Project Report

On

DROWSINESS DETECTION SYSTEM

Submitted by

ID: R170481

J SUCHITHRA

Under the guidance of

RATNA KUMARI CHALLA

M.Tech

Rajiv Gandhi University of Knowledge and Technologies(RGUKT),

R.K.Valley, Kadapa, Andhra Pradesh

Rajiv Gandhi University of Knowledge and Technologies(RGUKT),

R.K.Valley, Kadapa(Dist), Andhra Pradesh, 516330.

<u>CERTIFICATE</u>

This is to certify that the project work titled "**<u>DROWSINESS DETECTION SYSTEM</u>**" is a bonafide project work submitted by **J.SUCHITRA** in the department of COMPUTER SCIENCE AND ENGINEERING in partial fulfillment of requirements for the award of degree of Bachelor of Technology for the year 2022-2023 carried out the work under the supervision of Guide.

GUIDE	**HEAD OF THE DEPARTMENT**
RATNA KUMARI CHALLA	N SATYANANDHARAM

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts success.We are extremely grateful to our respected Director, **Prof. K. SANDHYA RANI** for fostering an excellent academic climate in our institution.

We also express my sincere gratitude to our respected Head of the Department **Mr. N SATYANANDHARAM** for his encouragement, overall guidance in viewing this project as a good asset and effort in bringing out this project. We would like to convey our thanks to our guide **Ms. RATNA KUMARI CHALLA** for her guidance, encouragement, cooperation and kindness during the entire duration of the course and academics. Our sincere thanks to all the members who helped us directly and indirectly in the completion of project work. We express our profound gratitude to all our friends and family members for their encouragement.

ABSTRACT

One of the leading causes and reasons of traffic accidents is Driver drowsiness. It’s a natural for drivers who travel long distances to fall asleep behind the wheel. Drivers might become tired while driving due to variety of factors, including stress and lack of sleep. By developing a drowsy detection agent in which this project study hopes to avoid and reduce such accidents.

We will use **Python, OpenCv, Scipy** and **Keras** to create a system that can detect drivers’ closed eyes are closed for a few seconds, this technology will alert the driver, preventing potentially fatal road accidents which can save his life and others too. We will use OpenCv to collect photos from a camera and feed them into a Deep Learning model that will classify whether the person’s eyes are **‘Opened’ or ‘Closed’** and whether the driver is **‘yawning’.**

TABLE OF CONTENTS

CHAPTER 1 : INTRODUCTION

1.1 Background

“1 in 25 adult drivers report that they have fallen asleep at the wheel in the past 30 days”. Additionally, we believe that drowsiness can negatively impact people in working and classroom environments as well. Although sleep deprivation and college go hand in hand, drowsiness in the workspace especially while working with heavy machinery may result in serious injuries similar to those that occur while driving drowsily. Our solution to this problem is to build a detection system that identifies key attributes of drowsiness and triggers an alert when someone is drowsy before it is too late.

1.2 Objectives

The main objective of this project on improving the safety of the driver and it can also be implemented in organization or work spaces to make sure the employees don’t fall asleep.

1.3 Purpose, Scope, and Applicability

1.3.1 Purpose

A countless number of people drive on the highway day and night. Taxi drivers, bus drivers, truck drivers and people traveling long-distance suffer from lack of sleep. Due to which it becomes very dangerous to drive when feeling sleepy. The majority of accidents happen due to the drowsiness of the driver.

1.3.2 Scope

The scope of this project is to enhance the safety of the driver as well as the passengers in the vehicle by alerting the driver when he is falling drowsy/sleepy.

The main objective of this application will include the following :

1) Facial Landmark detection

2) Eyes aspect ratio

3) Alerting the driver

1.3.3 Applicability

As the system works on real time data, the system does not store data permanently in any form of data storage. There is no need of internet and hence can be used anytime, anywhere by the user/driver.

1.4 Achievements

Undertaking the above project, personally has given me opportunity to learn completely new set of technologies and concepts which I was unaware before. This Project makes use of Python and opencv libraries which helps in making dynamic application. This project has introduced me to a lot of deep learning algorithms and the advanced concepts of computer vision.

CHAPTER 2: SURVEY OF TECHNOLOGIES

Python - Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach that aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural) object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Python interpreters are available for many operating systems.

Open CV - MySQL OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source BSD license. OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. All of the new developments and algorithms appear in the C++ interface. There are bindings in Python, Java and MATLAB/OCTAVE.

The API for these interfaces can be found in the online documentation. Wrappers in other languages such as C#, Perl, Ch, Haskell and Ruby have been developed to encourage adoption by a wider audience. Since version 3.4, OpenCV.js is a JavaScript binding for selected subset of OpenCV functions for the web platform. OpenCV runs on the following desktop operating systems: Windows, Linux, macOS, FreeBSD, NetBSD, OpenBSD. OpenCV runs on the following mobile operating systems: Android, iOS, Maemo, BlackBerry 10.The user can get official releases from source OpenCV uses CMake.

CHAPTER 3: REQUIREMENTS AND ANALYSIS

3.1 Problem Definition

A countless number of people drive on the highway day and night. Taxidrivers, bus drivers, truck drivers and people traveling long-distance suffer from lack of sleep. Due to which it becomes very dangerous to drive when feeling sleepy. The majority of accidents happen due to the drowsiness of the driver.

3.2 Requirements Specification

3.2.1 System Features

3.2.1.1 Facial Landmark detection:

Facial Landmark detection is the process of localizing key facial structures on a face, including the eyes, eyebrows, nose, mouth, and jawline, and from these features, in the context of drowsiness detection, we only extract the eye and mouth regions.

3.2.1.2 Eye Aspect Ratio:

After the eye regions have been extracted, we apply the eye aspect ratio to determine if the eyes are closed. If the eyes have been closed for a sufficiently long enough period of time, we can assume the user is at risk of falling asleep and sound an alarm to grab their attention.

3.2.1.3 Alerting system:

A system that can automatically detect driver drowsiness in a real-time video stream, using the above mentioned modules and then alert the driver with alarm or sound if the driver appears to be drowsy.

3.2.2 External Interface Requirements

User interface

The Since we have used python for developing the drowsiness detection system, Python provides various options for developing graphical user interfaces (GUIs) such as Tkinter. We have used tkinter for the user interface. Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. Tkinter is Standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

3.2.3 Performance Requirements

•The system only works well when the camera is able to get full view of the face of the driver.

•The system's initial load time depend on the device configuration on which it is running.

•The system must be interactive and the delays involved must be less.

•The overall performance of the system depends on the performance of the device.

3.2.4 Safety Requirements

As the system works on real time data, the system does not store data permanently in any form of data storage. Therefore there is minimum risk in terms of privacy of the driver or the user of the system.

3.2.5 Security Requirements

User Since the system works on real time drowsiness detection, the camera needs to be placed in full view of the driver's face in order to accurately detect the eyes and mouth regions.

3.2.6 Software Quality Attributes

I. Reliability

It's The solution should reliably detect drowsiness so that it can serve its purpose as a system for promoting driver safety.

II. Real time response

The operation of a vehicle can involve relatively high speeds, a system that cannot detect drowsiness and warn that driver promptly can lead to serious consequences.

3.3 Planning and Scheduling

Planning, Scheduling and completion of different milestones of the project is displayed in the below table: Table

Sl. No.	Milestone Name	Milestone Description	TimeLine (weeks)	Remarks
1	Requirement Specification	Completed specification	1	Attempt should be made to add some more relevant functionalities
2	Technology Familiarization	Understanding of the technology needed to implement the project.	2	The presentation should be from the point of view of being able to apply it to the project, rather than from a theoretical perspective.
3	High level & detailed design	Listdown all possible scenarios & then combining up with flowcharts	3	The scenarios should match to the requirement specification.
4	Implementation of the front end of the system	Implementation of the main screen, screen that follows sharing giving various options.	2	During this milestone period it would be a good idea for developer to start working on a test plan of entire system. This test plan can be updated as and when new scenarios come to mind.
5	Integration testing	The system should be thoroughly tested by running all the test cases written for the system.	2	Another one week should have been there to handle any issues found during testing of the system.
6	Final Review	Issues found during the previous milestone are fixed and the system is ready for the final review.	1	During the final review of the project, it should be checked that the entire requirement specified during the milestones are fulfilled.

3.4 Software and Hardware Requirements

Software Requirements:

Operating system	:	Ubuntu 18.04
Front end	:	Tkinter
Back end	:	Python

Hardware Requirements:

Processor	:	AMD® A9-9420 radeon r5, 5 compute cores 2c+3g × 2
RAM	:	4GB
Hard disk	:	487.0 GB

3.5 Preliminary Product Description

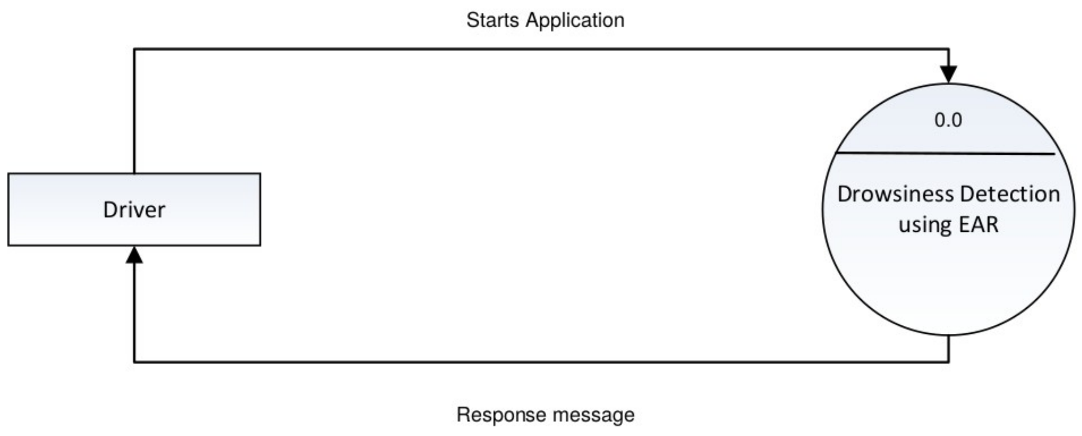
The main objective of this project/product is on improving the safety of the driver and it also can be implemented in organization or work spaces to make sure the employees don't fall asleep.

3.6 Conceptual Models

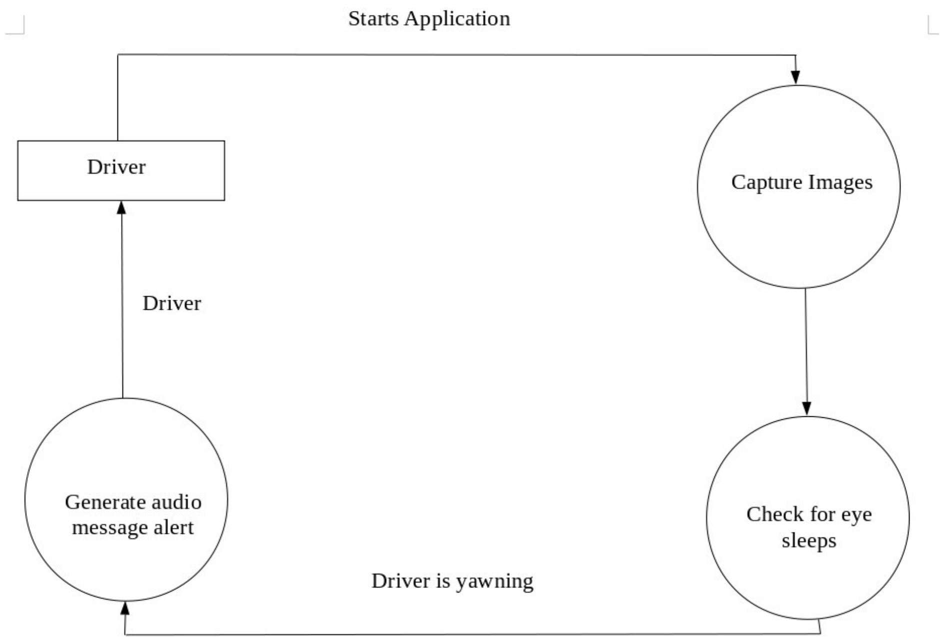
3.6.1 Data Flow Diagrams

Data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system; it differs from the flowchart as it shows the data flow instead of the control flow of the program.

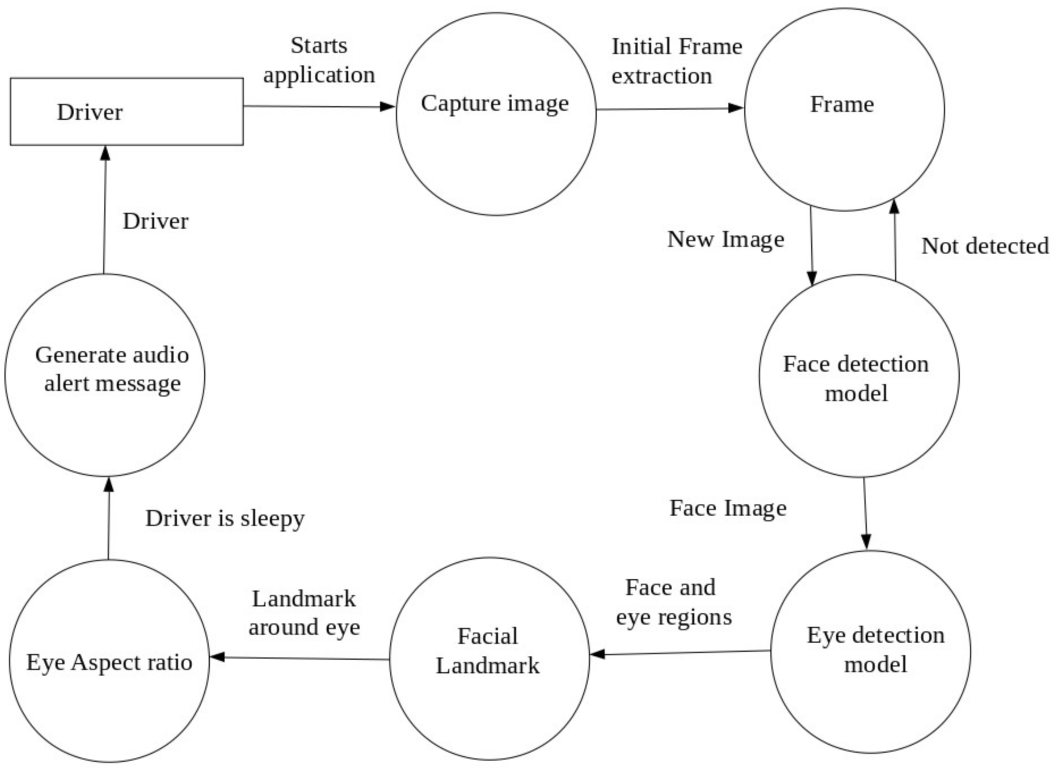
Data Flow Diagram – Level 0



Data Flow Diagram – Level 1



Data Flow Diagram – Level 2



CHAPTER 4: SYSTEM DESIGN

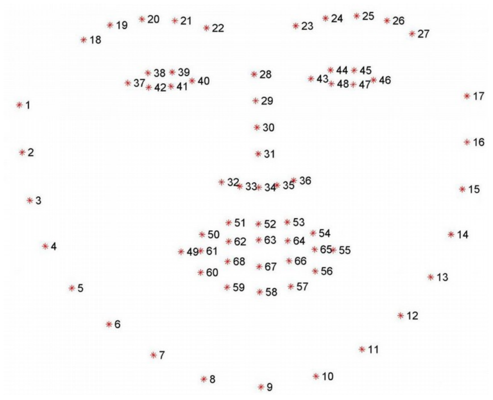
4.1 Main Modules

4.1.1 Facial Landmark detection

Facial Landmark detection is the process of localizing key facial structures on a face, including the eyes, eyebrows, nose, mouth, and jawline, and from these features, in the context of drowsiness detection, we only extract the eye and mouth regions.

• Localize the face in the image.

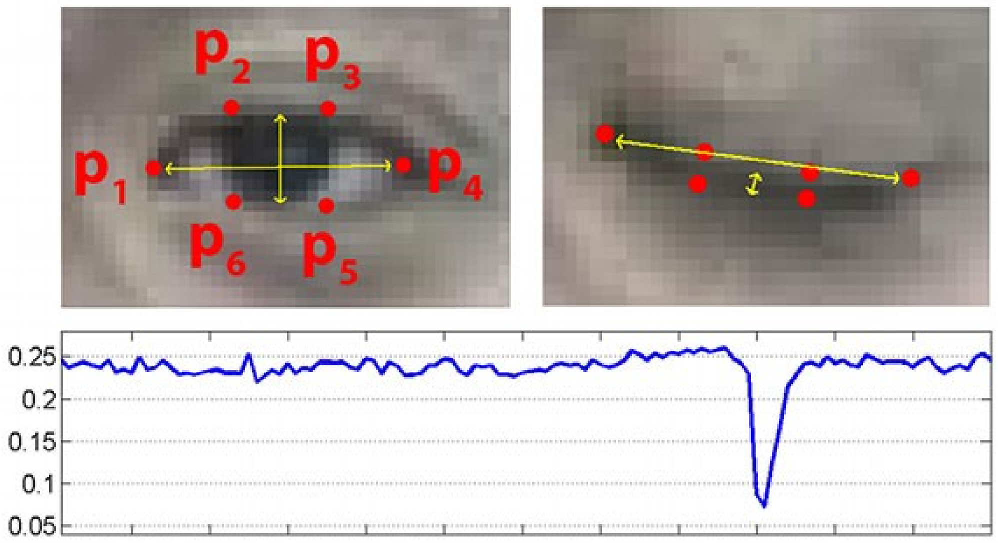
• Detect the key facial structures on the face ROI.



4.1.2 Eye aspect ratio (EAR)

After the eye regions have been extracted, we apply the eye aspect ratio to determine if the eyes are closed. If the eyes have been closed for a sufficiently long enough period of time, we can assume the user is at risk of falling asleep and sound an alarm to grab their attention.

$$\mathrm{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$



4.1.3 Alerting system

A system that can automatically detect driver drowsiness in a real-time video stream, using the above mentioned modules and then alert the driver with sound or alarm if the driver appears to be drowsy.

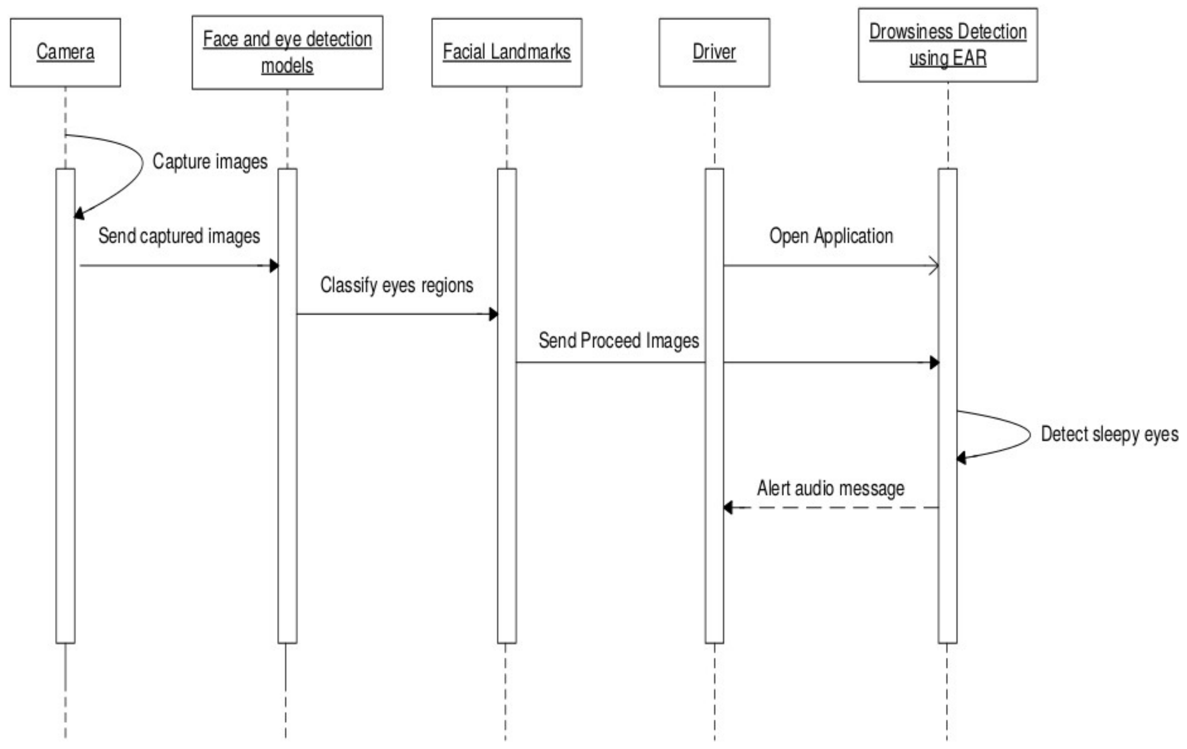
4.2 Procedural Design

4.2.1 Logic Diagrams

4.2.1.1 Sequence Diagrams

A Sequence Diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. UML sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artefact for dynamic modelling, which focuses on identifying the behaviour within your system.

Sequence diagram for Drowsiness detection :



4.2.1.2 Use Case Diagrams

Activity Use Case Diagram Symbols and Notations. Use case diagrams model the functionality of system using actors and use cases. Use case diagrams depict:

System: Draw your system's boundaries using a rectangle that contains use cases. Place actors outside the system's boundaries.

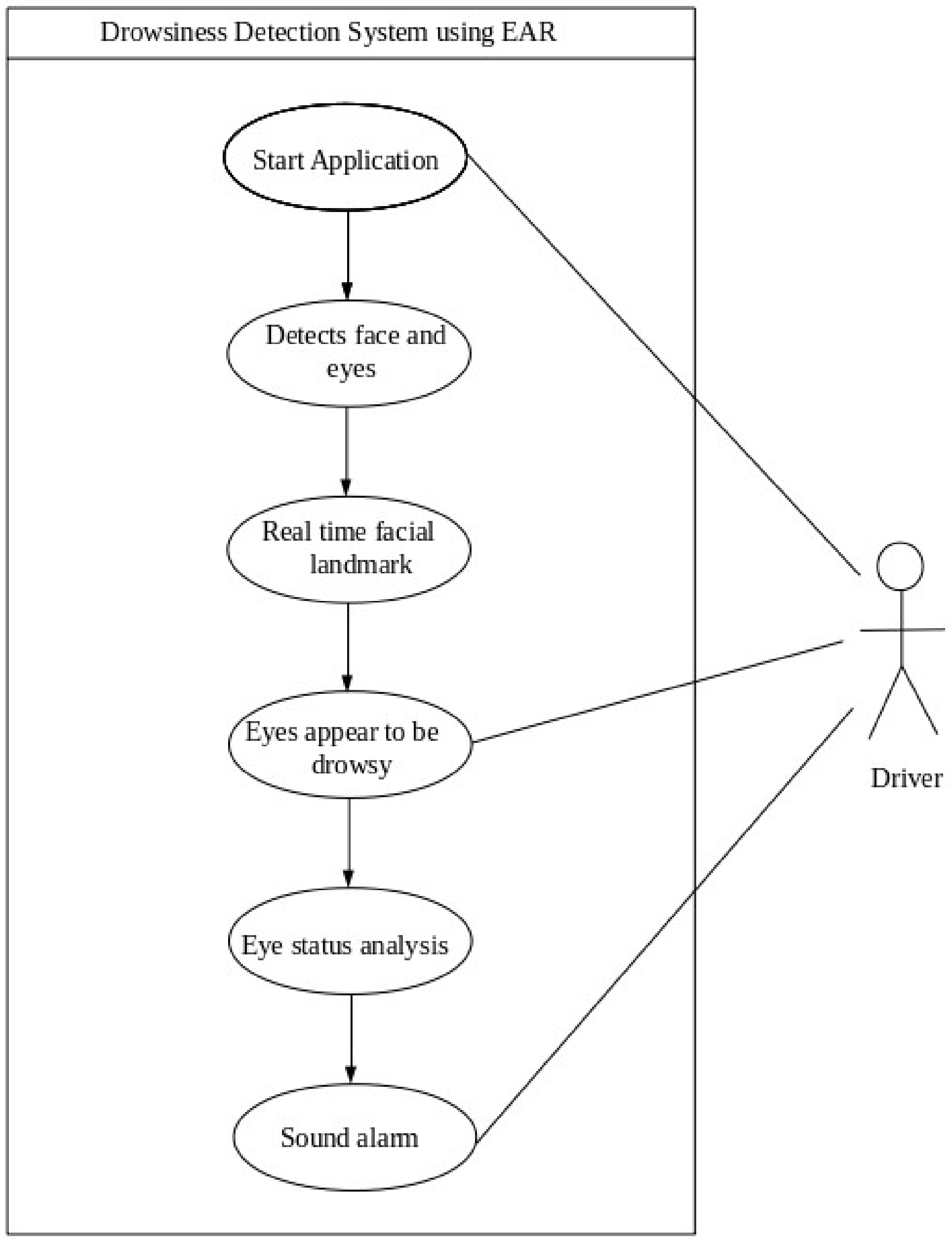
Use Case: Draw use cases using ovals. Label the ovals with verbs that represent the system's functions. A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

Actors: Actors are the users of a system. An actor is a person, organization, or external system that plays a role in one or more interactions with your system. Actors are drawn as stick figures. When one system is the actor of another system, label the actor system.

Scenario: A scenario is one hypothetical instance of how a particular use case might play out. A single use case thus inspires many different scenarios, in the same way that planning a driving trip from one city to another can involve many different routes.

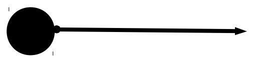
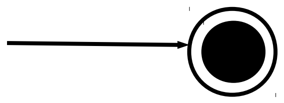
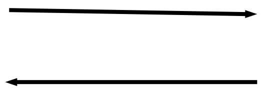
Associations: Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case.

Use Case diagram for Drowsiness detection :

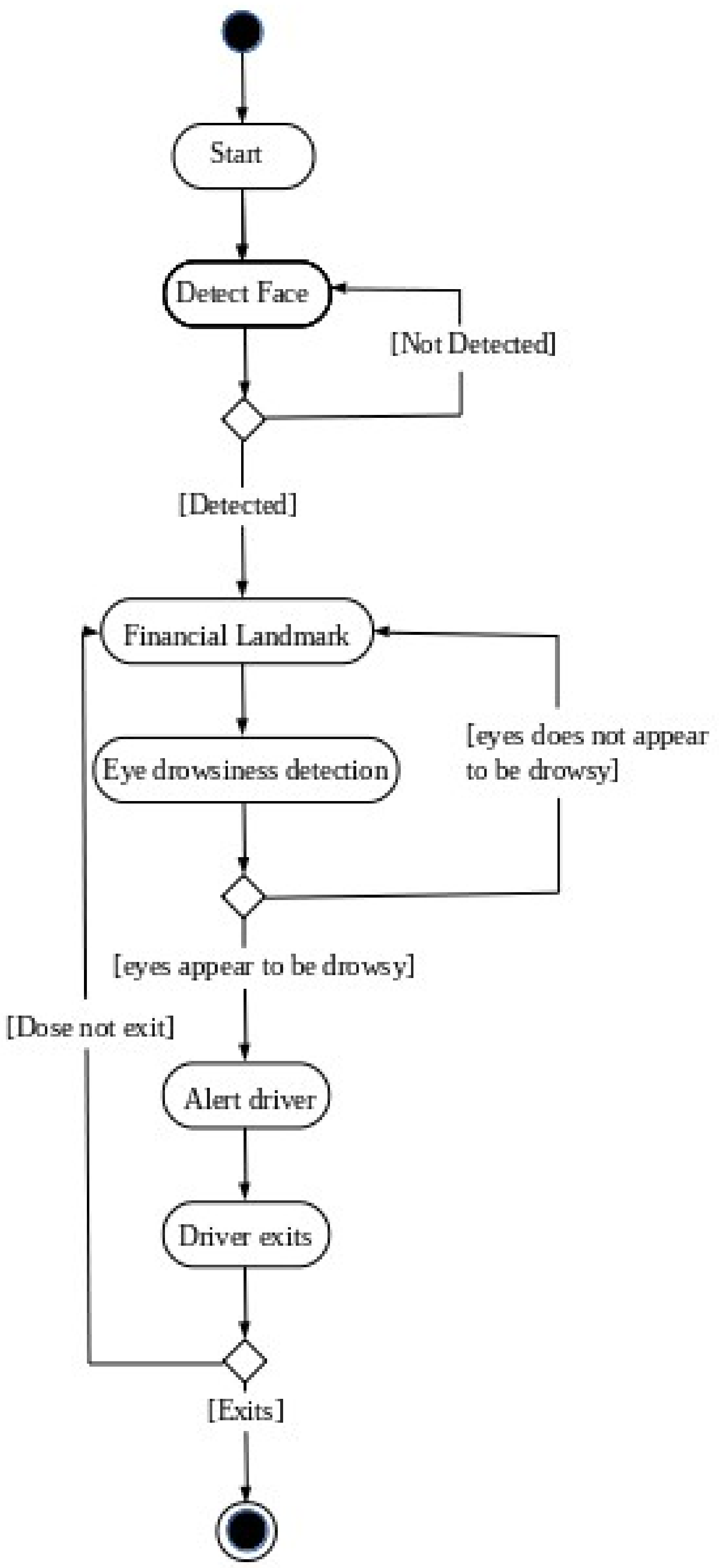


4.2.1.3 Activity Diagram:

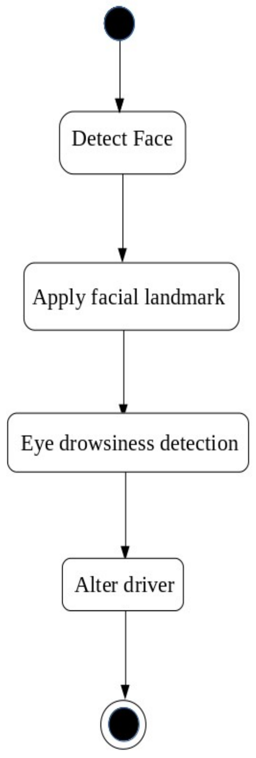
Activity diagrams are used to describe the operations of the system components in step-by-step workflows. It is a dynamic diagram that shows the activity and the event that causes the object to be in a particular state. It is a simple and intuitive illustration of what happens in a workflow, what activities can be done in parallel and whether there are alternative paths through the workflow. Rounded rectangles represent activities, Diamonds represent decisions, a black circle represents the start (initial state) of the work flow, and an encircled black circle represents the end (final state). Arrows run from the start towards the end and represent the order in which activities happen.

Shape	Notation	Description
	Initial Activity	Represents the starting point or first activity of the flow. It is denoted by a solid circle.
	Final Activity	Represents the end of activity diagram, also called as final activity. It is shown by a bull's eye symbol.
	Activity	Represents an activity. It is shown by are rectangle with rounded (almost oval) edges.
	Flow of Activity or control	Shows the direction of the workflow in the activity diagram. It is depicted with and arrow.

Activity diagram for drowsiness detection :



4.2.1.4 State chart diagram for drowsiness detection :



4.4 Security Issues

As the system works on real time data, the system does not store data permanently in any form of data storage. Therefore there is minimum risk in terms of privacy of the driver or the user of the system. And the camera needs to be placed in full view of the driver’s/user face in order to accurately detect the eyes and mouth regions.

4.5 Test Cases Design

Software testing:

It is a process used to identify the correctness, completeness and quality of developed computer software i.e., software testing is to execute a software program with the intent of finding bugs .

Software quality assurance:

It involves the entire software development process monitoring and improving the process, making sure that any agreed upon standards and procedures are followed and ensuring that problems are found and dealt with. It is oriented to ‘Prevention’.

CHAPTER 5 : IMPLEMENTATION AND RESULTS

Filename : drowsiness_detetion.py

```
from scipy.spatial import distance as dist
from imutils.video import VideoStream
from imutils import face_utils
from threading import Thread
import numpy as np
import argparse
import imutils
import time
import dlib
import cv2
import pygame
import os
import tkinter
global alarm_status
global alarm_status2
global saying
def detect():
   #this is for sound
   pygame.init()
   song = pygame.mixer.Sound('alert.wav')
         def alarm():  #this Function for alarm
                  while alarm_status:
         print("Drowsy")
         song.play()
                  if alarm_status2:
         print('yawning')
         saying = True
         song.play()
    def eye_aspect_ratio(eye): # This Is for eyse aspect ratio
      A = dist.euclidean(eye[1], eye[5])
      B = dist.euclidean(eye[2], eye[4])
                  C = dist.euclidean(eye[0], eye[3])
                  ear = (A + B) / (2.0 * C)
                  return ear
          def final_ear(shape):
      (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
      (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
                  leftEye = shape[lStart:lEnd]
      rightEye = shape[rStart:rEnd]
                  leftEAR = eye_aspect_ratio(leftEye)
      rightEAR = eye_aspect_ratio(rightEye)
```

```
                ear = (leftEAR + rightEAR) / 2.0
    return (ear, leftEye, rightEye)
        def lip_distance(shape):
    top_lip = shape[50:53]
    top_lip = np.concatenate((top_lip, shape[61:64]))
                low_lip = shape[56:59]
    low_lip = np.concatenate((low_lip, shape[65:68]))
                top_mean = np.mean(top_lip, axis=0)
    low_mean = np.mean(low_lip, axis=0)
                distance = abs(top_mean[1] - low_mean[1])
    return distance
ap = argparse.ArgumentParser()
ap.add_argument("-w", "--webcam", type=int, default=0,
              help="index of webcam on system")
args = vars(ap.parse_args())
      EYE_AR_THRESH = 0.3
EYE_AR_CONSEC_FRAMES = 30
YAWN_THRESH = 40
alarm_status = False
alarm_status2 = False
saying = False
COUNTER = 0
      print("-> Loading the predictor and detector...")
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
      print("-> Starting Video Stream")
vs = VideoStream(src=args["webcam"]).start()
      time.sleep(1.0)
      while True:
                frame = vs.read()
    frame = imutils.resize(frame, width=750)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                rects = detector(gray, 0)
      for rect in rects:
      shape = predictor(gray, rect)
      shape = face_utils.shape_to_np(shape)
                        eye = final_ear(shape)
      ear = eye[0]
      leftEye = eye[1]
      rightEye = eye[2]
                        distance = lip_distance(shape)
                        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)
                        lip = shape[48:60]
      cv2.drawContours(frame, [lip], -1, (0, 255, 0), 1)
```

```
                    if ear < EYE_AR_THRESH:
            COUNTER += 1
                            if COUNTER >= EYE_AR_CONSEC_FRAMES:
              if alarm_status == False:
                alarm_status = True
                t = Thread(target=alarm)
                t.deamon = True
                t.start()
                                    cv2.putText(frame, "!*DROWSINESS ALERT*!", (10, 30),
                  cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            else:
                COUNTER = 0
                alarm_status = False
                    if (distance > YAWN_THRESH):
            cv2.putText(frame, "***Yawn Alert***", (10, 30),
                 cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            if alarm_status2 == False and saying == False:
              alarm_status2 = True
              t = Thread(target=alarm)
              t.deamon = True
              t.start()
          else:
            alarm_status2 = False
                    cv2.putText(frame, "EYE ASPECT RATIO: {:.2f}".format(ear), (300, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
          cv2.putText(frame, "YAWN RATIO: {:.2f}".format(distance), (300, 60),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
        if key == ord("q"):
          break
      cv2.destroyAllWindows()
      vs.stop()
# Here we us Tkinter for GUI
from tkinter import *
root = Tk()
root.geometry("644x434")
root.minsize(644, 434)
root.maxsize(644, 434)
root.title("DrowsyDetectionSystem")
photo = PhotoImage(file="Computer-Vision.png")
photoLabel = Label(image=photo)
photoLabel.grid()
photoLabel.place(x=0,y=0)
label1 = Label(text="Welcome to Drowsy Detection System", bg="red", fg="white", padx=4,
pady=4,
        font=("comicsansms", 15, "bold"))
```

```
label1.pack()
def run():
    detect()
btn = Button(root, text="Start",bg = "white",padx=4, pady=4, width=40,command=run)
btn.pack()
f1 = Frame(root, bg="grey")
f1.pack(side=BOTTOM)
label2 = Label(f1, text="Please Enter Q to stop the app", bg="green", fg="white", padx=4,
pady=4,
             font=("comicsansms", 15, "bold"))
label2.pack()
root.mainloop()
```

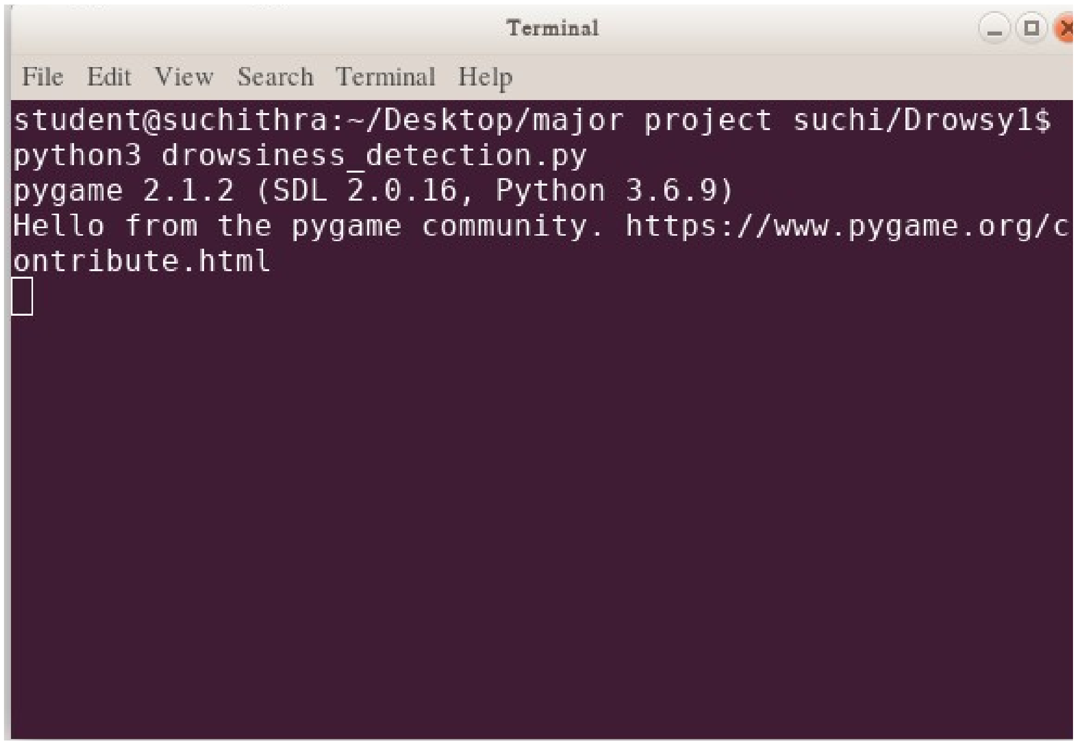
Steps to run the software :

1.Install the necessary packages

2.Open the python files in an IDE.

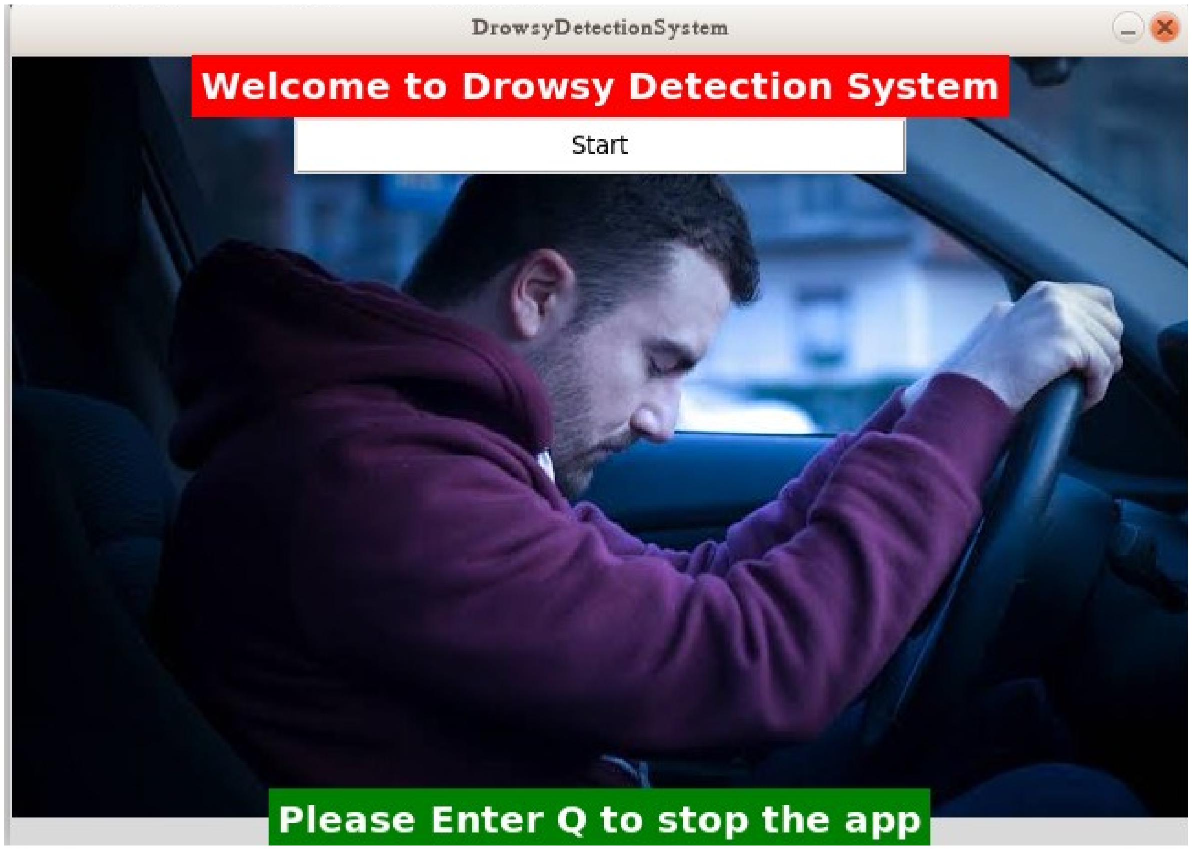
3.Click on the run button and open the terminal/ command promt and type python3 filename.py after entering into the directory.

Terminal Command :

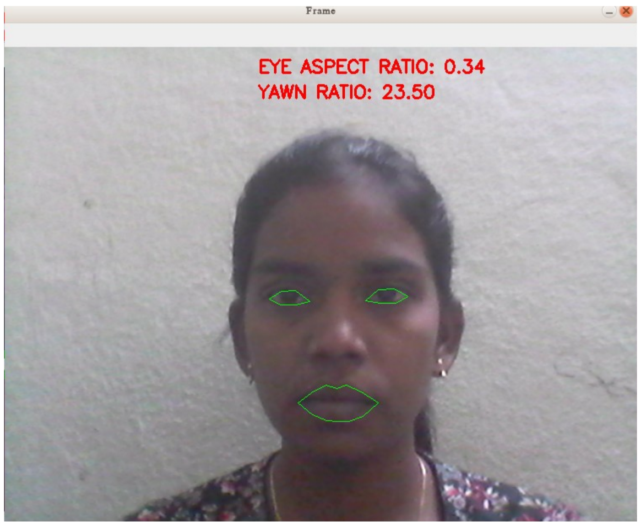


```
student@suchithra:~/Desktop/major project suchi/Drowsy1$
python3 drowsiness_detection.py
pygame 2.1.2 (SDL 2.0.16, Python 3.6.9)
Hello from the pygame community. https://www.pygame.org/c
ontribute.html
```

Frontend View :

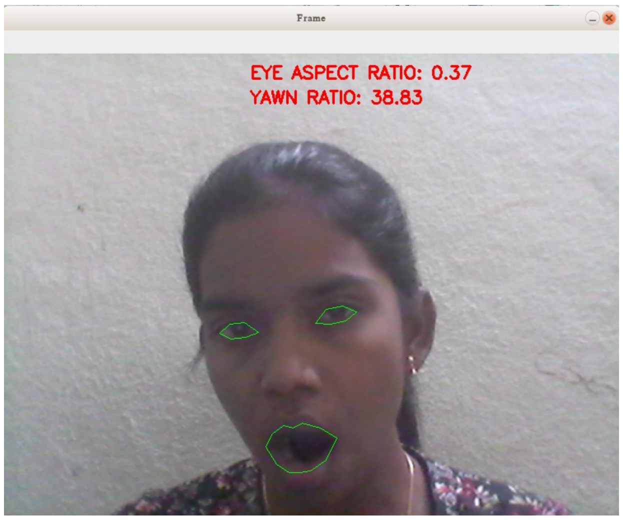

DrowsyDetectionSystem
Welcome to Drowsy Detection System
Start
Please Enter Q to stop the app

Normal frequency ratio :



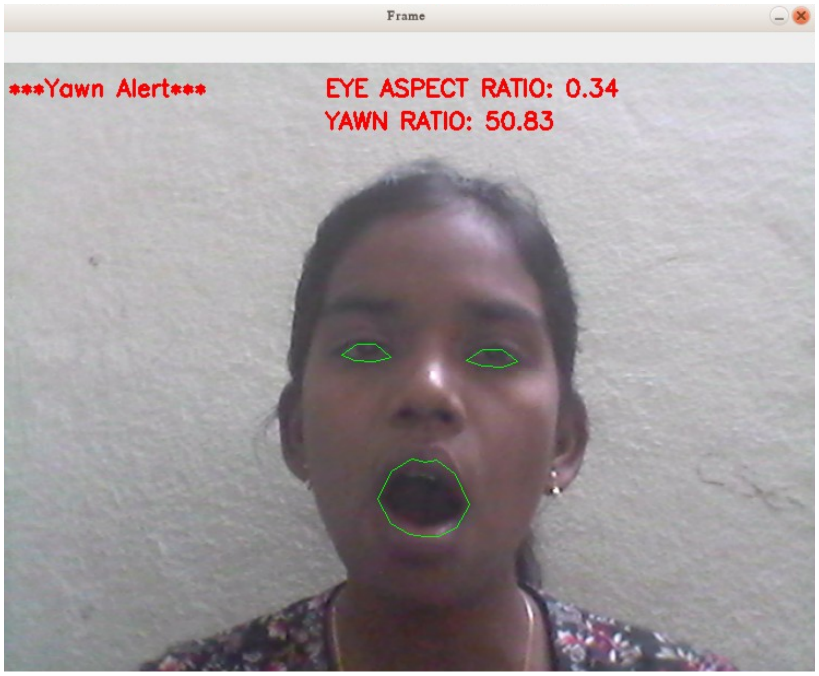
```
EYE ASPECT RATIO: 0.34
YAWN RATIO: 23.50
```

Change in Yawn ratio :



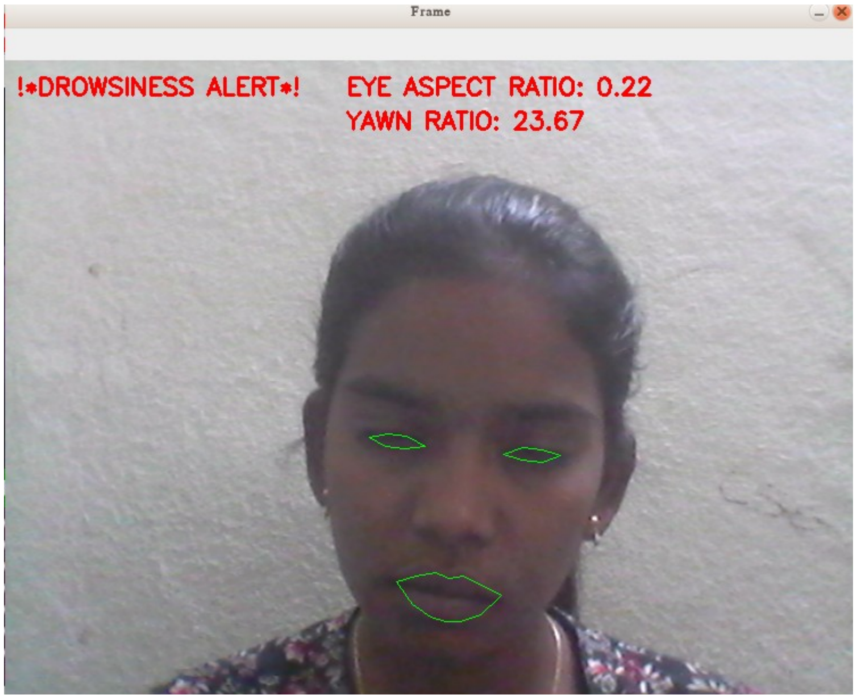
```
EYE ASPECT RATIO: 0.37
YAWN RATIO: 38.83
```

Yawn Alert : Alerting driver when he yawn with alert sound.



```
***Yawn Alert***          EYE ASPECT RATIO: 0.34
                          YAWN RATIO: 50.83
```

Drowsiness Alert : Alerting driver when he is sleepy/drowsy with alert sound.



```
!*DROWSINESS ALERT*!      EYE ASPECT RATIO: 0.22
                          YAWN RATIO: 23.67
```

CHAPTER 6 : RESULTS AND DISCUSSION

Test Reports :

SL.NO	USER ACTION	EXPECTED RESULTS	TEST RESULTS
1	Face of user not facing the camera.	Facial landmark model doesn’t pick up the face from the user.	Successful
2	User is yawning.	Model detects the yawn and alerts the user.	Successful
3	Face of user placed right in the front of camera.	Model detects the face and extracts from user.	Successful
4	User feeling drowsy.	Model detects user as drowsy and alerts the user.	Successful

CHAPTER 7 : CONCLUSIONS

7.1 Conclusion

The drowsiness detection System is mainly useful to the **Drivers**. This helps the drivers to reduce the risk of accidents and also the users in this case the students /employees to stay alert. The "Drowsiness detection System " is an easy and also an efficient way to manage the safety of the drivers.

7.2 Limitations of the System

• May require high quality camera for better results.
• Distance of the face from the camera may also play a vital role.
• It might be annoying if the system rings the alarm when the user is not drowsy.

7.3 Future Scope of the Project

• Our model can be improvised by the following methods : Learning to detect faces and eyes in varied lighting conditions, such as at night with infrared lights. In addition to this, the model should also be able to recognize drowsy eyes with sunglasses.

• With some modification this system can be used in combination with real time cameras to provide alert a driver while he is driving. This will however require exhaustive testing on a larger dataset.

CHAPTER 8 : REFERENCES

<u>**Website Referred**</u>

- https://data-flair.training/blogs/python-project-driver-drowsiness-detection-system/
- https://www.pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/
- https://blog.goodaudience.com/real-time-face-and-eyes-detection-with-opencv-54d9ccfee6a8
- https://towardsdatascience.com/a-guide-to-face-detection-in-python-3eab0f6b9fc1
- https://towardsdatascience.com/face-detection-for-beginners-e58e8f21aad9
- https://datascienceplus.com/face-and-eye-detection-with-opencv/