Drowsiness Detection System.

Revolutionizing urban mobility through real-time AI analysis and prediction.

Submitted By:-
Lubhit khandelwal (E23CSEU1689)
Md Abdul Ashiq (E23CSEU1725)
Vaibhav Pruthi (E23CSEU1681)

Submitted to:
Dr. Purushottam Kumar

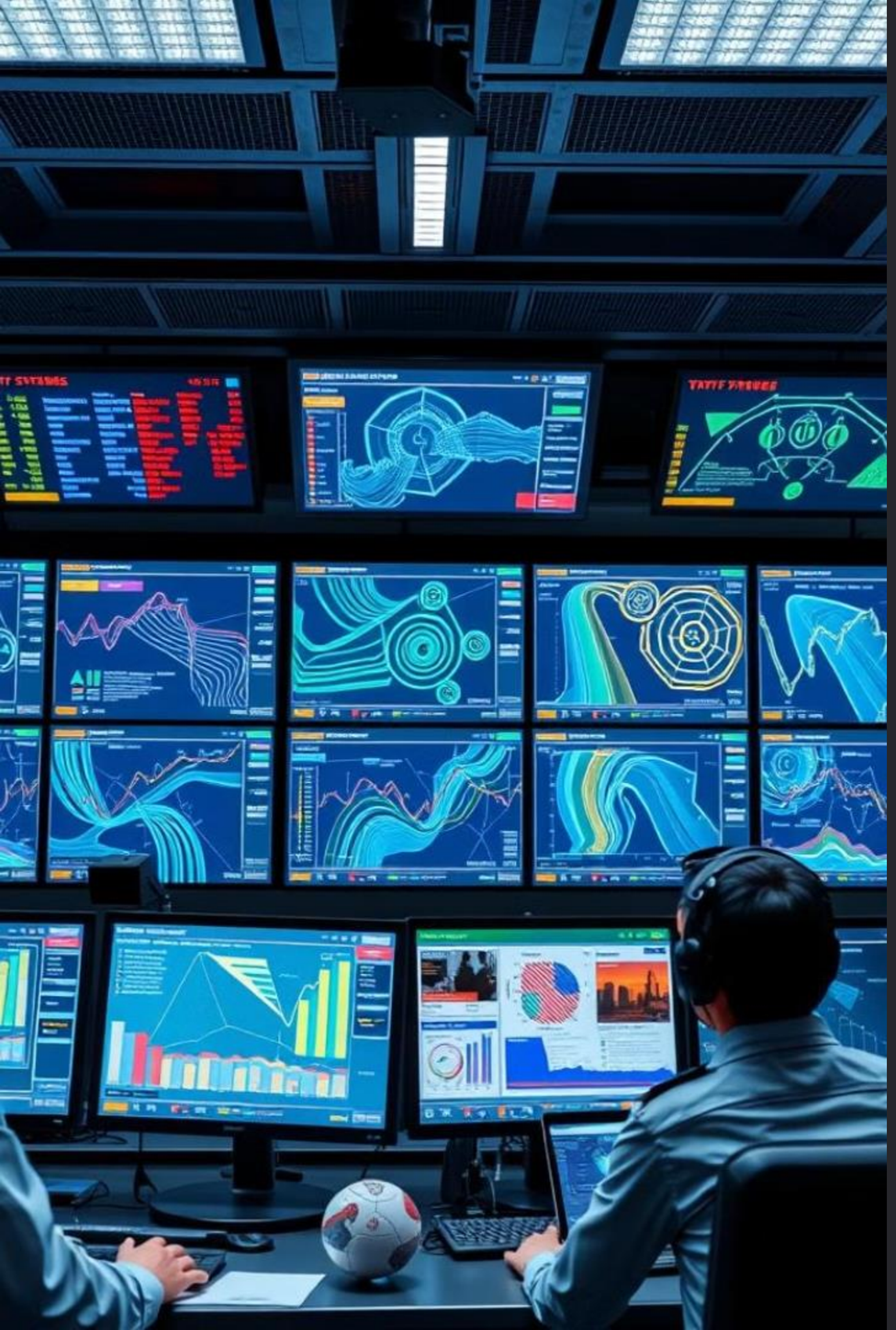
Table of Content

Introduction: The Problem of Drowsiness

1. **Risk Factor**
   Drowsiness is a major cause of accidents

2. **Causes**
   Fatigue, sleep deprivation, and monotony can lead to drowsiness.

3. **Consequences**
   Drowsiness can impair reaction time, judgment, and vigilance



The Solution: Our Drowsiness Detection System

Real-Time Monitoring

Alerting Mechanism

The system continuously analyzes driver behavior to detect signs of drowsiness.
It triggers alerts to warn the driver when drowsiness is detected.

Adaptive Learning

The system learns the driver's unique patterns to provide more personalized alerts.

Workflow Pipeline

1

Camera

Captures driver's facial expressions and eye movements.

2

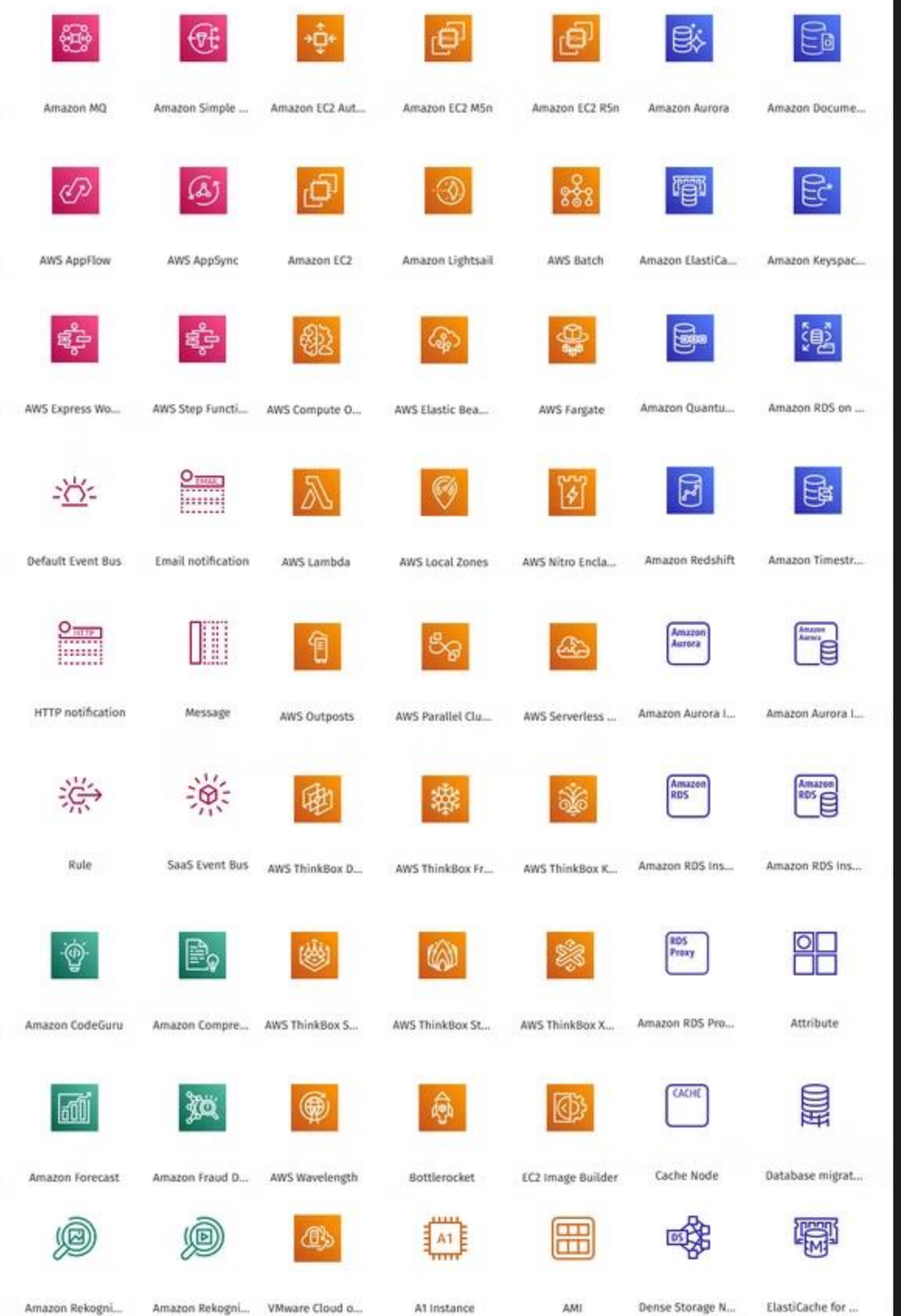
AI Algorithms

Analyze data from the camera and other sensors to detect drowsiness.

3

Alerting System

Notifies the driver through visual and auditory cues.



Technology Stack

```
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
```

```
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
```

- Tkinter for GUI elements.
- Scipy and Imutils for geometric calculations and landmark manipulation.
- Dlib for facial detection and landmark prediction.
- Pygame for playing alert sounds.
- Twilio for sending SMS alerts.

```python
# 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)
```

- Pygame Mixer: Initializes Pygame's mixer module to load and play an audio file for alert sound when drowsiness is detected.
- Twilio Client: Sets up Twilio credentials and client for sending SMS alerts when drowsiness is detected.
- This section prepares both audio and SMS services needed for alerts.

```python
# 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
```

- Calculates the Eye Aspect Ratio (EAR) by measuring the eye's vertical and horizontal distances.
- EAR helps detect whether eyes are closed, a common sign of drowsiness.
- This function is key to detecting drowsiness by checking if the EAR drops below a certain threshold.

```python
# 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
    )
    print(f"SMS sent: {message.sid}")
    sms_status_label.config(text="Alert SMS sent!", fg="blue")  # Update SMS status label
```

- Sends an SMS alert through Twilio when drowsiness is detected.
- Updates the GUI to show an “Alert SMS sent!” message.
- This function ensures an external notification is sent if drowsiness is detected.

```python
# Drowsiness detection parameters
thresh = 0.25  # EAR threshold for drowsiness detection
frame_check = 20  # Consecutive frames required to trigger drowsiness alert
flag = 0  # Frame counter for continuous drowsiness detection

# Initialize face detector and facial landmarks predictor using Dlib
detect = dlib.get_frontal_face_detector()
predict = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")

# Eye landmark indices in the 68-point facial landmarks model
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS["right_eye"]
```

- Sets thresholds for drowsiness detection:
- EAR Threshold: Minimum EAR for eyes to be considered open.
- Frame Check: Number of consecutive frames required to confirm drowsiness.
- Initializes Dlib’s face detector and facial landmark predictor for accurate face and eye tracking.
- Defines landmark indices for left and right eyes.

```
# Set up webcam feed
cap = cv2.VideoCapture(0)

# Tkinter GUI setup
root = tk.Tk()
root.title("Drowsiness Detection System")
root.geometry("800x600")
root.config(bg="#282828")

# Video feed display in GUI
video_frame = tk.Label(root, bg="#282828")
video_frame.pack(pady=20)

# Status label for drowsiness monitoring
status_label = tk.Label(root, text="Status: Monitoring...", font=("Helvetica", 16, "bold"), fg="green", bg="#282828")
status_label.pack(pady=10)

# SMS alert status label
sms_status_label = tk.Label(root, text="", font=("Helvetica", 14), fg="blue", bg="#282828")
sms_status_label.pack(pady=10)
```

- Creates a Tkinter window with a video frame, status label, and SMS status label.
- The video_frame displays live webcam feed, status_label shows monitoring status, and sms_status_label shows the SMS alert status.

```python
# 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
```

```python
        # 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

        # 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
```

- Captures video frame and processes it to detect faces and eyes.
- Calculates EAR and checks if it falls below threshold to detect drowsiness.
- If drowsiness is detected:
- Updates status_label to alert.
- Plays alert sound and sends SMS.
- Draws eye contours for visual feedback.
- Converts frame for GUI display and schedules next update every 10 ms.

```python
# 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()
```

- Stop Detection function releases camera and closes the GUI.
- Update Frame initiates the continuous video feed and drowsiness monitoring.
- Stop Button provides a way to manually end the program.

Advantages: Improved Safety and Productivity

1

Reduced Accidents

By detecting drowsiness, the system helps prevent accidents caused by fatigue.

2

Enhanced Driver Awareness

The alerts encourage drivers to take breaks when needed, improving alertness.

3

Improved Productivity

By preventing fatigue-related accidents, the system enhances driver efficiency.

Next Steps

Enhanced Data Sources

Integrate weather, event data for comprehensive analysis.

Smart City Integration

Collaborate with city planners for wider implementation.

Machine Learning Optimization

Refine AI models for even greater accuracy.

Public Engagement

Develop user-friendly apps for citizen participation.

Thank You