CAR MONITORING SYSTEM

A Project Report Submitted in the
Partial Fulfillment of the Requirements
for the Award of the Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

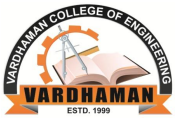
Submitted by

Jaina Sunhith	**19881A05E1**
Komire Aishwarya	**19881A05E8**
Dedavath Ganesh	**20885A0515**

SUPERVISOR
Dr. Raman Dugyala
Professor, CSE

Department of Computer Science and Engineering
VARDHAMAN COLLEGE OF ENGINEERING, HYDERABAD
An Autonomous Institute, Affiliated to JNTUH

May, 2022



VARDHAMAN COLLEGE OF ENGINEERING, HYDERABAD

An Autonomous Institute, Affiliated to JNTUH

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project titled **CAR MONITORING SYSTEM** is carried out by

Jaina Sunhith	**19881A05E1**
Komire Aishwarya	**19881A05E8**
Dedavath Ganesh	**20885A0515**

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** during the year 2021-22.

Signature of the Supervisor
Dr. Raman Dugyala
Professor, CSE

Signature of the HOD
Dr. Ramesh Karnati
Professor and Head, CSE

Kacharam (V), Shamshabad (M), Ranga Reddy (Dist.)–501218, Hyderabad, T.S.
Ph: 08413-253335, 253201, Fax: 08413-253482, www.vardhaman.org

Acknowledgement

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

We wish to express our deep sense of gratitude to **Dr. Raman Dugyala**, Professor, CSE and Project Supervisor, Department of Computer Science and Engineering, Vardhaman College of Engineering, for his able guidance and useful suggestions, which helped us in completing the project in time.

We are particularly thankful to **Dr. Ramesh Karnati**, the Head of the Department, Department of Computer Science and Engineering, his guidance, intense support and encouragement, which helped us to mould our project into a successful one.

We show gratitude to our honorable Principal **Dr. J.V.R. Ravindra**, for providing all facilities and support.

We avail this opportunity to express our deep sense of gratitude and heartful thanks to **Dr. Teegala Vijender Reddy**, Chairman and **Sri Teegala Upender Reddy**, Secretary of VCE, for providing a congenial atmosphere to complete this project successfully.

We also thank all the staff members of Computer Science and Engineering department for their valuable support and generous advice. Finally thanks to all our friends and family members for their continuous support and enthusiastic help.

Jaina Sunhith

Komire Aishwarya

Dedavath Ganesh

Abstract

The natural eye is an exceptionally quick processor of any picture, in contrast to some other machine across the globe. Be that as it may, sometimes, the rear of the vehicle and the edges of the vehicle are not apparent to the driver while driving and leaving. In this way, there is a requirement for such a fake natural eye to recognize the items at the edges. And also, now-a-days everyone feel drowsy due to the work stress or some other busy schedules and parallelly they have to drive home safe. So, every person need a system which addresses the above two issues. A calculation is planned which is utilized to identify the edges of the vehicle and the driver can undoubtedly see that view. Thus, first and foremost the driver can have the option to leave the vehicle and it is for the most part utilized in exceptionally populated regions like shopping centers, cinemas, and so forth with next to no conflicts with different vehicles.Also, it is useful to the drivers to see the edges while driving, which lessens unforeseen mishaps.

Keywords: rear;edges;drowsy;

Table of Contents

List of Figures

Abbreviations

Abbreviation	Description
UML	Unified Modeling Language
DD	Drowsiness Detection
SDLC	Software Development Life Cycle
SRS	Software Requirement Specification
IDE	Integrated Development Environment
RAM	Random Access Memory
ANN	Artificial Neural Network

CHAPTER 1

Introduction

1.1 Problem Definition

In-vehicle security innovation, driver tiredness location is extremely fundamental to forestall street mishaps. As indicated by the National Highway Traffic Safety Administration, consistently around 1,00,000 police-detailed crashes include sleepy driving. Sleepless drivers stay answerable for around 40% of street mishaps, as per authorization officials watching the thruways and significant streets here.

As many research projects and scientists proved that, human eye is extremely natural processor of any image, unlike any other machine or technology across the globe. In any case, on occasion, the back of the vehicle and the edges of the vehicle are not recognizable to the driver while driving and leaving. So, there is a necessity for such a phony regular eye to distinguish the articles at the edges. An estimation is arranged which is used to perceive the edges of the vehicle and the driver can see that view clearly.

1.2 Purpose

The number of road accidents are increasing enormously.In the past year there are almost 1,00,000 accidents are reported.In that around 40%-50% are due to tiredness and sleepiness of the driver.All the drivers doesn't have enough height so that when they sit in the driver seat all the corners of the car will be visible.Prominently when they want to drive car in the reverse the edges might not be clear and even in the auto driving it will be very help full for the system to detect the edges of the objects to maintain the appropriate distance.

1.3 Project Scope

Edge Detection can be achieved by the usage of the Canny Edge Detection algorithm. This algorithm is used to detect the edges of a car which makes driving and parking safe. It exclusively consists of 5 steps, in which the complete process of edge detection can be achieved.
Drowsiness Detection can be achieved by following a sequence of steps which start from image segmentation to moment of the points across the specified face parts.

CHAPTER 2

Literature Survey

2.1 Drowsiness Detection

W.L.Ou,M.H.Shih,C.W.Chang,X.H.Yu,C.P.Fan,"Intelligent Video-Based Drowsy Driver Detection System under Various Illuminations and Embedded Software Implementation", 2015 international Conf. on Consumer Electronics - Taiwan, 2015.

In this study an intelligent video-based driver drowsiness detection system which is not affected many factors is developed.Even though the driver wears glasses this system detects the drowsiness.

2.2 Edge Detection

Since the development of image processing there are many edge detection algorithms became available.The most common thing in all these algorithms is there are three main steps in each of them.

They are:

1) Smoothing
2) Differentiation
3) Labeling

But, Canny Edge Detection algorithm is composed of 5 steps which helps in detection of the edges in an efficient manner:

1) Noise reduction;
2) Gradient calculation;
3) Non-maximum suppression;
4) Double threshold;
5) Edge Tracking by Hysteresis;

Canny edge detector uses a multi-stage algorithm which can be used to detect large number of images.It was developed by John F.canny in 1986.

CHAPTER 3

Design

3.1 Software Development Life Cycle

Software Development Life Cycle (SDLC) is an interaction utilized by the product business to configuration, create and test top notch programming projects. The SDLC means to create a great programming that meets or surpasses client assumptions, arrives at consummation inside times and quotes.

- SDLC is a structure characterizing assignments performed at each progression in the product advancement process.
- ISO/IEC 12207 is a global norm for programming life-cycle processes. It means to be the standard that characterizes every one of the undertakings expected for creating and keeping up with programming.
- The existence cycle characterizes a procedure for working on the nature of programming and the general improvement process.

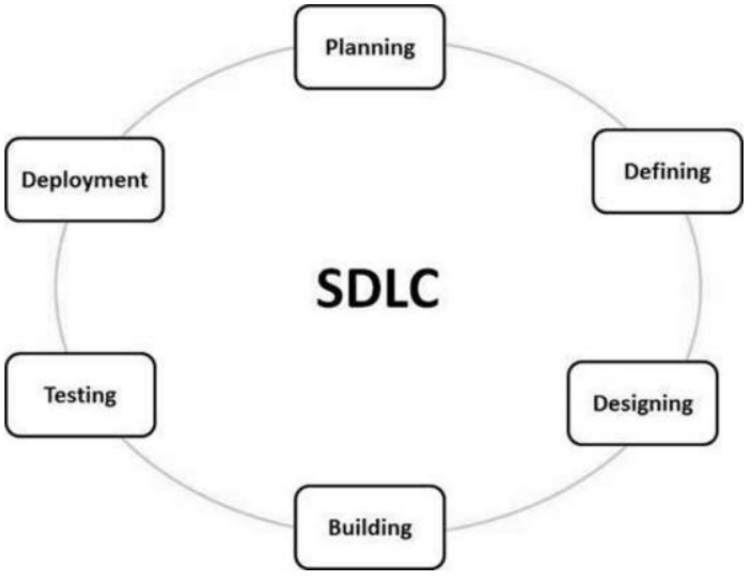


Figure 3.1: Software Development Life Cycle

3.1.1 Process Model

Waterfall Model Description: The Waterfall model is a basic linear sequential flow model. The progress is measured in form of phases of software development.Every phase of the development carries its importance.The waterfall approach is the oldest approach and the process can't go back to the previous phase anytime in the development process.

Advantages:

- Easy to implement.
- Helps to have a clear idea and plan of the project.
- Each phase has specific results.

Disadvantages:

- Requirements need to be handled precisely.
- Very difficult to go back to any stage after it is finished.

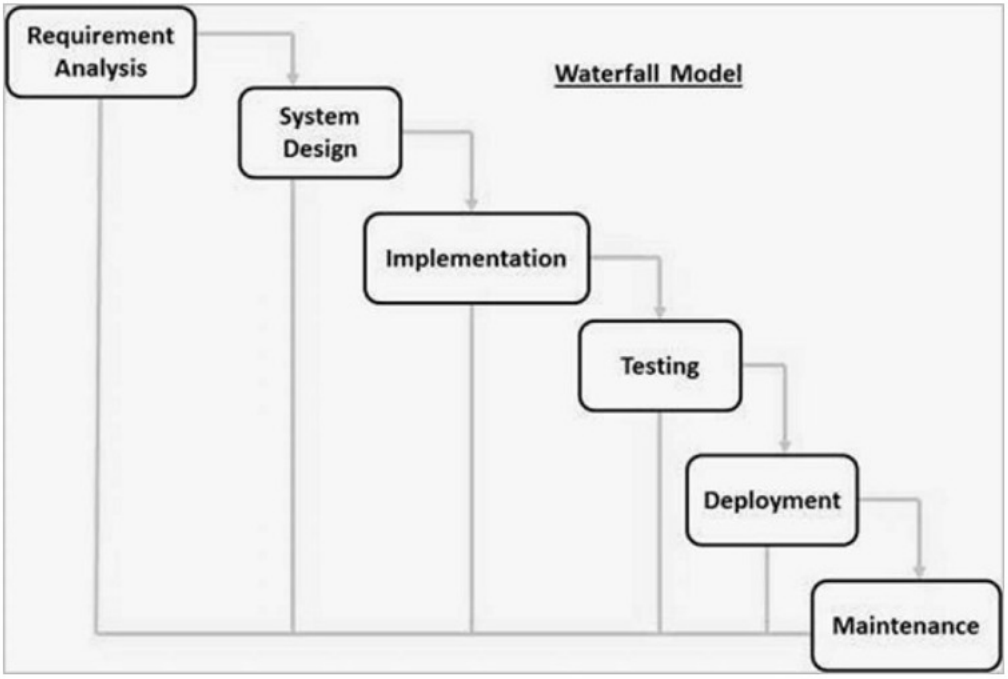


Figure 3.2: Waterfall Model

3.2 Software Requirement Specification

Functional requirements define the calculations, operations, and functionalities that a system must do.Behavioral requirements describe all the cases where functional requirements are used.This should be contrasted with non-functional requirements which specify overall characteristics such as cost and reliability.

3.2.1 Role of SRS

The purpose of Software Requirements Specification is to a better understanding of requirements.It gives a clear picture of the requirements of the proposed system.SRS is a medium through which the client and user needs are accurately specified.

3.2.2 Functional Requirements Specifications

The functions that define the system are called Functional Requirements. Functional Requirements of the system for those requirements are expressed in the natural language style.

Functional Requirements are:

- Drowsiness should be detected.
- Persons drowsiness is detected when the driver closes eyes or yawn.
- Edges should be detected.

3.3 Non-Functional Requirements:

Usability:

This section includes all the requirements that effect usability.The response time of the system depends on the latency of the user device.The Software system must have a user-friendly interface so customers can save time and

confusion.

Reliability:

The system is fault-tolerant within a specific range of capabilities of several functionalities.The react-Native platform provides a good range of functions to handle exceptions.Good documentation of react-native gives good reliability

Supportability:

The proposed system is designed to be cross-platform supportable.The system is supported on a wide range of hardware and any software platform of both android and iOS.

3.4 Software Requirements

- windowsOS/macOS/linuxOS.
- 64 bit processor.
- Python IDE.
- TensorFlow.
- OpenCV.
- Alarm (To alert the user).

3.5 Hardware Requirements

- Laptop/Desktop (for implementation)
- Alarm(Inbuilt or Explicit alarm)
- 6GB RAM or more
- Webcam

3.6 Languages Used

- Python

Python is a trending programming language with optimized lines of code. It is a high-level and an interpreted language. It has many inbuilt libraries which support the implementation of most complicated tasks like machine learning, deep learning and even ANN models. It is also used to automate tasks, build websites and software, and conduct data analysis. Object-oriented, functional programming and many more programming paradigms are supported by python.

3.7 Libraries Used

- face_utils

face_utils provides a set of face utilities including face cropping. It is an open-source wrapper library for face detection approaches or algorithms and models. Some of the examples where face_utils can be used are face recognition, retina face model, haar cascade face detection, etc.

- OpenCV

OpenCV is a python library used for image processing and to solve computer vision tasks. Though, C/C++ is the best language to use OpenCV, we can also use python or java API. This image processing is done with the methods and modules of OpenCV.

- NumPy

NumPy is a python library used to perform high-level mathematical function like random number generations, linear algebra routines, Fourier transforms, etc. It is used for multi-dimensional arrays and matrix processing, scientific

computations in Machine Learning.

- time

time is a python module which handles all time-related tasks. It consists many built-in functions, in which, every function performs certain particular task. One should import time module to access all those time-related in-built functions. Usage of this module any representation of time in various datatypes like int, string, objects, etc.

- datetime

datetime is a python module which handles the manipulations of date and time. To access all the built-in datetime functions, we need to import datetime module. It also has many in-built classes which can be overridden according to our use.

- dlib

dlib is a toolkit comprising of machine learning algorithms and also provides all the requirements to create a real-world software. It is also used in data analysis applications. It is usually written in C++ programming language. However, it is also used with python bindings.

- imutils

imutils is a package which is based on OpenCV and it provides an easy interface for the functioning of OpenCV. It contains a progression of accommodation capacities to make fundamental picture handling capacities like interpretation, turn, resizing, skeletonization.

- winsound

winsound is a python module which is used to play sounds. It provides admittance to sound-playing apparatus. It comprises of various constants and functions which have their own specifications and functionalities.

- argparse

argparse is parser which is used to pass the arguments in the functions. The argparse module makes it simple to compose easy to understand order line or command-line interfaces.

3.8 Flow Diagram

The flow chart in the figure 3.1 clearly explains the work flow of the drowsiness detection and the edge detection.

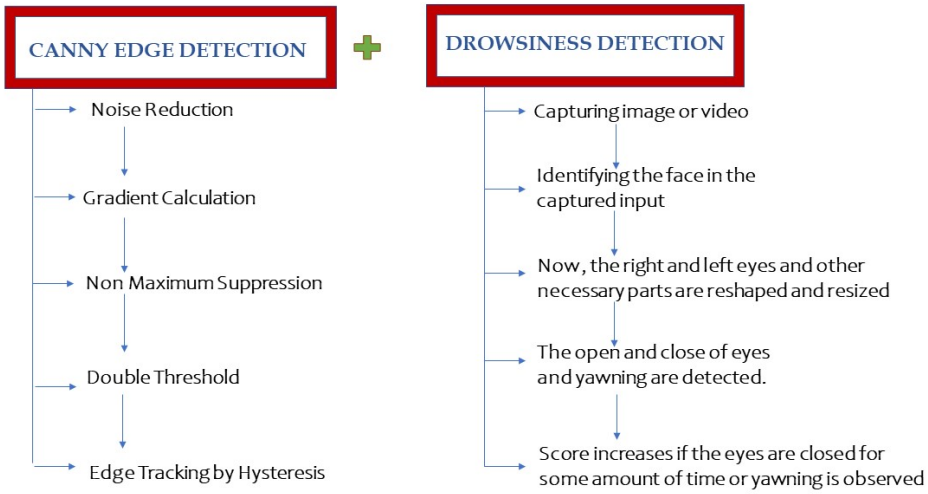


Figure 3.3: Edge Detection+Drowsiness Detection

CHAPTER 4

Algorithms or Approaches

4.1 Canny Edge Detection Algorithm

Canny Edge Detection Algorithm is an Edge Detection operator that utilizes a multi-stage calculation to distinguish a wide scope of edges in pictures. It is developed by John F Canny in 1986. It is based on gray scale pictures. Canny Edge Detection Algorithm follows 5 steps to identify the edges:

1. Noise Reduction: Gaussian Blur is applied to smoothen the image, such that, the unnecessary noise from the image get reduced, which makes the image ready to process.

2. Gradient Calculation: In this step, gradient image is calculated using edge detection operators which help in the detection of the edge intensity and direction.

3. Non-Maximum Suppression: The algorithm goes through all the points of the gradient intensity matrix. The pixels of the maximum value in the edge directions are found. Edges will be slimmed out.

4. Double Threshold: The previous step results in the image with the varied edge intensities which are named as strong, weak and non-relevant in this step and this identification is done based on the high and low thresholds. The outcome of this step contains only strong and weak pixels.

5. Edge Tracking by Hysteresis: In this step, weak pixels are transformed into strong pixels, if there exists at least one strong pixel around them to be processed.

Figure 4.1: Edge detection example

4.2 Drowsiness Detection Approach

The steps followed to detect the drowsiness are:

1. Image or video is captured through the web cam.

2. As soon as the image or a video frame is read, face is identified from the read input.

3. Image segmentation is done to mark the points on the face using dlib module, 68 points are plotted on the face for identification of the facial movements.

4. Now the eyes and other necessary parts(mouth and around) are resized and reshaped for further processing.

5. The opening and closing of eyes(to detect whether the subject is sleeping or not) and mouth(to detect whether the subject is yawning or not) are detected using the euclidean distance between the facial points.

6. Score will be increased if the eyes are closed or mouth is opened(yawning) for an amount of time which is greater than or equal to the given threshold values.

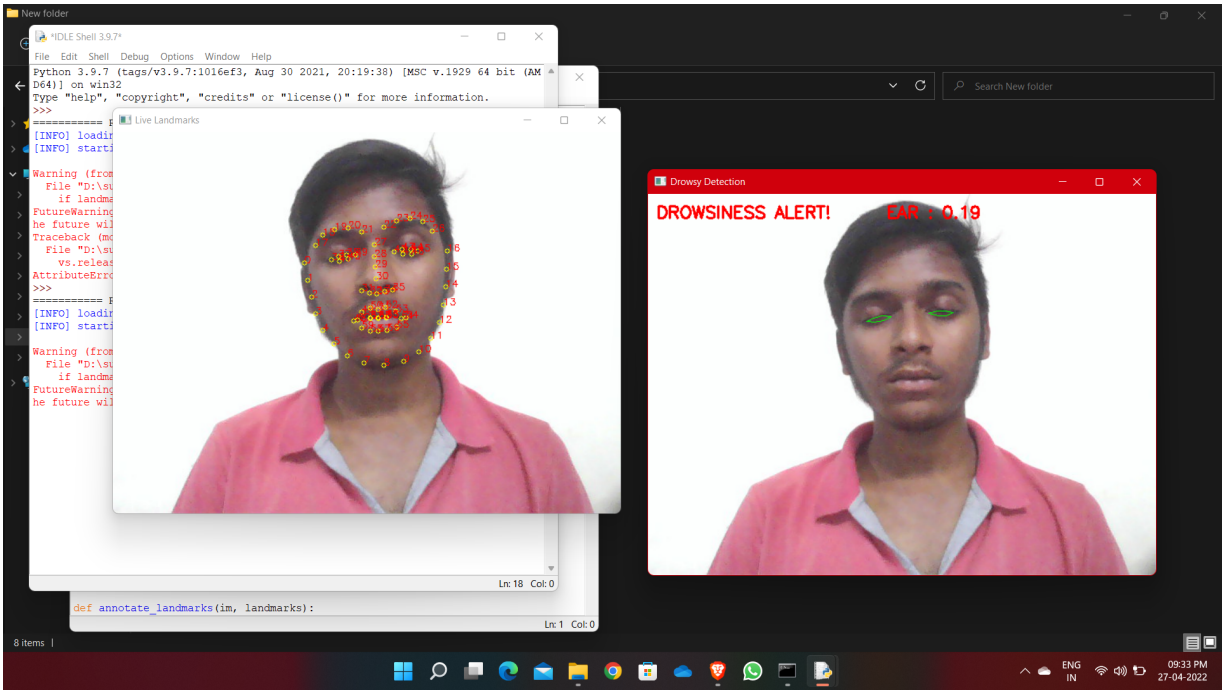


Figure 4.2: Drowsiness alert of the user

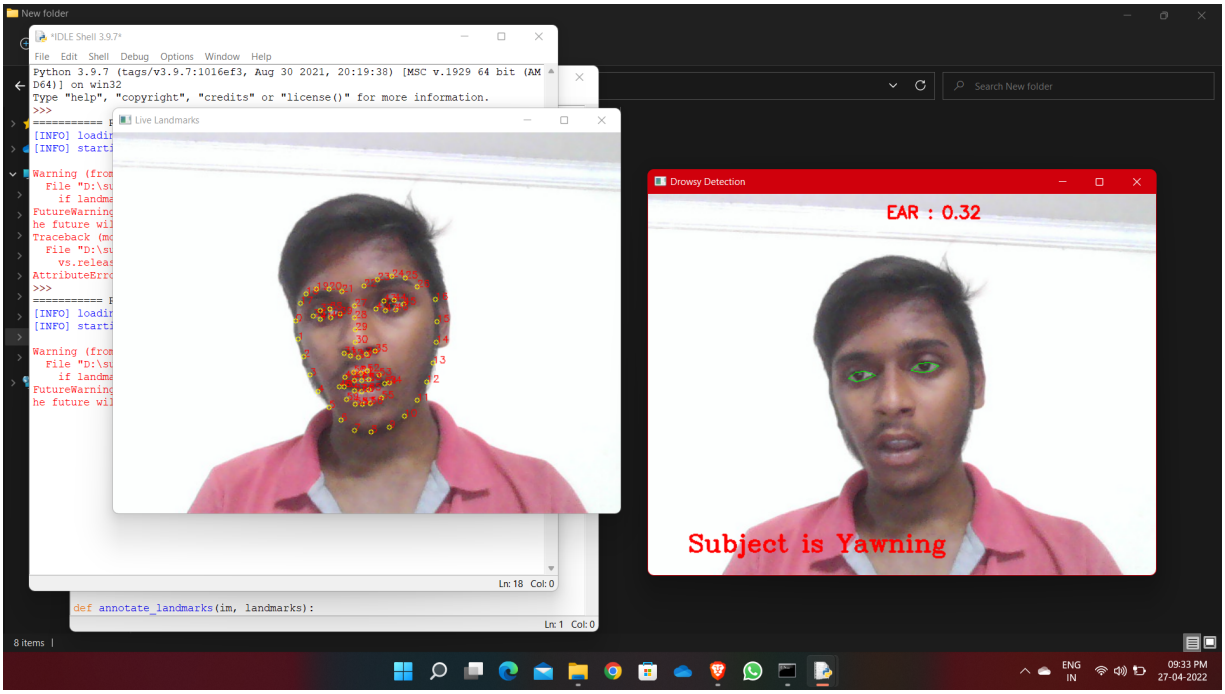


Figure 4.3: Yawning alert of the user

CHAPTER 5

UML diagrams

- UML stands for Unified Modeling Language.
- It is a visual representation of a complex software system in the form of design,architecture and implementation.
- It helps in the easy understanding of any system.
- It attracts the client and gives the clear picture of the system without the detailed documentation.

5.1 Activity diagram

Activity diagram pictures the behaviour of the system.It shows the various decision paths from starting to the ending that exists while the activity is executed.It actually shows the development in the actions of software and business processes.

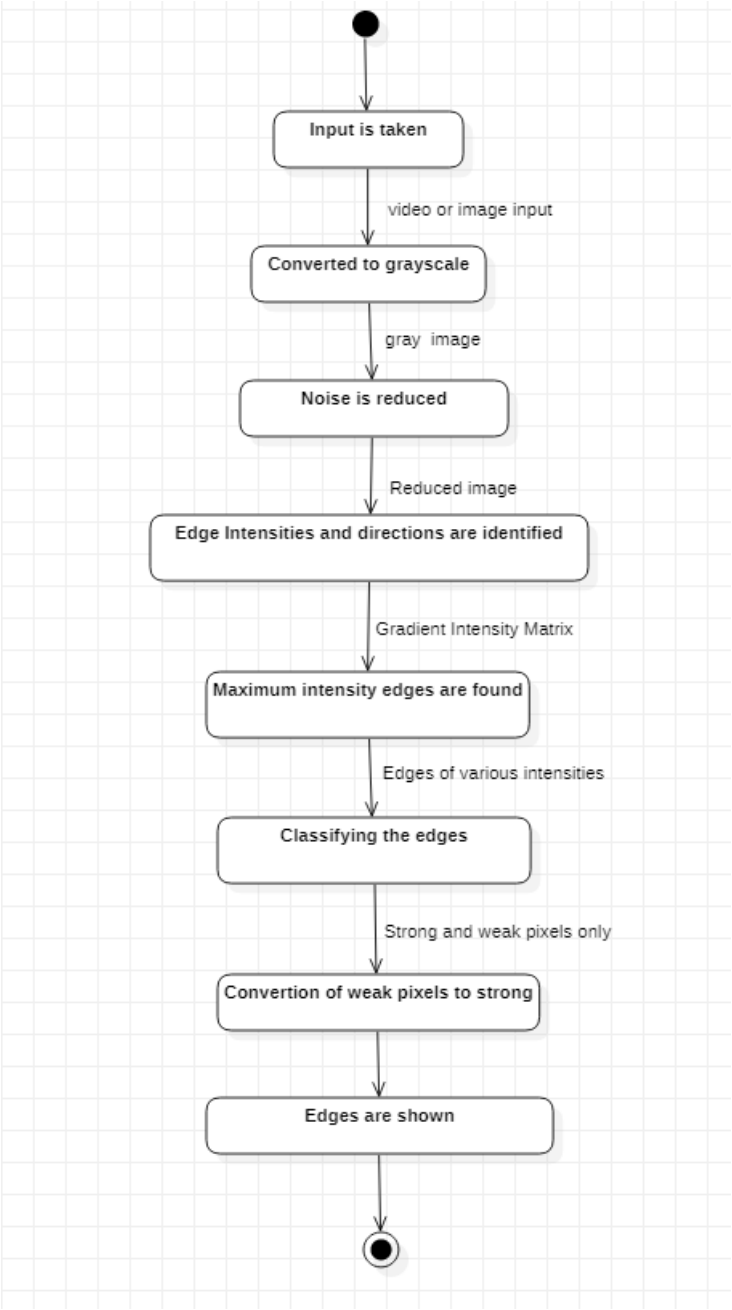


Figure 5.1: Edge detection

Description First The video input from the camera is taken.Since image segmentation can be done for only gray-scale images the input is converted into gray-scale later for the clear edges noise reduction is done then directions are identified.Now weaker pixels are converted into strong and the edges are shown.

Figure 5.2: Drowsiness detection

Description First The video input from the camera is taken.Since image segmentation can be done for only gray-scale images the input is converted into gray-scale.Eyes and mouth segmented,resized and reshaped.now the eyes and mouth will only be scanned if the eyes are closed and the mouth is opened the score will increase and the alarm sounds.

5.2 Use Case diagram

A use case diagram is a graphical depiction of a user's possible interactions with a system..It is used to describe the high level function of a system.The various users and the cases of the system are shown in the use case diagram.

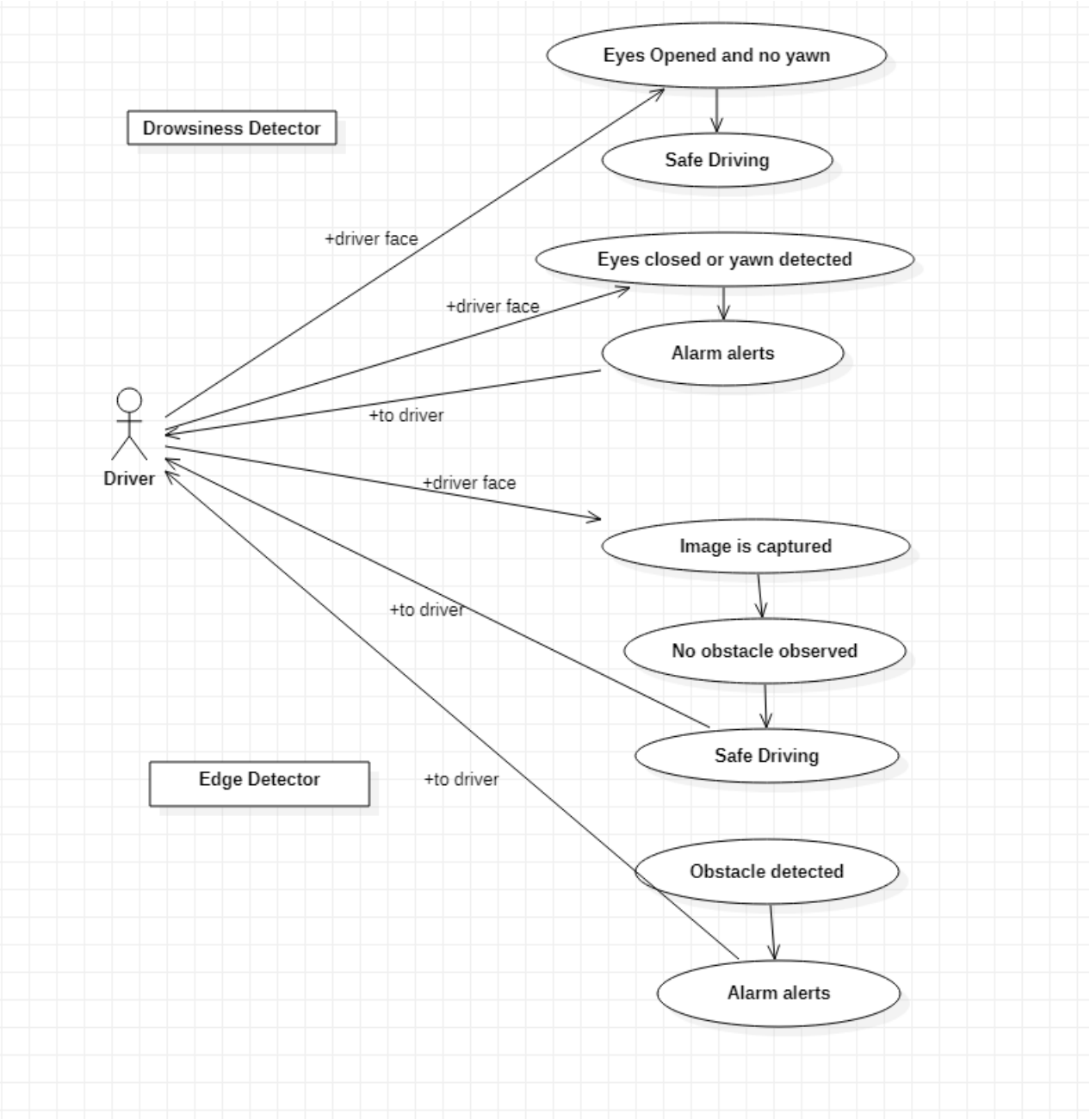


Figure 5.3: Use case

Description

Drowsiness detection: If the eyes of the driver are opened and there is no yawn then the driver is safe driving.If the eyes are closed or yawn is detected then the alarm turns on.

Edge detection: First the image is captured then if there is no edge of any object is detected then it is safe driving.If any edge of any obstacle is detected then the alarm turns on.

5.3 Deployment diagram

Deployment diagram shows the execution architecture of a system.It includes hardware and software execution environments and connections connection them i.e., it shows what hardware components are there in system and what software components are connected to which nodes.

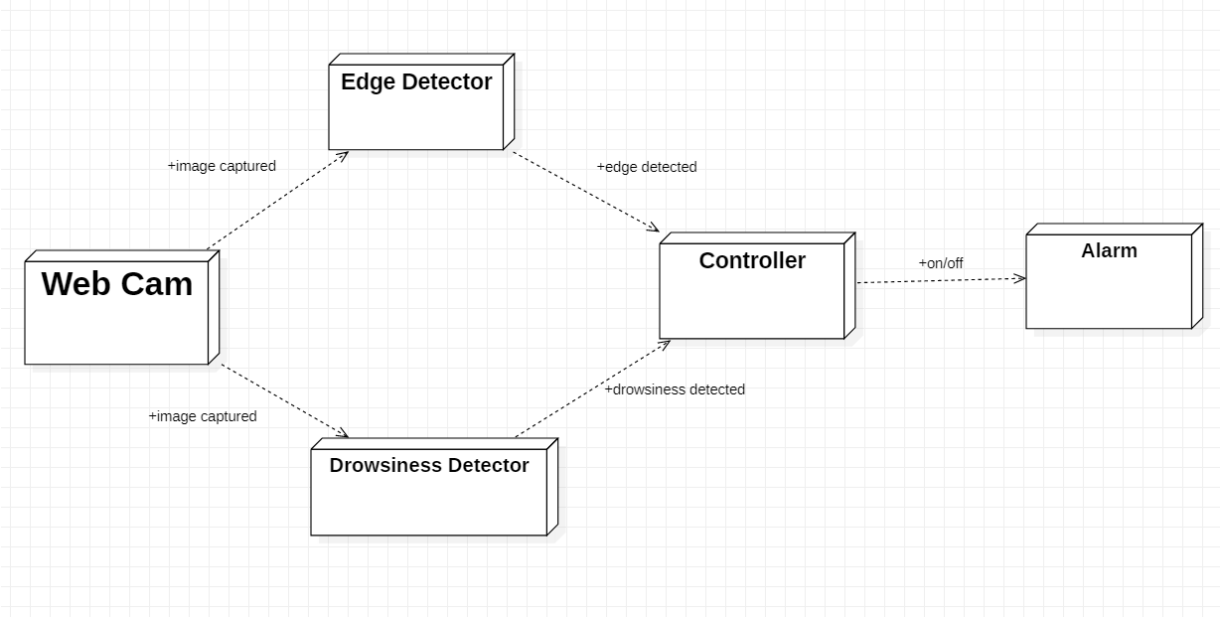


Figure 5.4: Deployment diagram

Description The webcam takes the video input and in case of edge detector the video will be sent to the edge detector algorithm .After the edges are detected the output will be sent to controller.Then the controller decide to turn the alarm on or off.In case of drowsiness detection the video input will be sent to drowsiness detection algorithm and then the input will be

processed.later the out put will be sent to controller,now the controller will decide to turn the alarm on or off.

5.4 Class Diagram

These are the blueprints of the system.Class diagrams are actually used to model the objects of the system.These can be used to display the relationships between the objects,and to describe the services of those objects.A class diagram describes the attributes and operations of a class and also the constrains imposed on the system.

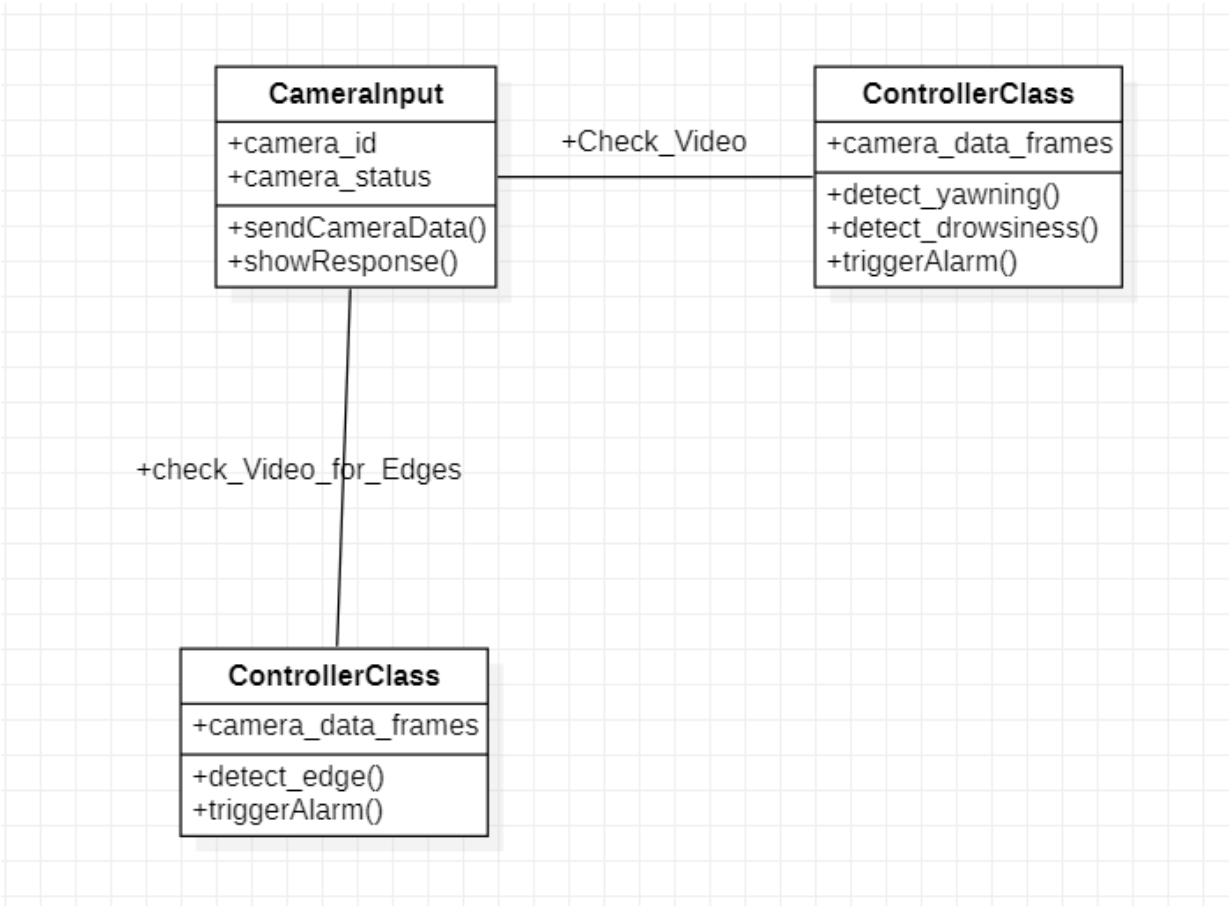


Figure 5.5: Class diagram

Description

Classes:

1.CameraInput class.

2.Controllerclass.

The input class has camera_id and camera_status attributes.This class send the camera data to controller class and receive responses and shows those responses.There are controller classes to check the drowsiness of the driver and the other controller checks for the edges of the objects.

CHAPTER 6

Code snippets and output images

6.1 Code snippets

6.1.1 Edge detection

```
def canny_webcam():
    cap = cv2.VideoCapture(0)

    while True:
        ret, frame = cap.read()
        frame = cv2.GaussianBlur(frame, (7, 7), 1.41)
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        edge = cv2.Canny(frame, 25, 75)

        cv2.imshow('Canny Edge', edge)

        if cv2.waitKey(1) & 0xFF == ord('q'):
```

Figure 6.1: Edge detection

Above is the code snippet of edge detection using open cv.

6.1.2 Drowsiness detection

```
print("[INFO] starting video stream thread....")
vs = VideoStream(src=0).start()
time.sleep(1.0)

frame = vs.read()
frame = imutils.resize(frame,width=450)
img = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
font = cv2.FONT_HERSHEY_SIMPLEX
input_frame = img

while True:
    frame = vs.read()
    image_landmarks, lip_distance = mouth_open(frame)

    img = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = detector(img, 0)

    prev_yawn_status = yawn_status

    if lip_distance > 25:
        yawn_status = True

        cv2.putText(frame, "Subject is Yawning", (50,450),
                    cv2.FONT_HERSHEY_COMPLEX, 1,(0,0,255),2)
        winsound.Beep(freq, duration)

    else:
        yawn_status = False

    if prev_yawn_status == True and yawn_status == False:
        yawns += 1


    for face in faces:

        face_data = face_utils.shape_to_np(predictor(img,face))
        left_eye = face_data[36:42]
        right_eye = face_data[42:48]
        leftEyeHull = cv2.convexHull(left_eye)
        rightEyeHull = cv2.convexHull(right_eye)

        cv2.drawContours(frame,[leftEyeHull], -1, (0, 255, 0), 1)
        cv2.drawContours(frame,[rightEyeHull], -1, (0, 255,0), 1)

        leftEAR = eye_ratio(left_eye)
        rightEAR = eye_ratio(right_eye)

        eye_avg_ratio = (leftEAR + rightEAR)/2.0

        if eye_avg_ratio < EYE_AR_THRESH:
            COUNTER = COUNTER + 1
```

Figure 6.2: Drowsiness detection

6.2 output screens

6.2.1 Edge detection

output images of edge detection before processing and after processing.

Figure 6.3: Image before processing

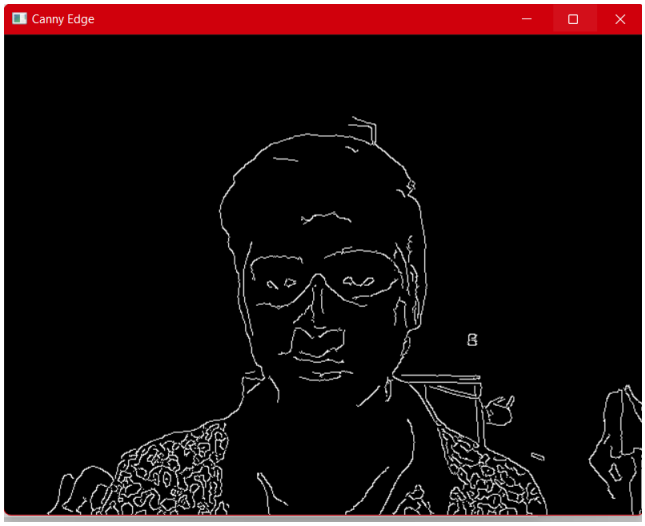


Figure 6.4: Image after processing

6.2.2 Drowsiness Detection

output screens of drowsiness detection while the driver is sleepy in case of closing eyes and yawning.

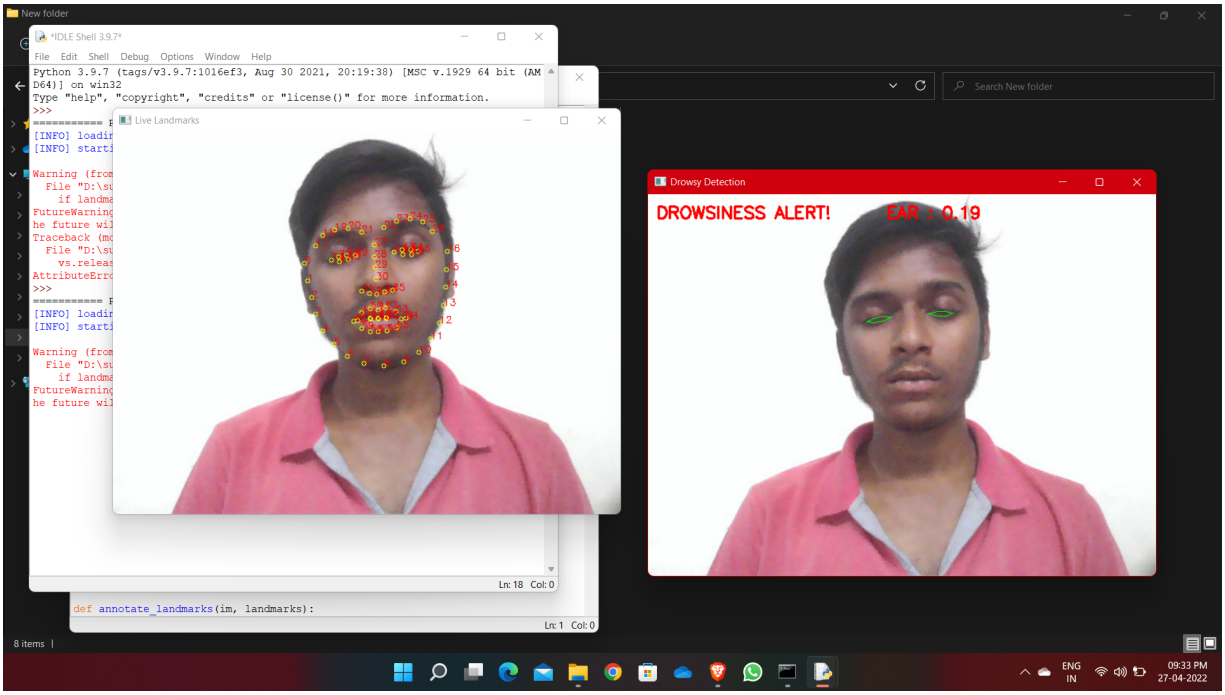


Figure 6.5: Drowsiness alert of the user

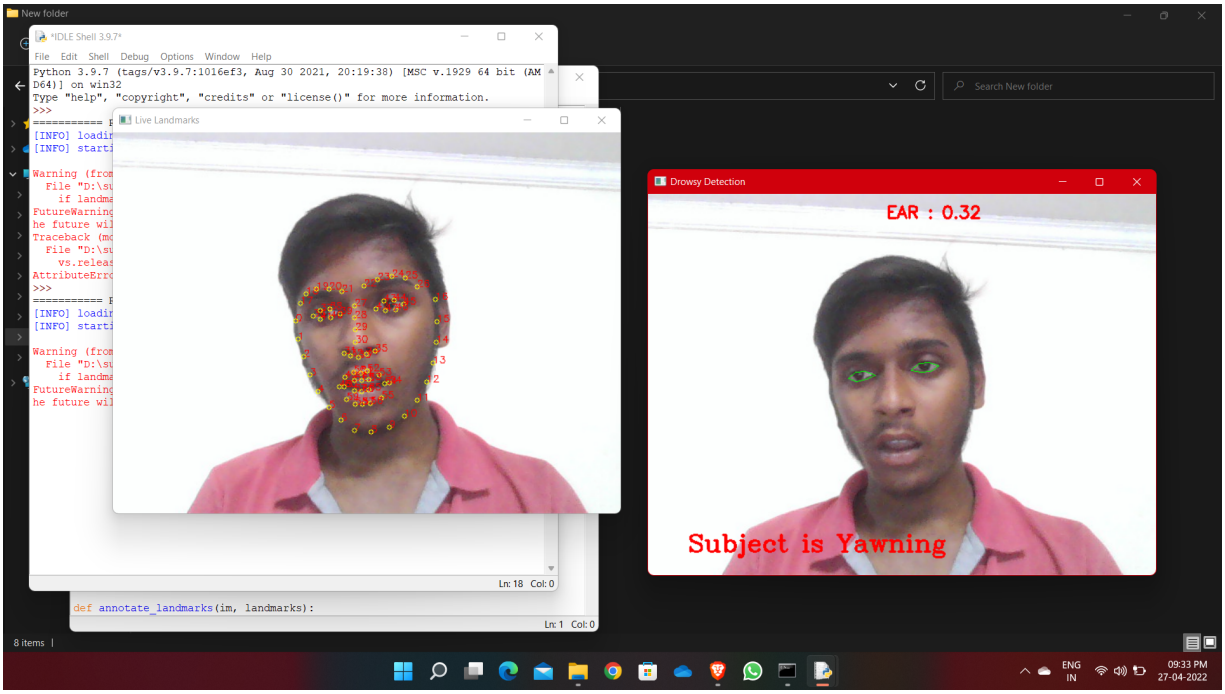


Figure 6.6: Yawning alert of the user

CHAPTER 7

Conclusions and Future Scope

7.1 Conclusions

Using this system we can detect the drowsiness of the driver even if he wear glasses.A low cost and real time drowsiness detection and edge detection system has introduced using visual behaviour and machine learning.Here visual behaviour is captured by a HD cam.

A threshold value is given and when the system crosses that threshold value drowsiness will be detected in real time.This can be used when a driver is involved and along with that the data can be collected and used for classification of the model.The edges can be detected using the canny edge detection algorithm.This can be used in place of reverse cam and the automated cars to reduce the accidents and to clearly detect the edges of the objects in front of the vehicle.

7.2 Future Scope

Both the algorithms can be used in one application created by using nodeJS. This edge detection and drowsiness detection can also deployed into automatic driving and also can be used separately based on the requirements. This can be further extended and integrated by adding tensor flow to operate through mobile phone.

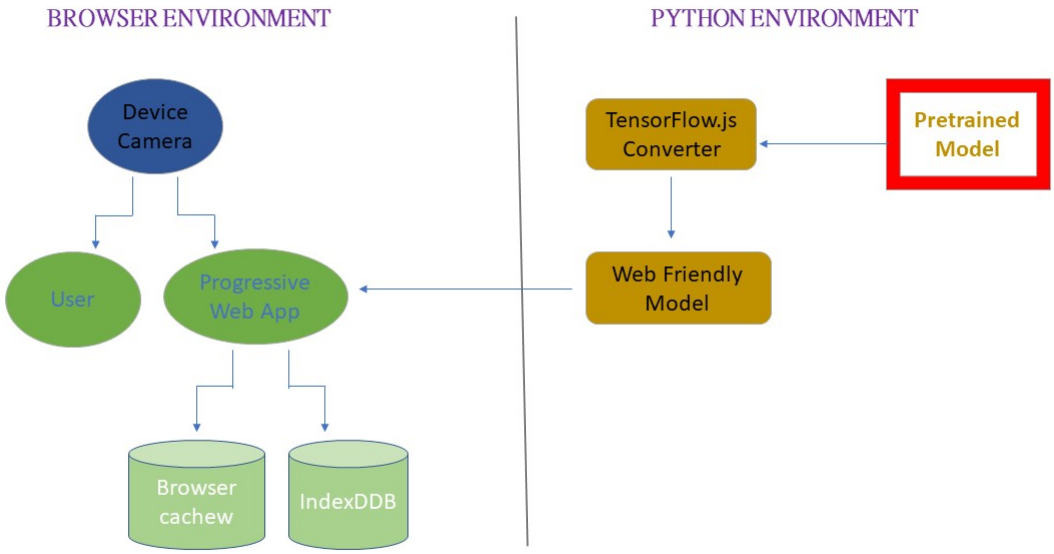


Figure 7.1: Application Infrastructure