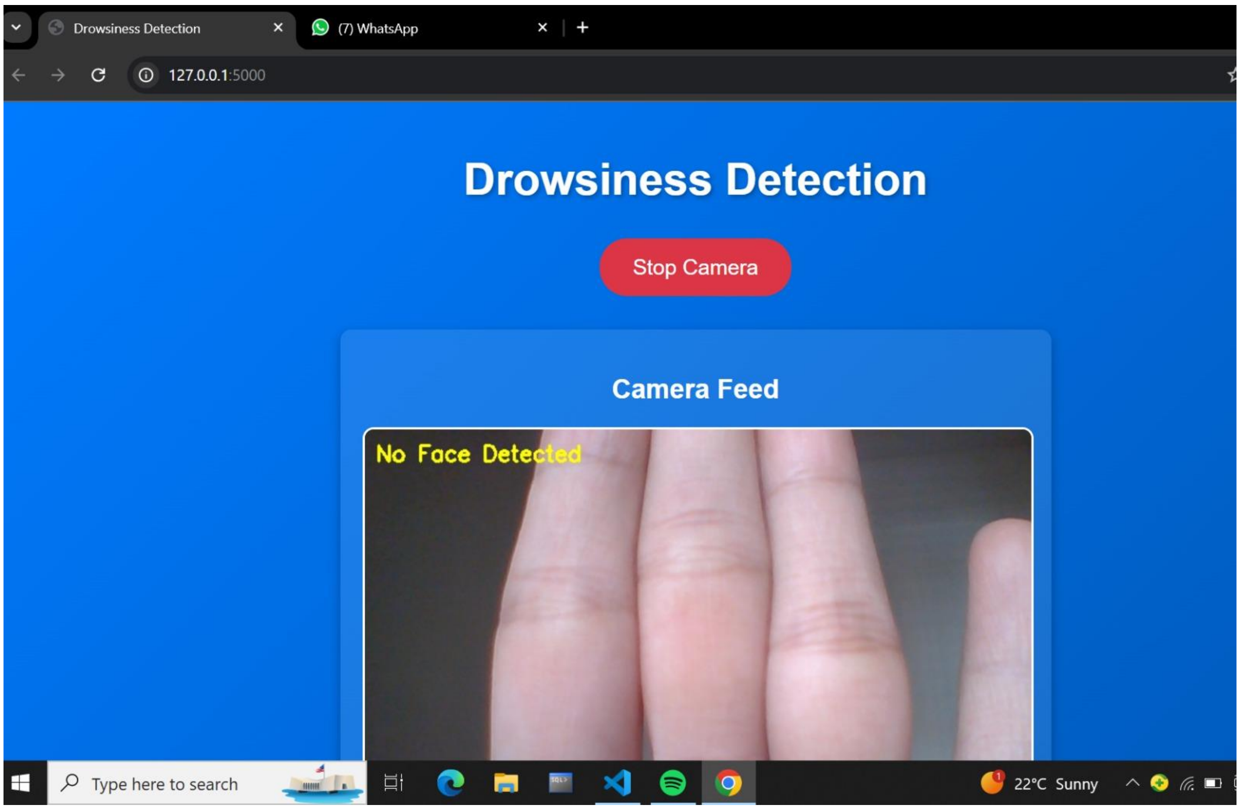
Snapshot 5.1.4: Frame for test2 user

"The snapshot captures a user2 with their eyes open, indicating they are alert. As a result, no alert message is displayed, confirming that the individual is not drowsy."



Snapshot 5.1.5: user2 result

“The snapshot shows a user2 with their eye closed for over two seconds, triggering the alert message.This indicates the detection of potential drowsiness, prompting the necessary alert”.



Snapshot 5.1.6: No Face Detected

If a hand is placed in front of the camera, the system fails to detect a face, and no eye state monitoring or alert is triggered.

5.2 Discussion

5.2.1Challenges Encountered:

- **Occlusions:** Placing a hand or object in front of the camera prevents proper face detection.
- **Lighting Conditions:** Performance is affected in low-light or overly bright environments.
- **Facial Variations:** Glasses, masks, or rapid head movements can interfere with detection.

5.2.2 Limitations of the current system:

- **Single Face Detection:** The system only tracks one face at a time.
- **Fixed EAR Threshold:** Not adaptive to different users' facial structures.
- **No Fatigue Analysis:** Only detects eye closure, without considering blinking patterns or head movements.
- **Limited Platform Support:** Designed for desktop use; not optimized for mobile or edge devices.

Chapter 6

Conclusion and Future Enhancements

6.1 Conclusion

- The code successfully implements real-time **eye state detection** using Mediapipe Face Mesh and Eye Aspect Ratio (EAR). It raises an alert when eyes remain closed, making it suitable for applications like **drowsiness detection**. The integration with Flask enables live video streaming through a web interface, making it versatile and user-friendly.

6.2 Future Enhancements

- **Multi-Face Support**: Extend detection to multiple faces for group monitoring.
- **Mobile Compatibility:** Deploy the system on mobile or edge devices for portability.
- **Fatigue Detection**: Integrate blinking patterns and head position analysis for advanced fatigue detection.
- **Data Logging**: Record alerts and events for analysis and reporting.
- These enhancements can expand the code's usability across industries like transportation, healthcare, and security.

REFERENCES

Citation Format:

- **Books**: James D Foley, Andries Van Dam, Steven K Feiner, John F Huges Computer graphics with OpenGL: Pearson education.
- **Journal Articles**: Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version,3rd/4th Edition, Pearson Education,2011.
- **Online Resources**:
  - https://chat.openai.com/
  - https://github.com/AnshumanSrivastava108/Real-Time-Drowsiness-Detection-System/blob/main/drowsiness_yawn.py
- **Software Tools**:
  - OpenCV
  - Pygame
  - Mediapipe