Driver Drowsiness Detection System

BS Software Engineering

Department of Software Engineering

Capital University of Science and Technology, Islamabad

Driver Drowsiness Detection System

Malik Hamza Nawaz	**(BSE221044)**
Javeria Khalid	**(BSE221001)**

Supervised By

Ma'am Syeda Hafsa Ali

Fall 2024

Bachelor of Software Engineering (BSSE)

Department of Software Engineering

Capital University of Science & Technology, Islamabad

Project Report

VERSION	V 1.4	**NUMBER OF MEMBERS**	2

TITLE	Driver Drowsiness Detection System
SUPERVISOR NAME	**Syeda Hafsa Ali**

MEMBER NAME	REG. NO.	EMAIL ADDRESS
Malik Hamza Nawaz	BSE221044	Bse221044@cust.pk
Javeria Khalid	BSE221001	Bse221001@cust.pk

MEMBERS' SIGNATURES

Malikhamza

Javeriakhalid

Supervisor's Signature

Note 1: This paper must be signed by your supervisor
Note 2: The soft-copies of your project report, source codes, schematics, and executable should be delivered in a CD

Approval Certificate

This project, entitled as “Driver Drowsiness Detection System ” has been approved for the award of

Bachelors of Science in Software Engineering

Committee Signatures:

Supervisor: ___________________________

(Ms. Syeda Hafsa Ali)

Project Coordinator: ___________________________

(Mr. Ibrar Arshad)

Head of Department: ___________________________

(Dr. Nadeem Anjum)

Declaration

I/We, hereby, declare that “No portion of the work referred to, in this project has been submitted in support of an application for another degree or qualification of this or any other university/institute or other institution of learning”. It is further declared thatthis undergraduate project, neither as a whole nor as a part thereof has been copied out from any sources, wherever references have been provided.

MEMBERS’ SIGNATURES

Malikhamza

Javeriakhalid

Acknowledgements

It is usual to thank those individuals who have provided particularly useful assistance, technical or otherwise, during your project. Your supervisor will obviously be pleased to be acknowledged as he or she will have invested quite a lot of time overseeing your progress.

Dedication

Executive Summary

The **Drowsiness Detection System** is a real-time monitoring solution designed to ensure the safety and well-being of individuals by detecting signs of fatigue, such as eye closure and yawning. Using a webcam and advanced computer vision technology, the system analyzes facial landmarks in real-time to determine key metrics, such as the Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR). These metrics are critical in identifying potential drowsiness, with thresholds set to trigger warnings when detected. The system uses **MediaPipe** for accurate facial landmark detection and **OpenCV** for video processing, while **pyttsx3** provides auditory feedback by alerting users when drowsiness or yawning is detected. This proactive approach aims to reduce fatigue-related accidents and enhance overall alertness during activities requiring attention, such as driving or long hours of work. The system is designed to be user-friendly, offering both visual and audio alerts to keep users informed and safe.

Table of Contents

List of Figures

LIST OF TABLES

This page is kept blank

Chapter 1

1. Introduction

Driver fatigue is a leading cause of road accidents, with falling asleep behind the wheel accounting for thousands of fatalities each year. Long hours, monotonous driving, and insufficient rest impair a driver's alertness, increasing the risk of collisions. A Driver Drowsiness Detection System offers a proactive solution by monitoring signs of fatigue, such as eye closure or head movements, and issuing timely alerts. By identifying drowsiness early, this technology helps prevent accidents, saving lives and ensuring safer roads.

1.1. Project Introduction

The project aims to develop a Driver Drowsiness Detection System with advanced features to prevent accidents caused by fatigue. The system will monitor drivers' real-time behaviors, such as eye movement and head movements, to detect signs of drowsiness. Upon detecting fatigue, it will issue alerts to keep the driver awake and attentive. This solution will significantly enhance road safety by reducing the risk of accidents caused by falling asleep at the wheel, benefiting both drivers and passengers.

Module 1: Face Detection

During this phase, the focus is on developing the core features of our system, which is the implementation of the facial detection, and it consists of the detection of eye and mouth dilation through **media pipeline**, which is an **open source framework** in **python** and uses a **Euclidian function** that calculates the distance between two points and it will calculate the ratios of length and width of the mouth and eyes opening and if the ratio increases it will show detection by indicating that the driver is drowsy and is yawning and will create an alert and if its ratio decreases it will indicate that the driver may be sleepy or has fallen asleep and will generate an alert or buzzer. The system will ensures it doesn’t falsely trigger drowsiness detection for regular eye blinks by setting appropriate thresholds.

Module 2: Cloud Server (Firebase)

During the second phase we need to connect our code with a cloud based server we will be using fire base as our database for this purpose we will simply go to fire base and will create an empty project there and a j-son file will be generated and which we will download and will attach to our code to enable communication with the database and now every image or detection result processed by the system will be stored in the Firebase database,this includes details such as timestamps, images, or detection outcomes.Over time, as the system processes more detections, a dataset will be built in Firebase. This dataset could be used for further analysis, training machine learning models, or reporting.

1.2. Existing Examples / Solutions

BOSCH	**Bosch Driver Drowsiness Detection System**	Developed by Bosch, this system is integrated into modern vehicles to enhance road safety. It uses advanced sensors to monitor driver behavior and provide timely alerts. • The system monitors over 70 signals, including steering patterns, lane deviations, and driving duration. • By analyzing this data, it calculates a "tiredness index."
netradyne	**Netradyne Driver Monitoring System**	Netradyne's Driver Monitoring System (DMS) is a cutting-edge technology designed to enhance driver safety by identifying signs of fatigue and distraction in real-time. It is primarily targeted at commercial fleet vehicles to improve overall road safety and reduce accidents.

Table 1- Existing Examples

1.3. Business Scope

The project focuses on providing drivers and fleet operators with a reliable solution for detecting and mitigating drowsiness through an intelligent Driver Drowsiness Detection System. The aim is to address the critical challenges faced by drivers during extended hours on the road, ensuring safety and reducing the risk of accidents.

By concentrating on this target audience, the project seeks to understand the behaviors, patterns, and risks associated with driver fatigue. The primary objective is to deliver a system that not only enhances driver alertness but also integrates seamlessly into vehicles, providing real-time alerts and insights.

This solution will enable individuals and businesses to improve road safety, optimize driving efficiency, and foster a culture of responsible driving. The system's design and features aim to empower users with actionable feedback, making roads safer and journeys more secure for all stakeholders involved.

1.4. Useful Tools and Technologies

1.3.1 Programming Language:

Prototypes:

- Figma

Front-end:

- HTML & CSS
- JavaScript

Back-end:

- Python

1.3.2 Database

Google Firebase is a Google-backed application development software that allows developers to develop IOS, Android and Web apps. Firebase provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and products experiment. Hence, we will be using Fire Base for storing data on cloud.

1.3.3 Libraries:

1. **OpenCV (opencv-python):**

For real-time image and video processing, including accessing the camera and detecting facial features.

Example use: Capturing video feed, preprocessing images.

2. **MediaPipe:**

For detecting and tracking facial landmarks, including eyes and mouth.

Example use: Locating specific facial points to calculate dilation ratios.

3. **NumPy:**

For numerical computations, such as calculating the Euclidean distance between facial landmarks.

Example use: Performing mathematical operations on arrays to compute ratios.

4. **Math:**

A standard Python library for mathematical functions like square root, trigonometric functions, etc.

Example use: Calculating Euclidean distances or other mathematical ratios

5. **Firebase Admin SDK (firebase-admin):**

For integrating your system with Firebase, allowing you to upload and retrieve data from the cloud.

Example use: Authenticating and storing detection data like images or logs.

6. **JSON:**

For handling Firebase configuration files and other data serialization needs.

Example use: Reading the Firebase configuration file (JSON) to establish database connectivity.

7. Requests:

For making HTTP requests, which might be needed for interacting with Firebase REST APIs if you prefer direct API calls instead of using the SDK.

Example use: Sending or retrieving data from the Firebase database.

Chapter 2

2. Requirement Specification and Analysis

2.1. Functional Requirements

S. No.	Functional Requirement	Type	Status
1	System shall record the user using the camera.	Input	Mandatory
2	System shall generate an audio alert if requirements for drowsiness are being fulfilled.	Output	Mandatory
3	System shall detect eye movements correctly.	Processing	Mandatory
4	System shall detect mouth movements correctly.	Processing	Mandatory
5	System shall generate an automatic audio alert before any sort of shutdown.	Systematic	Mandatory

i) Functional Requirements

2.2. Non-Functional Requirements

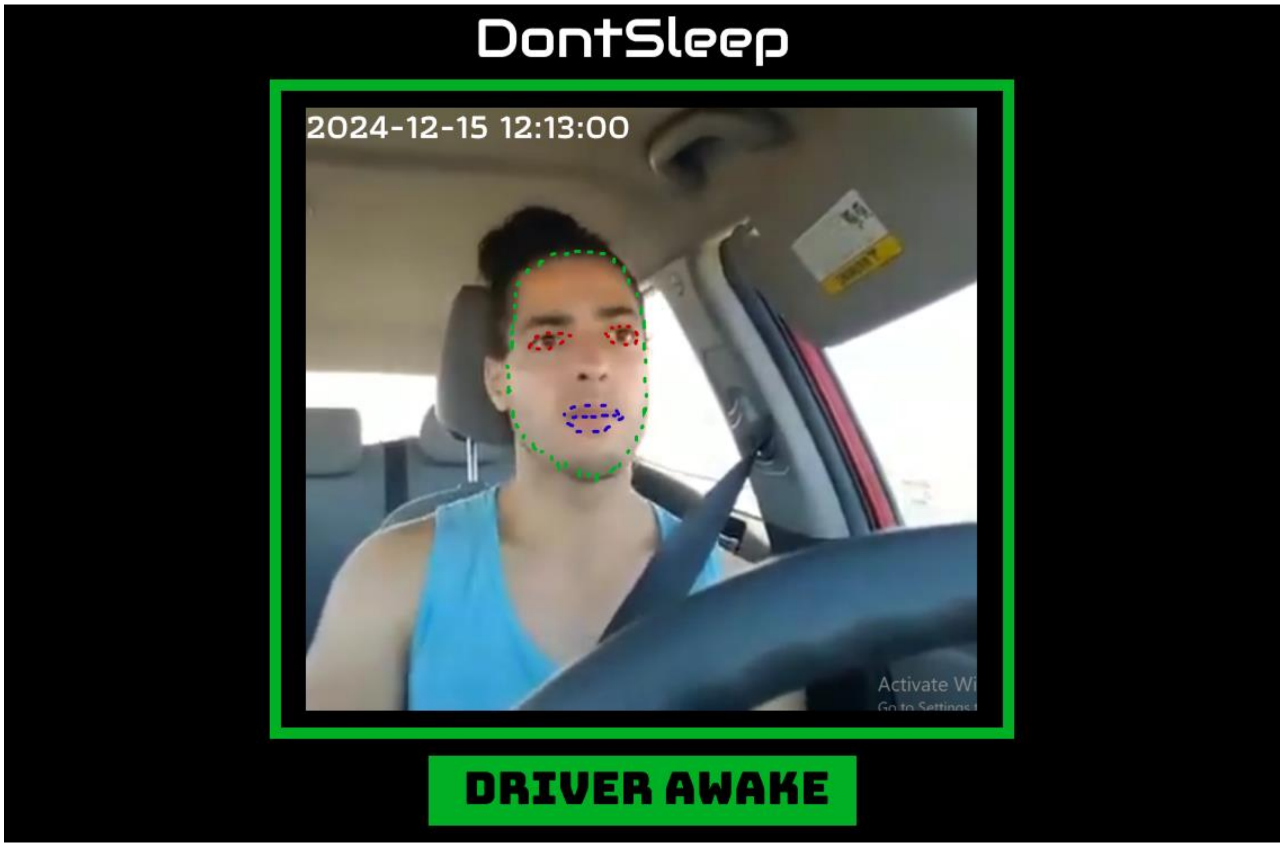
S. No.	Non Functional Requirements	Category
1	System shall not experience more than 1 second of lag or delay.	Availability
2	System shall not experience more than 5 seconds of downtime.	Availability
3	System shall implement correct drowsiness detection	Reliability

	algorithm.	
4	System shall detect drowsiness within 3 seconds of monitoring eye or mouth.	Performance
5	System shall protect collected driver data by strong encryption techniques.	Security
6	System shall maintain accurate drowsiness detection regardless of lighting variations inside the vehicle.	Reliability
7	System shall have a simple interface with minimal distractions for the driver driving the vehicle.	Usability

ii) Non-Functional Requirement

2.3. User Interface Design (Prototypes)

2.3.1. DontSleep UI (Driver Awake Page)

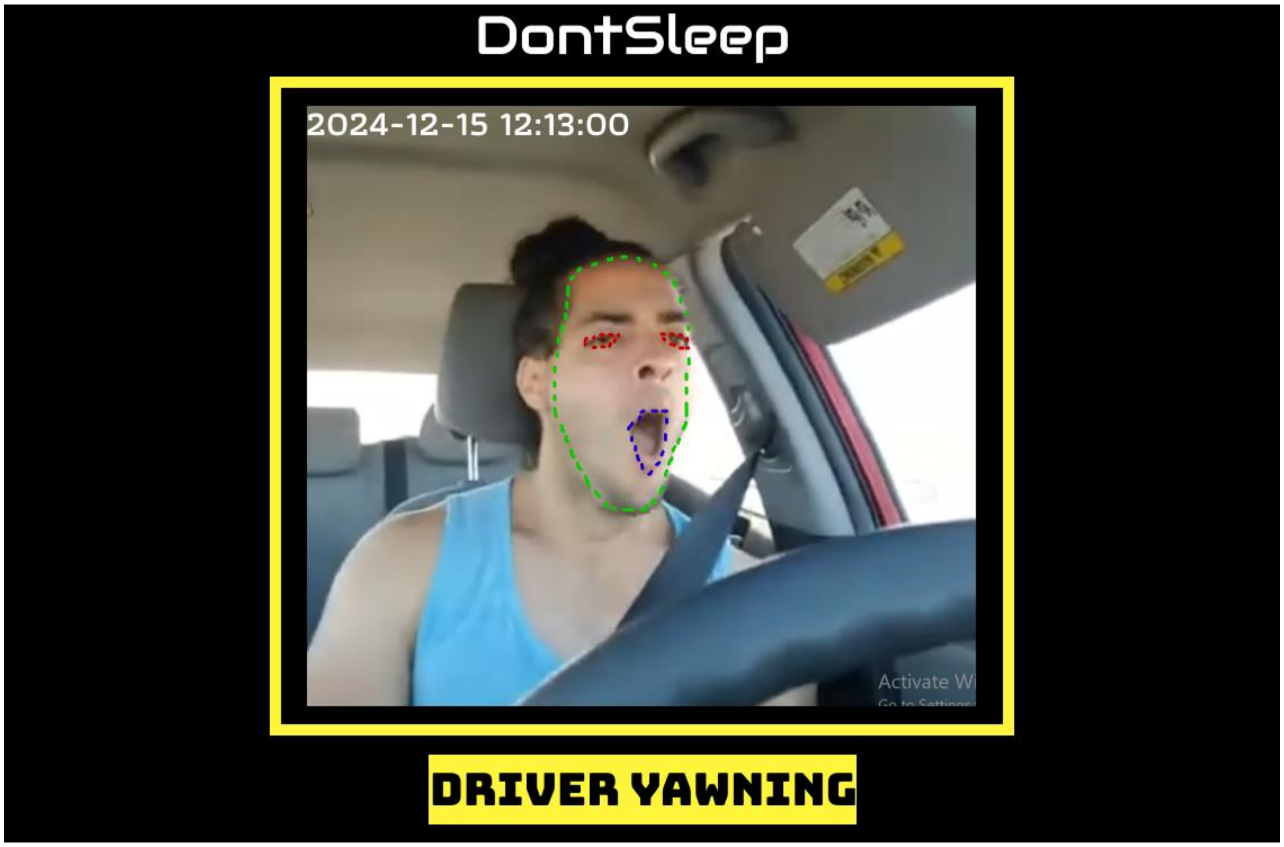


iii Driver Awake UI Prototype

The mini-screen connected to the camera (mounted on the dashboard) shows the company name (DontSleep) at the top, below that it shows the real-time view being captured by the camera / dash-cam with the current date and time.

The boundary surrounding the camera feed is green, meaning the driver is fully awake and there is no sign of drowsiness. Similarly, the rectangle block below the camera feed is also green and shows “Driver Awake”.

2.3.2. DontSleep UI (Driver Yawning Page)

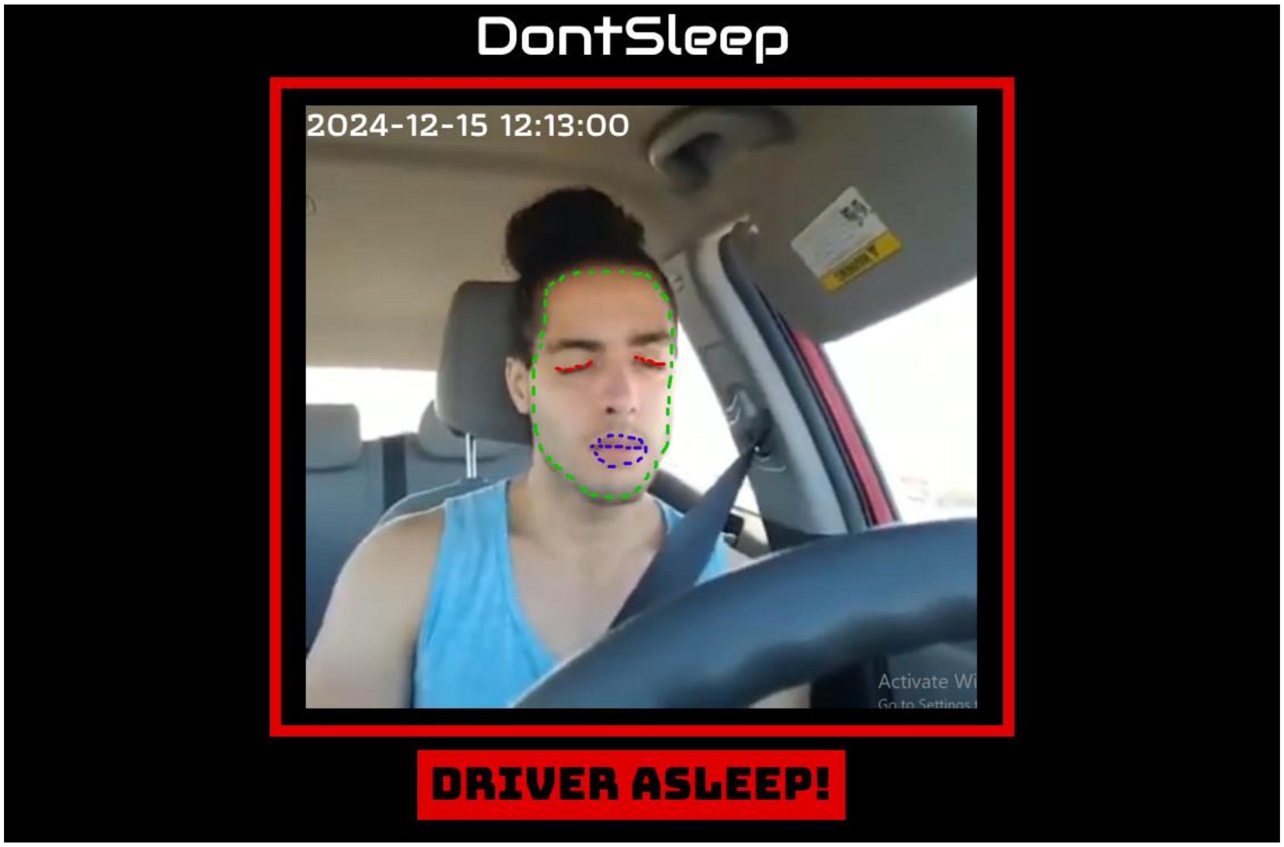


iv Driver Yawning UI Prototype

The second page also shows the company name (DontSleep) at the top, below that it shows the real-time view being captured by the camera / dash-cam with the current date and time.

The boundary surrounding the camera feed is yellow, meaning the driver is not fully awake and due to the fact that the system is watching the driver “yawn”, it is in warning state. Similarly, the rectangle block below the camera feed is also yellow and shows “Driver Yawning”. At this point, the system is on standy to send an error.

2.3.3. DontSleep UI (Driver Asleep Page)



v Driver Asleep UI Prototype

The third screen also shows the company name (DontSleep) at the top, below that it shows the real-time view being captured by the camera / dash-cam with the current date and time.

The boundary surrounding the camera feed is now red, meaning the driver is drowsy according to the system’s judgement and so it has turned the boundary and the rectangle block below the camera feed red and is showing “Driver Asleep”. In addition to this, the system also sends out an audio alerts / alarm.

2.4. Dataset

2.4.1. Explanation

The dataset we have selected is Ismail Nasri's Driver Drowsiness Dataset (DDD) from Kaggle. This specific dataset follows the following research paper: https://link.springer.com/chapter/10.1007/978-981-33-6893-4_6 and was last updated 3 years ago.

2.4.2. Variables

The variables from our dataset are as following:

- **Eye Aspect Ratio (EAR)**
  EAR is measuring the Euclidean distance between the landmarks of each eye to determine if they are open or closed.

- **Mouth Aspect Ratio (MAR)**
  MAR is measuring the Euclidean distance between the landmarks of mouth to determine if the mouth is open or closed.

- **Lighting**
  The lighting inside the car, as the system should be capable of detecting the eye and mouth aspect ratio even in poor lighting.

- **Frame ID**
  The ID that defines the order of the number of images (frames) in a video sequence.

2.4.3. Labels

The labels from our dataset are as following:

- **Driver State**
  - **Awake:**
    - **Eye Behavior**

      Open
    - **Mouth Behavior**

      Closed
  - **Yawning:**
    - **Eye Behavior**

      Open
    - **Mouth Behavior**

      Open
  - **Asleep:**
    - **Eye Behavior**

      Closed
    - **Mouth Behavior**

      Closed

2.4.4. Metadata

The metadata from our dataset are as following:

- **Timestamp**

  The time and date of when the image was captured, this helps define the order and sequencing of frames (data) so we can consider it as metadata.

- **Driver ID**

  Each collection of frames of a single driver are identified by a unique driver ID, for the same purpose of efficient order and sequencing so it can be considered metadata.

- **File Name**

  The file names have a proper structure with proper driver ID, frame ID and driver state so this is also considered metadata.

- **Image Format**

  There can be different image formats (png or jpeg etc) so this also defines our dataset and its data so it can be considered as metadata.

2.5. Framework

DontSleep (Driver Drowsiness Detection System) aims to detect driver fatigue using image processing and machine learning techniques. This section outlines the necessary steps, preprocessing techniques, core processing tasks, and integration components, along with the technologies and tools used.

2.5.1. Pre-processing Steps

The Preprocessing of our project includes the collection of raw image data for accurate detection and analysis. The key steps we will be following are:

i. Data Collection:

We will download the datasets from Kaggle containing images of various facial expressions, such as **open eyes, yawning and closed eyes.**

ii. Data Cleaning:

We will remove irrelevant, corrupted, or duplicate images to ensure high-quality and efficient data.

iii. Data Labeling:

We will label each of the images with their corresponding tags like "open eyes", "closed eyes".

2.5.2. Storage and Integration:

To manage and store data effectively, we will implement following steps:

- **Firebase Setup:**

  Configure a Firebase project to handle cloud-based data storage.

- **Data Storage:**

  Store pre-processed data in the Firebase. Moreover, the facial landmark outputs and drowsiness detection results will also be stored in the Firebase.

- **Data Retrieval:**

  The system will retrieve stored data in real-time to perform:

    - Real-time monitoring.
    - Model training and performance evaluation.
    - Any further analysis and visualization.

2.5.3. Processing Steps:

These steps involves real-time analysis of the input to detect signs of driver drowsiness:

1. **Face Detection:**

MediaPipe's Face Detection library will be used to identify and isolate the driver's face in each frame.

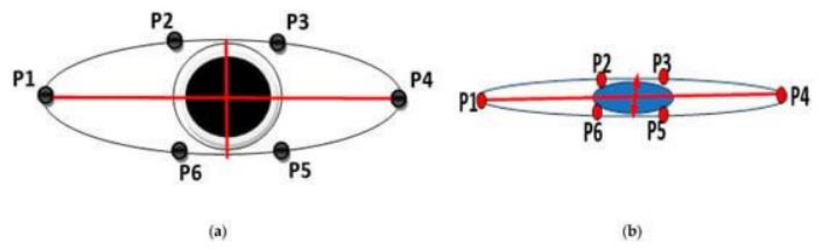
2. **Facial Landmark Detection:**

Detection of the key facial landmarks (eyes and mouth) will take place using MediaPipe's Facial Landmark Detection.

3. **Eye and Mouth State Detection:**

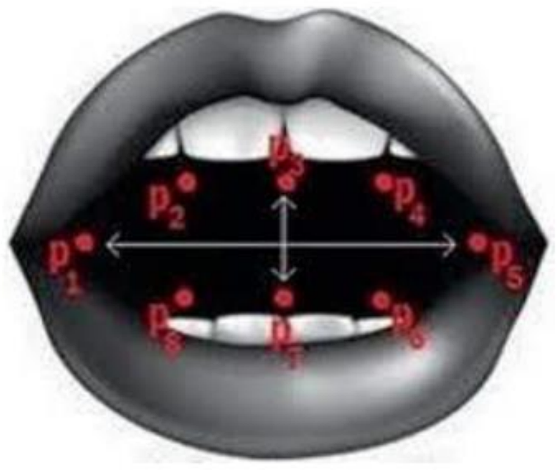
Here we will Implement algorithms to analyze facial landmarks:

- **Eye Aspect Ratio (EAR):** Calculate the Euclidean distance between eye landmarks to detect if eyes are open or closed.



vi Eye Aspect Ratio

- **Mouth Aspect Ratio (MAR):** Measure the distance between mouth landmarks to determine if the mouth is open (indicating yawning).



vii Mouth Aspect Ratio

4. **Driver Drowsiness Detection:**

   Finally, we will combine the outputs of eye and mouth state detection to determine if the driver is drowsy and an alert will be generated when drowsiness is detected.

2.6. Tools and Technologies:

- **Programming Language:** Python

- **Libraries and Frameworks:**

  MediaPipe: For face detection and landmark detection.

  OpenCV: For image processing and video analysis.

- **Cloud Platform / Database:**

  Firebase: For real-time data storage and retrieval.

2.7. Existing Example

The existing example we have chosen is:

https://github.com/Nightskull100/Drowsiness-Detection-using-Python

Chapter 3

3.1. Implementation

This chapter introduces how this software is implemented. Implementation contains all details which are required to make the system operational. We have also discussed the development, tools and technologies to implement the system.

The whole implemenataion of the system consist of further small chunks that makes it a whole system

3.1.1. Face Detection and Landmark Extraction

The **MediaPipe Face Mesh** model is initialized with the following parameters:

- max_num_faces=1: The system is designed to track only one face at a time.
- refine_landmarks=True: This option ensures more accurate detection of facial features.
- min_detection_confidence=0.5: The minimum confidence required to detect a face.
- min_tracking_confidence=0.5: The minimum confidence required to track the detected face.

Once the face is detected, **landmarks** are extracted, representing key points on the face. These landmarks include the eyes, eyebrows, nose, and mouth. For each landmark, its (x, y) position is normalized with respect to the width and height of the frame, allowing for accurate positioning regardless of the face's size or location within the image.

3.1.2. Code Snippet for Face Landmark Detection

```
mp_face_mesh = mp.solutions.face_mesh

face_mesh = mp_face_mesh.FaceMesh(max_num_faces=1, refine_landmarks=True,
min_detection_confidence=0.5, min_tracking_confidence=0.5)
```

```
results = face_mesh.process(rgb_frame)

if results.multi_face_landmarks:

    landmarks = results.multi_face_landmarks[0].landmark

    normalized_landmarks = [(int(lm.x * w), int(lm.y * h)) for lm in landmarks]
```

3.1.3. Eye Aspect Ratio (EAR)

The **Eye Aspect Ratio (EAR)** is a metric used to determine whether the eyes are closed, which can indicate drowsiness. The EAR is calculated based on the distance between six key landmarks around the eyes. If the EAR falls below a predefined threshold, the system considers the eyes to be closed.

3.1.4. Code Snippet for EAR Calculation

```
def aspect_ratio(landmarks, indices):

    A = np.linalg.norm(np.array(landmarks[indices[1]]) - np.array(landmarks[indices[5]]))

    B = np.linalg.norm(np.array(landmarks[indices[2]]) - np.array(landmarks[indices[4]]))

    C = np.linalg.norm(np.array(landmarks[indices[0]]) - np.array(landmarks[indices[3]]))

    return (A + B) / (2.0 * C)
```

3.1.5. Eye Closure Detection

If both the left and right eyes have an EAR below the threshold (EYE_AR_THRESHOLD = 0.2), the system considers the eyes closed and triggers an alert after a consecutive number of frames.

```
if left_eye_ar < EYE_AR_THRESHOLD and right_eye_ar < EYE_AR_THRESHOLD:

    eye_close_count += 1

    if eye_close_count >= EYE_AR_CONSEC_FRAMES:
        cv2.putText(frame, "Drowsiness Alert!", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
        engine.say("You are feeling drowsy. Please take a break.")
        engine.runAndWait()
        eye_close_count = 0
```

3.1.6. Yawning Detection

Yawning is another indicator of drowsiness. The **Mouth Aspect Ratio (MAR)** is used to detect yawning. If the mouth opens wider than a predefined threshold, the system identifies a yawn and alerts the user.

3.1.7. Code Snippet for MAR Calculation and Yawn Detection

```
mouth_ar = aspect_ratio(normalized_landmarks, MOUTH)
if mouth_ar > YAWN_AR_THRESHOLD:
    cv2.putText(frame, "Yawning Detected!", (50, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
    engine.say("Yawning detected. Please stay alert.")
    engine.runAndWait()
```

3.1.8. Text-to-Speech Alerts

The **pyttsx3** library is used to convert the detected drowsiness or yawning events into auditory alerts. Upon detection, a corresponding message is spoken aloud to remind the user to stay alert.

3.1.9. Code Snippet for Text-to-Speech

```
engine = pyttsx3.init()
engine.say("You are feeling drowsy. Please take a break.")
```

engine.runAndWait()

3.1.10. Displaying the Results

For each frame, rectangles are drawn around the face, eyes, and mouth to provide a visual representation of the detected features. The system also displays text alerts on the screen when drowsiness or yawning is detected.

3.1.11. Code Snippet for Drawing Rectangles

cv2.rectangle(frame, (x_min, y_min), (x_max, y_max), (0, 255, 0), 2)

cv2.rectangle(frame, (left_eye_x_min, left_eye_y_min), (left_eye_x_max, left_eye_y_max), (0, 255, 0), 2)

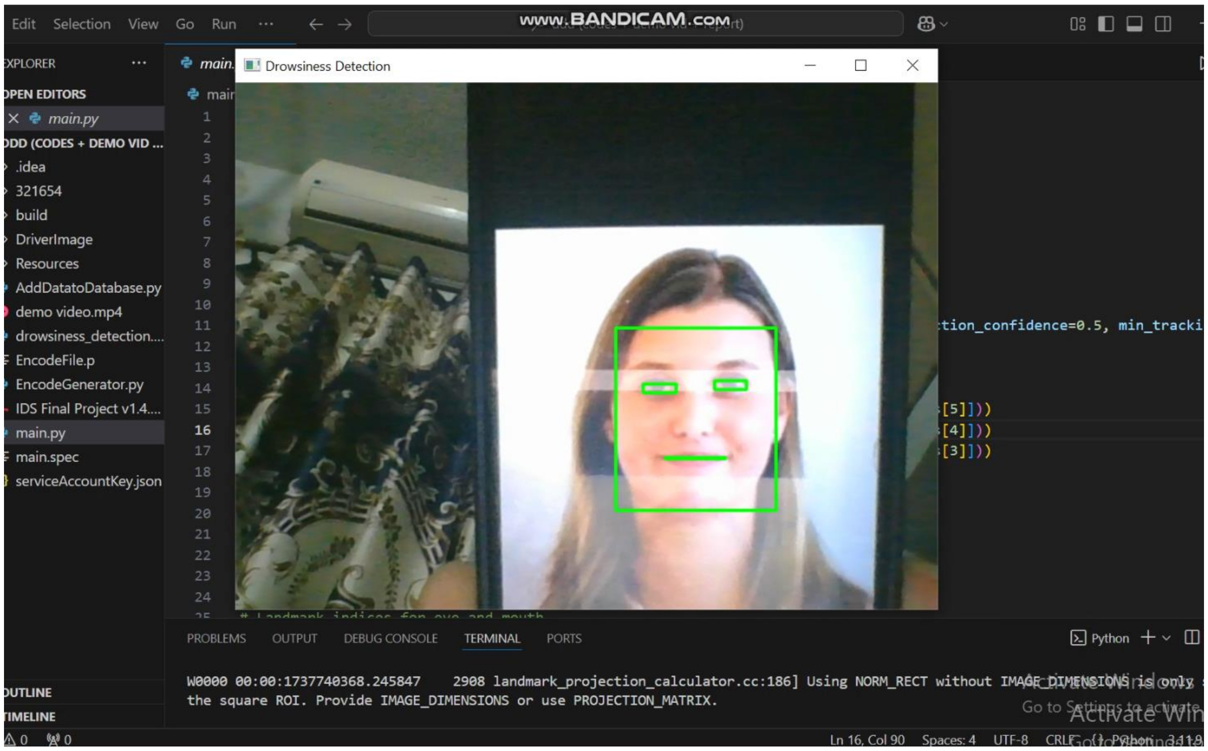
After combining all these sub portions of the code the final code and its out put looks something like this:

Code:

```python
import cv2
import mediapipe as mp
import numpy as np
import pyttsx3

# Initialize text-to-speech engine
engine = pyttsx3.init()

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

# Function to calculate aspect ratio
def aspect_ratio(landmarks, indices):
    A = np.linalg.norm(np.array(landmarks[indices[1]]) -
np.array(landmarks[indices[5]]))
    B = np.linalg.norm(np.array(landmarks[indices[2]]) -
np.array(landmarks[indices[4]]))
```

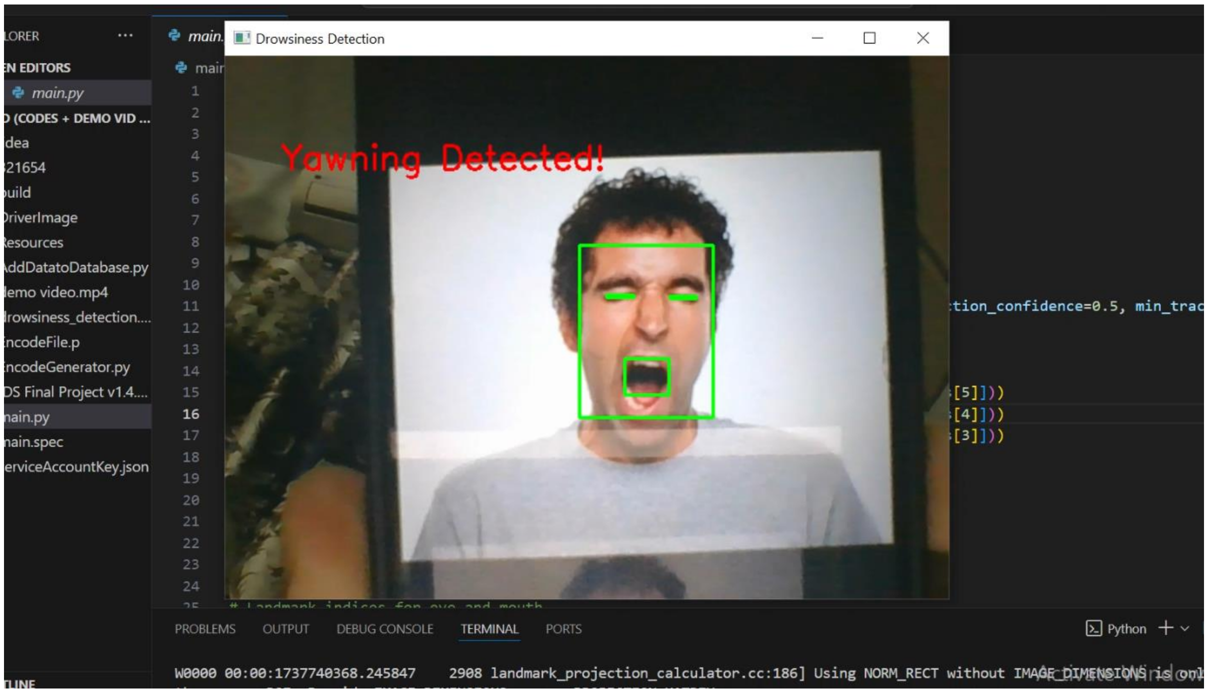
```python
    C = np.linalg.norm(np.array(landmarks[indices[0]]) -
np.array(landmarks[indices[3]]))
    return (A + B) / (2.0 * C)

# Drowsiness thresholds
EYE_AR_THRESHOLD = 0.2
EYE_AR_CONSEC_FRAMES = 48
YAWN_AR_THRESHOLD = 0.7

# Landmark indices for eye and mouth
LEFT_EYE = [33, 160, 158, 133, 153, 144]
RIGHT_EYE = [362, 385, 387, 263, 373, 380]
MOUTH = [78, 308, 13, 14, 87, 317]

# Variables for counting frames
eye_close_count = 0

# Initialize webcam
cap = cv2.VideoCapture(0)

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

    # Flip frame for a mirror-like effect
    frame = cv2.flip(frame, 1)
    h, w, _ = frame.shape

    # Convert frame to RGB
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # Process frame with MediaPipe Face Mesh
    results = face_mesh.process(rgb_frame)

    if results.multi_face_landmarks:
        landmarks = results.multi_face_landmarks[0].landmark
        normalized_landmarks = [(int(lm.x * w), int(lm.y * h)) for lm in
landmarks]

        # Calculate bounding box
        x_coords = [lm.x * w for lm in landmarks]
        y_coords = [lm.y * h for lm in landmarks]
        x_min, x_max = int(min(x_coords)), int(max(x_coords))
        y_min, y_max = int(min(y_coords)), int(max(y_coords))

        # Draw rectangle around the face
        cv2.rectangle(frame, (x_min, y_min), (x_max, y_max), (0, 255, 0), 2)

        left_eye_coords = [normalized_landmarks[i] for i in LEFT_EYE]
        right_eye_coords = [normalized_landmarks[i] for i in RIGHT_EYE]

        # For left eye
        left_eye_x_min = min([coord[0] for coord in left_eye_coords])
        left_eye_x_max = max([coord[0] for coord in left_eye_coords])
        left_eye_y_min = min([coord[1] for coord in left_eye_coords])
        left_eye_y_max = max([coord[1] for coord in left_eye_coords])
        cv2.rectangle(frame, (left_eye_x_min, left_eye_y_min),
(left_eye_x_max, left_eye_y_max), (0, 255, 0), 2)

        # For right eye
        right_eye_x_min = min([coord[0] for coord in right_eye_coords])
        right_eye_x_max = max([coord[0] for coord in right_eye_coords])
        right_eye_y_min = min([coord[1] for coord in right_eye_coords])
        right_eye_y_max = max([coord[1] for coord in right_eye_coords])
        cv2.rectangle(frame, (right_eye_x_min, right_eye_y_min),
(right_eye_x_max, right_eye_y_max), (0, 255, 0), 2)

        # Calculate and draw rectangle for the mouth
        mouth_coords = [normalized_landmarks[i] for i in MOUTH]
        mouth_x_min = min([coord[0] for coord in mouth_coords])
        mouth_x_max = max([coord[0] for coord in mouth_coords])
        mouth_y_min = min([coord[1] for coord in mouth_coords])
        mouth_y_max = max([coord[1] for coord in mouth_coords])
        cv2.rectangle(frame, (mouth_x_min, mouth_y_min), (mouth_x_max,
mouth_y_max), (0, 255, 0), 2)


        # Calculate eye aspect ratios
        left_eye_ar = aspect_ratio(normalized_landmarks, LEFT_EYE)
        right_eye_ar = aspect_ratio(normalized_landmarks, RIGHT_EYE)

        # Check if eyes are closed
        if left_eye_ar < EYE_AR_THRESHOLD and right_eye_ar <
EYE_AR_THRESHOLD:
            eye_close_count += 1
            if eye_close_count >= EYE_AR_CONSEC_FRAMES:
                cv2.putText(frame, "Drowsiness Alert!", (50, 50),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
                engine.say("You are feeling drowsy. Please take a break.")
                engine.runAndWait()
                eye_close_count = 0
        else:
            eye_close_count = 0

        # Calculate mouth aspect ratio
        mouth_ar = aspect_ratio(normalized_landmarks, MOUTH)
        if mouth_ar > YAWN_AR_THRESHOLD:
            cv2.putText(frame, "Yawning Detected!", (50, 100),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
            engine.say("Yawning detected. Please stay alert.")
            engine.runAndWait()

    # Display the frame
    cv2.imshow("Drowsiness Detection", frame)

    # Break the loop on 'q' key press
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release resources
cap.release()
cv2.destroyAllWindows()
```

Output:



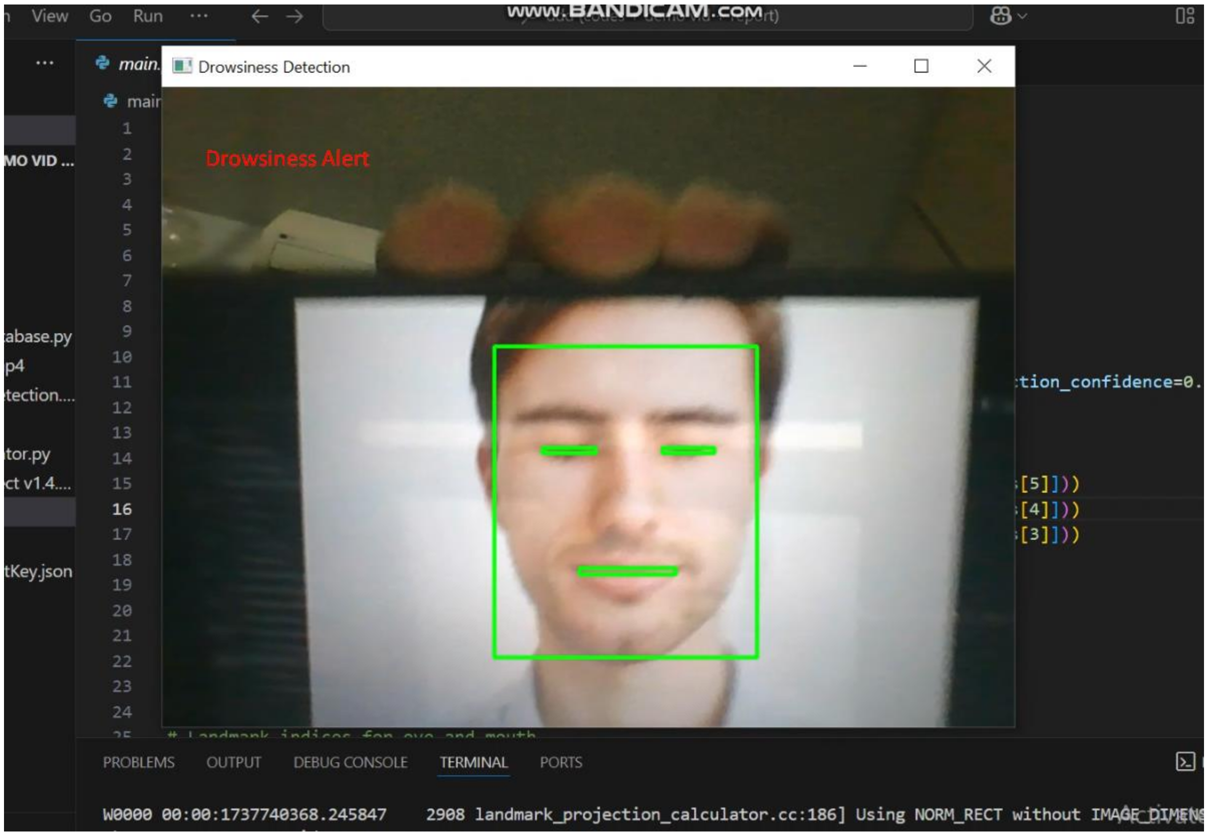
```
tion_confidence=0.5, min_tracki
[5]]))
[4]]))
[3]]))
# landmark indices for eye and mouth
```

```
W0000 00:00:1737740368.245847    2908 landmark_projection_calculator.cc:186] Using NORM_RECT without IMAGE_DIMENSIONS is only
the square ROI. Provide IMAGE_DIMENSIONS or use PROJECTION_MATRIX.
```



```
tion_confidence=0.5, min_trac
[5]]))
[4]]))
[3]]))
# landmark indices for eye and mouth
```

```
W0000 00:00:1737740368.245847    2908 landmark_projection_calculator.cc:186] Using NORM_RECT without IMAGE_DIMENSIONS is onl
```



```
tion_confidence=0.
[5]]))
[4]]))
[3]]))
# Landmark indices for eye and mouth
```

```
W0000 00:00:1737740368.245847    2908 landmark_projection_calculator.cc:186] Using NORM_RECT without IMAGE_DIMENS
```

Conclusion

This project provides a real-time solution for detecting drowsiness and yawning using facial landmark detection. By leveraging the power of MediaPipe, OpenCV, and pyttsx3, the system delivers both visual and auditory feedback, making it an effective tool for promoting safety, particularly in scenarios like driving or working long hours.

In future iterations, the system can be improved by integrating more robust machine learning models for facial recognition and combining it with other sensors to enhance drowsiness detection accuracy.

References

All the documents, papers, articles and WebPages that you have taken help from must be cited in the references section

Book
Author(s), Book Title. Place of publication: Publisher, year, volume, page number(s).

Example: [1] W.K. Chen, Linear Networks and Systems. Belmont, CA: Wadsworth, 1993, pp. 123-35.

Webpage
Author(s) and/or organization, date of publish or date the page was last updated, title of web page document, website address that provides a direct link to the document, and the date you last accessed the document

Example: Winston, J 1999, A look at referencing, AAA Educational Services, accessed 20 October 2015, <http://www.aaa.edu.au/aaa.html>. United Nations Web Services 2006,

Research Paper
Author(s), "Article title, Journal Title, vol., no., page number(s), Month year.

Example: [2] G. Pevere, "Infrared Nation, International Journal of Infrared Design, vol. 33, pp. 56-99, Jan. 1979.

If you need to reference any item that is not the the list, you should consult IEEE citation format available at the following link http://library.queensu.ca/book/export/html/5846

Plagiarism Report

Attached plagiarism report here that can be obtained from the library.

Report Approval Certificate

The report of the project, “Title” has been approved based on the following evaluation guideline.

Project Evaluation Guidelines

Artifacts Guidelines	
Analysis and Design artifacts are syntactically correct (use-case model, SSDs, domain model, class diagram, SDs, ERDs, Flow charts, Activity Diagram, DFDs)	
Consistency and traceability have been maintained among different artifacts	
General Guidelines	
Formatting (font style, indentation) is according to the FYP template and consistent throughout the document	
Captions are added to all the figures and tables. Figure captions must be placed below each figure, and table captions must be provided above the table	
Each figure or table is followed by some text describing what it represents	

Name & Signature
(Examiner 1)

Name & Signature
(Examiner 2)

Name & Signature
(Examiner3)

Name & Signature
(Supervisor)

Appendix

User Manual of the software