

# **SKILL DEVELOPMENT LAB**

## **MINI PROJECT REPORT**

### **DRIVER DROWSINESS DETECTION SYSTEM**

*Submitted by*

**AROOP KUMAR(3324)**

**ASHISH(3327)**

**CHAITANYA KUMAR(3349)**



**Department of Computer Engineering**

**ARMY INSTITUTE OF TECHNOLOGY, PUNE**

**2019-20**



**ARMY INSTITUTE OF TECHNOLOGY, PUNE**  
**DEPARTMENT OF COMPUTER ENGINEERING**

**CERTIFICATE**

*This is to certify that the SDL Mini Project  
titled*

**DRIVER DROWSINESS  
DETECTION SYSTEM**

*has been prepared and presented by*

**AROOP KUMAR(3324)**

**ASHISH(3327)**

**CHAITANYA KUMAR(3349)**

**PROJECT INCHARGE**

Prof. P.R Sonawane

**HEAD OF THE DEPARTMENT**

Prof.(Dr.) Sunil R Dhore

## **ACKNOWLEDGEMENT**

This report is based on a Driver Drowsiness Detection System. We are very thankful to Prof. P.R Sonawane for giving us an opportunity to work under his valuable guidance. It is his consistent support which helped us complete this project as early as possible. His advice, guidance, help, insights and regular reminders were instrumental in shaping our studies.

We extend our sincere gratitude to Prof. Dr. S. R. Dhore, Head of Computer Engineering Department for creating a competitive environment and providing us all the essential facilities and encouragement at the department and institute level.

We would also like to acknowledge all our friends who assisted us in preparing, presenting and submitting this project.

Team Members

## **Abstract**

Road accidents are a major cause of deaths, injuries and property damage every year. Various reports by the World Health Organisation (WHO) indicate that the total number of road traffic deaths across the world has plateaued at 1.25 million per year, with the highest fatality rates in low and middle-income countries. Drowsy and fatigued drivers are often attributed to be responsible for this deplorable situation.

The development of technologies for detecting & preventing drowsiness at the wheel has thus become a major challenge in the field of accident avoidance systems.

Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its affects. Our project does the same.

The aim of this project is to develop a prototype drowsiness detection system. The focus will be placed on designing a system that will accurately monitor the open or closed state of the driver's eyes in real-time using OpenCV. The system will monitor the driver's eye movements & blink patterns using a camera. A warning alarm will be issued if a prolonged blink is observed.

This whole system can be deployed on portable hardware in the near future which could be easily installed in the car for use. The project will thus be able to lower the risk of fatal road accidents to a great extent.

## TABLE OF CONTENTS

<b>SNO.</b>	<b>TITLE</b>	<b>PAGE NO</b>
	<b>CERTIFICATE</b>	<b>2</b>
	<b>ACKNOWLEDGEMENT</b>	<b>3</b>
	<b>ABSTRACT</b>	<b>4</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>8</b>
	1.1 Overview	8
	1.2 Motivation	8
	1.3 Problem Statement	8
	1.4 Objective	9
	1.5 Approach	9
<b>2</b>	<b>SOFTWARE REQUIREMENT SPECIFICATION</b>	<b>10</b>
<b>3</b>	<b>THEORETICAL BACKGROUND</b>	<b>13</b>
	3.1 Client-Server Model	13
	3.2 OpenCV	14
	3.3 Dlib Toolkit	15
	3.4 PyCharm	15
	3.5 YOLO	16
<b>4</b>	<b>HARDWARE/SOFTWARE REQUIREMENTS</b>	<b>18</b>
	4.1 Processor	18
	4.2 Operating System	18
	4.3 Programming Language	18
	4.4 IDE	18

<b>5</b>	<b>IMPLEMENTATION</b>	<b>19</b>
	5.1 Code Snippet	19
	5.2 Program Output	22
<b>6</b>	<b>CONCLUSION &amp; FUTURE SCOPE</b>	<b>23</b>
	6.1 Conclusion	23
	6.2 Future scope	23
<b>7</b>	<b>REFERENCES</b>	<b>24</b>

## **LIST OF FIGURES**

<b>SR NO</b>	<b>NAME OF FIGURE</b>	<b>PAGE NO.</b>
1	<b>CLIENT-SERVER ARCHITECHTURE</b>	<b>13</b>
2	<b>YOLO ARCHITECHTURE</b>	<b>17</b>

# **CHAPTER 1 - INTRODUCTION**

## **1.1 OVERVIEW**

The aim of this project is to develop a prototype drowsiness detection system. The focus will be placed on designing a system that will accurately monitor the open or closed state of the driver's eyes in real-time using OpenCV. The system will monitor the driver's eye movements & blink patterns using a camera. A warning alarm will be issued if a prolonged blink is observed.

This whole system can be deployed on portable hardware in the near future which could be easily installed in the car for use. The project will thus be able to lower the risk of fatal road accidents to a great extent.

## **1.2 MOTIVATION**

The motivation for doing this project was primarily an interest in undertaking a challenging project in the domain of computer vision. The opportunity to learn about various aspects of OpenCV and object detection was appealing. Computer Vision as a field is still in the developing stages and hence, we took up this project as it has a promising future.

## **1.3 PROBLEM STATEMENT**

The project would focus on developing a prototype driver drowsiness detection system using OpenCV. Driver drowsiness is recognized as an important factor in the vehicle accidents. It was demonstrated that driving performance deteriorates with increased drowsiness with resulting crashes constituting more than 20% of all vehicle accidents. But the life lost once cannot be re-winded. Advanced technology offers some hope avoid these up to some extent. This project involves measure and controls the eye blink.



## **1.4 OBJECTIVE**

The objective of the project is to develop a prototype drowsiness detection system. The focus will be placed on designing a system that will accurately monitor the open or closed state of the driver's eyes in real-time using OpenCV. The system will monitor the driver's eye movements & blink patterns using a camera. A warning alarm will be issued if a prolonged blink is observed.

## **1.5 APPROACH**

The aim of this project is to develop a prototype drowsiness detection system. The focus will be placed on designing a system that will accurately monitor the open or closed state of the driver's eyes in real-time using OpenCV. The system will monitor the driver's eye movements & blink patterns using a camera. A warning alarm will be issued if a prolonged blink is observed. The algorithm working in the backend is YOLO (You Only Look Once). This algorithm helps in providing a high frame rate during face detection. The project has thus been very agile and any future changes can be easily incorporated within it.

## **CHAPTER 2 – SOFTWARE REQUIREMENT SPECIFICATION**

- **Overall Description**

### *2.1 Product Perspective*

- 2.1.1 Software Interface: The system will provide an eye tracking interface which will monitor the blink patterns of the driver using a webcam embedded into the vehicle itself.
- 2.1.2 Hardware Interface: The system will consist of a webcam and an Arduino interface to control its operation.
- 2.1.3 Communications Interface: The system components will coordinate by means of an inbuilt Wi-Fi network module. A weekly report will be generated which will provide the blink patterns of the user.

### *2.2 Product Functions*

The product functions are broken down into four use cases. A Simulate Camera use case is provided to remotely read and send images to the Track Eye use case for further processing. Next, the primary use case, Track Eye, takes images and executes the YOLO algorithm and provides the results to the Warning Alarm use case which generate a warning signal at appropriate moments. The Track Eye use case also provides both tracked and candidate target information to the Log Target Data use case which records the information to a log file. The log

file will be used for generating a report of the blink patterns of the driver under consideration.

### 2.3 Constraints, Assumptions and Dependencies

For simplicity and ensuring computability, the system makes the following assumptions:

- Users are expected to look at the augmented view provided on the screen.
- Camera should not have any flaws on its lens.
- There should not be any obstacle between the camera and the driver.
- The Alarm system must not be flawed.

In addition, system will be dependent on the following constraints:

- a. Heavy processing will be done in limited time thus computer must have more than two CPU cores to handle concurrent processing better.
- b. Lighting of the area, air effects like fog might reduce precision. System should work on a general computer/mobile system. It will not contain any hardware designed for image processing. Implementation will be done taking this constraint into account.

- **Specific Requirements**

- a. Interface Requirements

The primary input to the system is the image acquired from the device, i.e. camera. After object recognition and eye tracking phases, this image is used in augmented reality phase with the resultant information.

Eventually, the watcher can see enhanced view of the area on the screen.

Thus, the camera should be integrated with the vehicle through a mobile handset or a self-sufficient isolated camera embedded in it.

*b. Software Requirements*

The System will provide a single shot detector interface for the driver through the application. PyCharm python IDE has been used as it enhances productivity while coding by providing features like suggestions, Local VCS, Autocomplete etc. OpenCV libraries need to be coupled with high performance Dlib utilities for efficient working of the application. YOLO algorithm has been deployed for detection purposes. On detection, the camera images need to be stored in a database from where they can be retrieved for further processing. Thus, a suitable database must be present to accommodate such a large dataset.

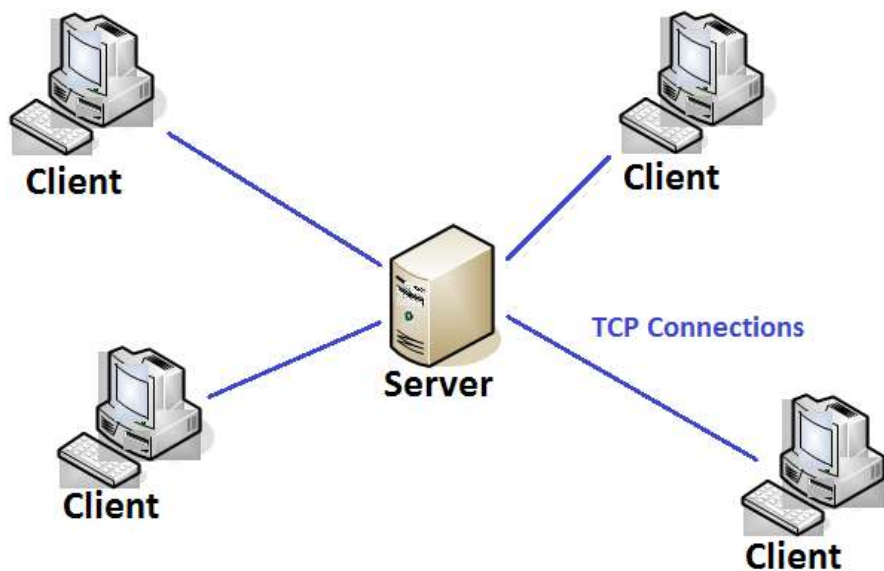
*c. Hardware and Communication Requirements*

The main hardware component required for the project is a web-camera which will capture images for detection. The system components will coordinate by means of an inbuilt Wi-Fi network module. Thus, Wireless internet is the fundamental requirement for the project.

## **CHAPTER 3 – THEORETICAL BACKGROUND**

### **3.1 Client-Server Architecture**

Client/server architecture is a producer/consumer computing architecture where the server acts as the producer and the client as a consumer. The server houses and provides high-end, computing-intensive services to the client on demand. These services can include application access, storage, file sharing, printer access and/or direct access to the server's raw computing power. Client/server architecture works when the client computer sends a resource or process request to the server over the network connection, which is then processed and delivered to the client. A server computer can manage several clients simultaneously, whereas one client can be connected to several servers at a time, each providing a different set of services. In its simplest form, the internet is also based on client/server architecture where web servers serve many simultaneous users with website data.



## 3.2 OpenCV

OpenCV was started at Intel in 1999 by Gary Bradsky and the first release came out in 2000. Vadim Pisarevsky joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle who won 2005 DARPA Grand Challenge. Later its active development continued under the support of Willow Garage, with Gary Bradsky and Vadim Pisarevsky leading the project. Right now, OpenCV supports a lot of algorithms related to Computer Vision and Machine Learning and it is expanding day-by-day.

Currently OpenCV supports a wide variety of programming languages like C++, Python, Java etc and is available on different platforms including Windows, Linux, OS X, Android, iOS etc. Also, interfaces based on CUDA and OpenCL are also under active development for high-speed GPU operations.

OpenCV-Python is the Python API of OpenCV. It combines the best qualities of OpenCV C++ API and Python language. It is a library of Python bindings designed to solve computer vision problems. OpenCV-Python makes use of NumPy, which is a highly optimized library for numerical operations with a MATLAB-style syntax.

All the OpenCV array structures are converted to and from NumPy arrays. This also makes it easier to integrate with other libraries that use NumPy such as SciPy and Matplotlib. Thus, it is highly suitable for creating optimized eye tracking models with significantly high frame rates.

### 3.3 Dlib Toolkit

Dlib is a general-purpose cross-platform software library written in the programming language C++. It is used in both industry and academia in a wide range of domains including robotics, embedded devices, mobile phones, and large high-performance computing environments.

Since development began in 2002, Dlib has grown to include a wide variety of tools. As of 2016, it contains software components for dealing with networking, threads, graphical user interfaces, data structures, linear algebra, machine learning, image processing, data mining, XML and text parsing, numerical optimization, Bayesian networks, and many other tasks. In recent years, much of the development has been focused on creating a broad set of statistical machine learning tools and in 2009 Dlib was published in the *Journal of Machine Learning Research*. Since then it has been used in a wide range of domains.

### 3.4 PyCharm

PyCharm is an integrated development environment used in computer programming, specifically for the Python language. It can be customized with themes and plugins. It also enhances productivity while coding by providing some features like suggestions, Local VCS, Autocomplete etc. This makes it suitable for the project.

### 3.5 YOLO (You Only Look Once)

YOLO is an extremely fast real time multi object detection algorithm. YOLO stands for “You Only Look Once”. The algorithm applies a neural network to an entire image. The network divides the image into an  $S \times S$  grid and comes up with bounding boxes, which are boxes drawn around images and predicted probabilities for each of these regions. The method used to come up with these probabilities is logistic regression. The bounding boxes are weighted by the associated probabilities. For class prediction, independent logistic classifiers are used.

When it comes to deep learning-based object detection, there are three primary object detectors you’ll encounter:

- R-CNN and their variants, including the original R-CNN, Fast R- CNN, and Faster R-CNN
- Single Shot Detector (SSDs)
- YOLO

While R-CNNs tend to very accurate, the biggest problem with the R-CNN family of networks is their speed — they were incredibly slow, obtaining only 5 FPS on a GPU.

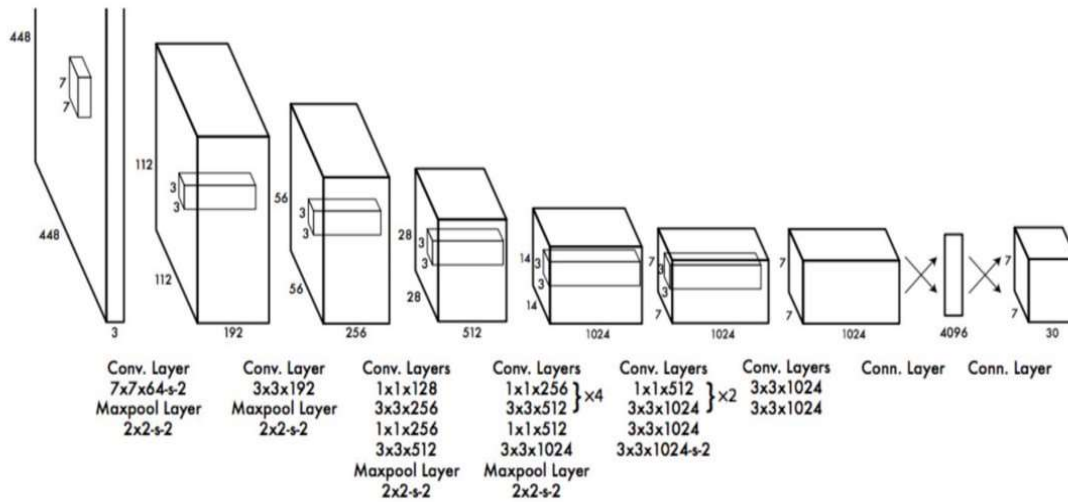
To help increase the speed of deep learning-based object detectors, both Single Shot Detectors (SSDs) and YOLO use a *one-stage detector strategy*.

These algorithms treat object detection as a regression problem, taking a given input image and simultaneously learning bounding box coordinates and corresponding class label probabilities.

In general, single-stage detectors tend to be less accurate than two-stage detectors *but are significantly faster*.

YOLO Architecture has been shown below





**Figure 3: The Architecture.** Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating  $1 \times 1$  convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution ( $224 \times 224$  input image) and then double the resolution for detection.

Biggest advantages:

- Speed (45 frames per second — better than realtime)
- Network understands generalized object representation (This allowed them to train the network on real world images and predictions on artwork was still fairly accurate).
- Faster version (with smaller architecture) — 155 frames per sec but is less accurate.

## **CHAPTER 4 - HARDWARE / SOFTWARE**

### **REQUIREMENTS**

#### **4.1 Processor**

Intel Core 2 duo/i3/i5/i7 - 64 bit processors are recommended. The project has been developed in intel i7-64 bit.

#### **4.2 Operating System**

For the purpose of this project we have used Windows 10 64 bit.

#### **4.3 Programming Language – Python**

The project has been developed using OpenCV python APIs. Python is an interpreted, high-level, general-purpose programming language.

Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.

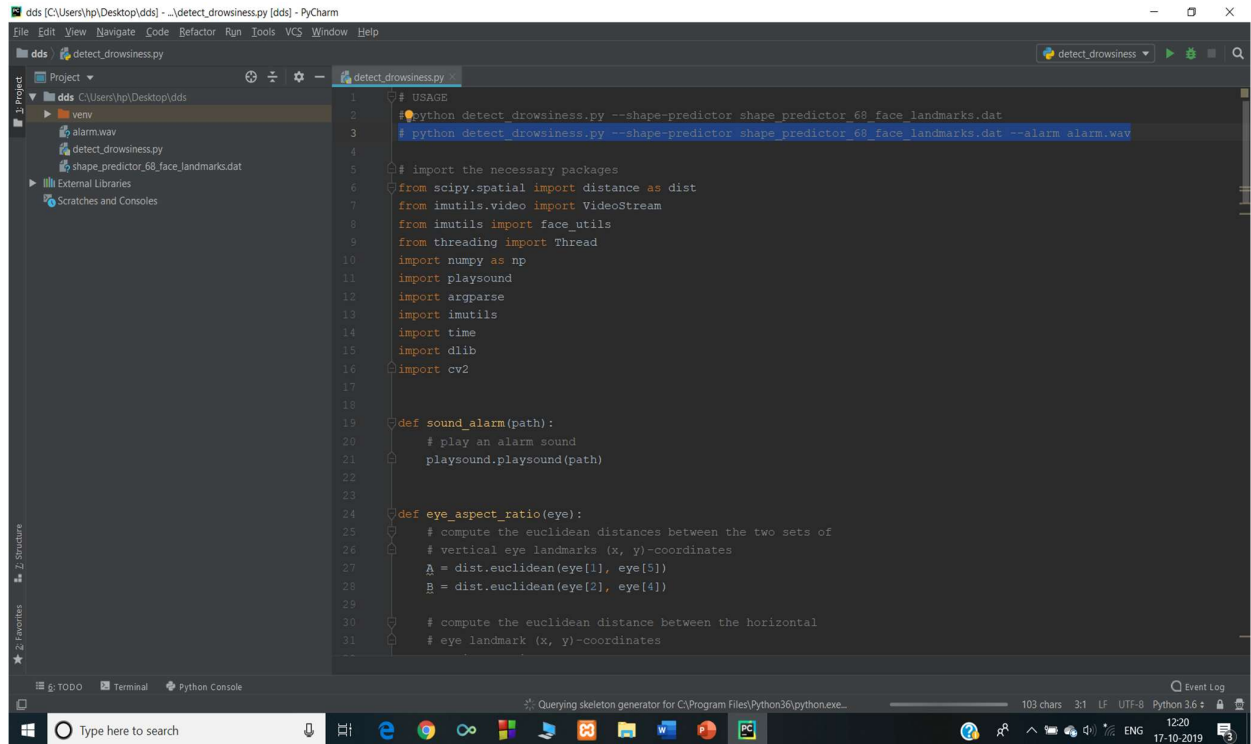
Our project has been developed using python 3.6.8 as it provides all utilities required for the project.

#### **4.4 Integrated Development Environment - PyCharm**

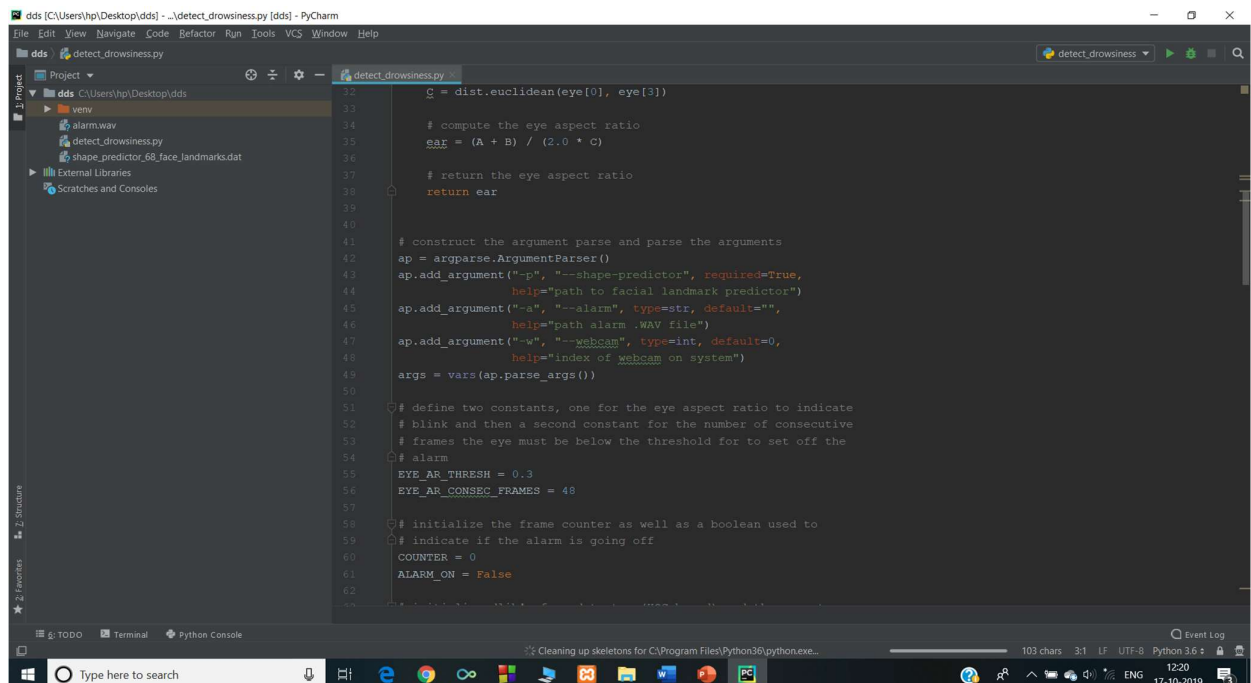
PyCharm is an integrated development environment used in computer programming, specifically for the Python language. It can be customized with themes and plugins. It also enhances productivity while coding by providing some features like suggestions, Local VCS, Autocomplete etc. This makes it suitable for the project.

# CHAPTER 5 - IMPLEMENTATION

## 5.1 CODE SNIPPET



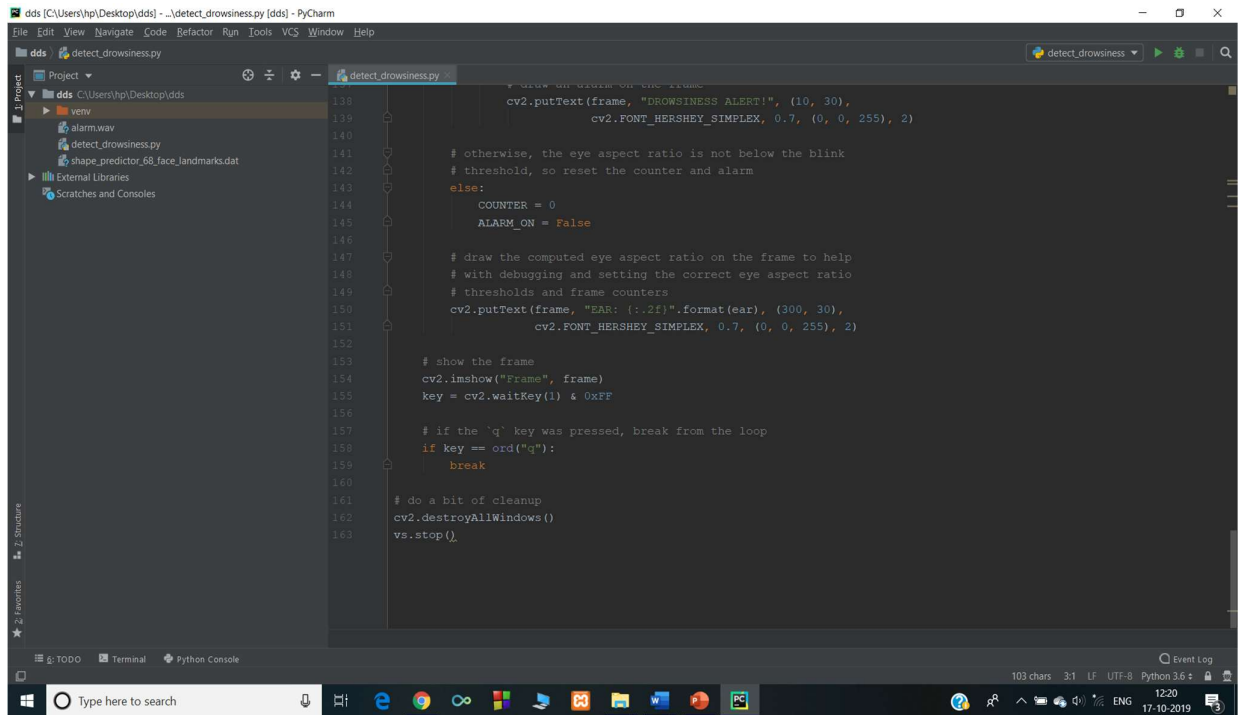
```
1  # USAGE
2  python detect_drowsiness.py --shape-predictor shape_predictor_68_face_landmarks.dat
3  python detect_drowsiness.py --shape-predictor shape_predictor_68_face_landmarks.dat --alarm alarm.wav
4
5  # import the necessary packages
6  from scipy.spatial import distance as dist
7  from imutils.video import VideoStream
8  from imutils import face_utils
9  from threading import Thread
10 import numpy as np
11 import playsound
12 import argparse
13 import imutils
14 import time
15 import dlib
16 import cv2
17
18
19 def sound_alarm(path):
20     # play an alarm sound
21     playsound.playsound(path)
22
23
24 def eye_aspect_ratio(eye):
25     # compute the euclidean distances between the two sets of
26     # vertical eye landmarks (x, y)-coordinates
27     A = dist.euclidean(eye[1], eye[5])
28     B = dist.euclidean(eye[2], eye[4])
29
30     # compute the euclidean distance between the horizontal
31     # eye landmark (x, y)-coordinates
```



```
32     C = dist.euclidean(eye[0], eye[3])
33
34     # compute the eye aspect ratio
35     EAR = (A + B) / (2.0 * C)
36
37     # return the eye aspect ratio
38     return ear
39
40
41 # construct the argument parse and parse the arguments
42 ap = argparse.ArgumentParser()
43 ap.add_argument("-p", "--shape-predictor", required=True,
44                 help="path to facial landmark predictor")
45 ap.add_argument("-a", "--alarm", type=str, default="",
46                 help="path alarm .wav file")
47 ap.add_argument("-w", "--webcam", type=int, default=0,
48                 help="index of webcam on system")
49 args = vars(ap.parse_args())
50
51
52 # define two constants, one for the eye aspect ratio to indicate
53 # blink and then a second constant for the number of consecutive
54 # frames the eye must be below the threshold for to set off the
55 # alarm
56 EYE_AR_THRESH = 0.3
57 EYE_AR_CONSEC_FRAMES = 48
58
59 # initialize the frame counter as well as a boolean used to
60 # indicate if the alarm is going off
61 COUNTER = 0
62 ALARM_ON = False
```

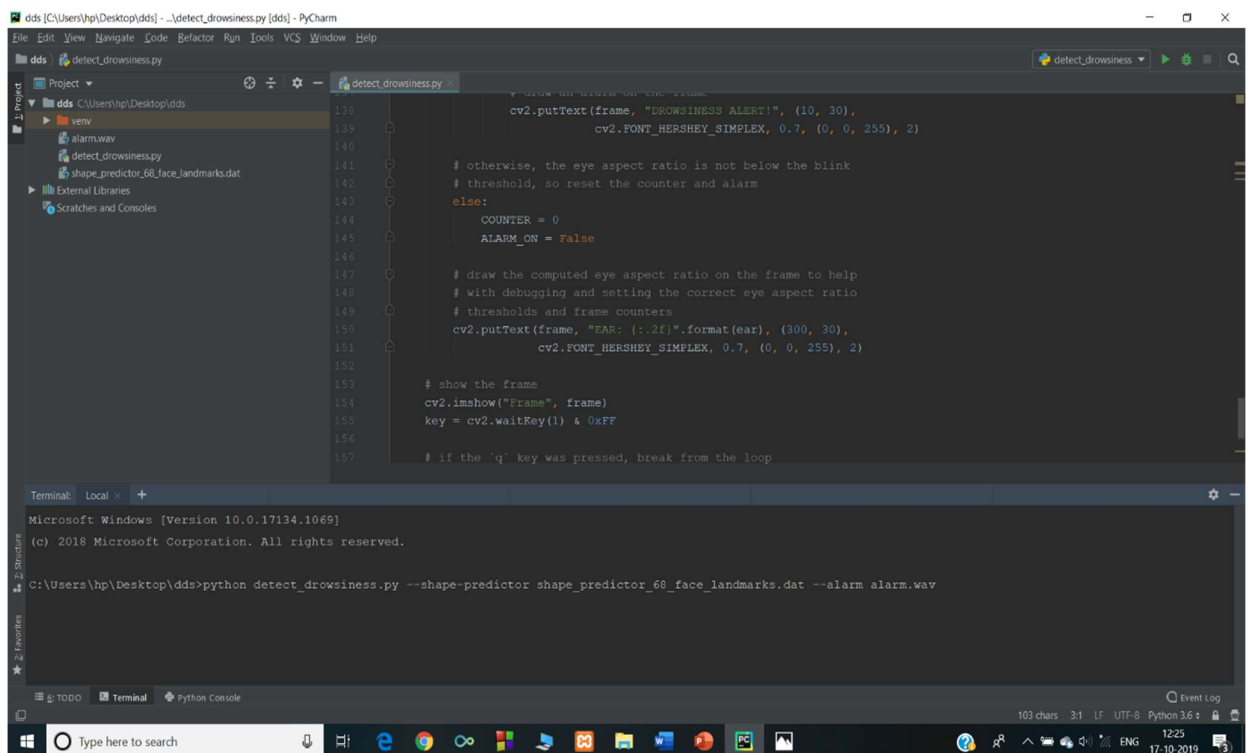
```
dds [C:\Users\hp\Desktop\dds] - ...detect_drowsiness.py [dds] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Project detect_drowsiness.py
Project
  dds C:\Users\hp\Desktop\dds
    venv
    alarm.wav
    detect_drowsiness.py
    shape_predictor_68_face_landmarks.dat
  External Libraries
  Scratches and Consoles
detect_drowsiness.py
62 # initialize dlib's face detector (HOG-based) and then create
63 # the facial landmark predictor
64 print("[INFO] loading facial landmark predictor...")
65 detector = dlib.get_frontal_face_detector()
66 predictor = dlib.shape_predictor(args["shape_predictor"])
67
68 # grab the indexes of the facial landmarks for the left and
69 # right eye, respectively
70 (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
71 (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
72
73 # start the video stream thread
74 print("[INFO] starting video stream thread...")
75 vs = VideoStream(src=args["webcam"]).start()
76 time.sleep(1.0)
77
78 # loop over frames from the video stream
79 while True:
80     # grab the frame from the threaded video file stream, resize
81     # it, and convert it to grayscale
82     # channels)
83     frame = vs.read()
84     frame = imutils.resize(frame, width=450)
85     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
86
87     # detect faces in the grayscale frame
88     rects = detector(gray, 0)
89
90     # loop over the face detections
91     for rect in rects:
92         # determine the facial landmarks for the face region, then
93         # compute the eye aspect ratio for the face region, then
94         # check to see if the eye aspect ratio is below the blink
95         # threshold, and if so, increment the blink frame counter
96         # if ear < EYE_AR_THRESH:
97             # counter += 1
98
99         # if the eyes were closed for a sufficient number of
100         # then sound the alarm
101         if counter >= EYE_AR_CONSEC_FRAMES:
102             # if the alarm is not on, turn it on
103             if not ALARM_ON:
104                 ALARM_ON = True
105
106             # check to see if an alarm file was supplied,
107             # and if so, start a thread to have the alarm
108             # sound played in the background
109             if args["alarm"] != "":
110                 t = Thread(target=sound_alarm,
111                           args=(args["alarm"],))
112                 t.daemon = True
113                 t.start()
114
115             # draw an alarm on the frame
116             cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
117                       cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
118
119         # otherwise, the eye aspect ratio is not below the blink
120         # threshold, so reset the counter and alarm
121         else:
122             counter = 0
123             ALARM_ON = False
```

```
dds [C:\Users\hp\Desktop\dds] - ...detect_drowsiness.py [dds] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Project detect_drowsiness.py
Project
  dds C:\Users\hp\Desktop\dds
    venv
    alarm.wav
    detect_drowsiness.py
    shape_predictor_68_face_landmarks.dat
  External Libraries
  Scratches and Consoles
detect_drowsiness.py
115 # check to see if the eye aspect ratio is below the blink
116 # threshold, and if so, increment the blink frame counter
117 if ear < EYE_AR_THRESH:
118     counter += 1
119
120 # if the eyes were closed for a sufficient number of
121 # then sound the alarm
122 if counter >= EYE_AR_CONSEC_FRAMES:
123     # if the alarm is not on, turn it on
124     if not ALARM_ON:
125         ALARM_ON = True
126
127     # check to see if an alarm file was supplied,
128     # and if so, start a thread to have the alarm
129     # sound played in the background
130     if args["alarm"] != "":
131         t = Thread(target=sound_alarm,
132                   args=(args["alarm"],))
133         t.daemon = True
134         t.start()
135
136     # draw an alarm on the frame
137     cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
138               cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
139
140 # otherwise, the eye aspect ratio is not below the blink
141 # threshold, so reset the counter and alarm
142 else:
143     counter = 0
144     ALARM_ON = False
145
146
```

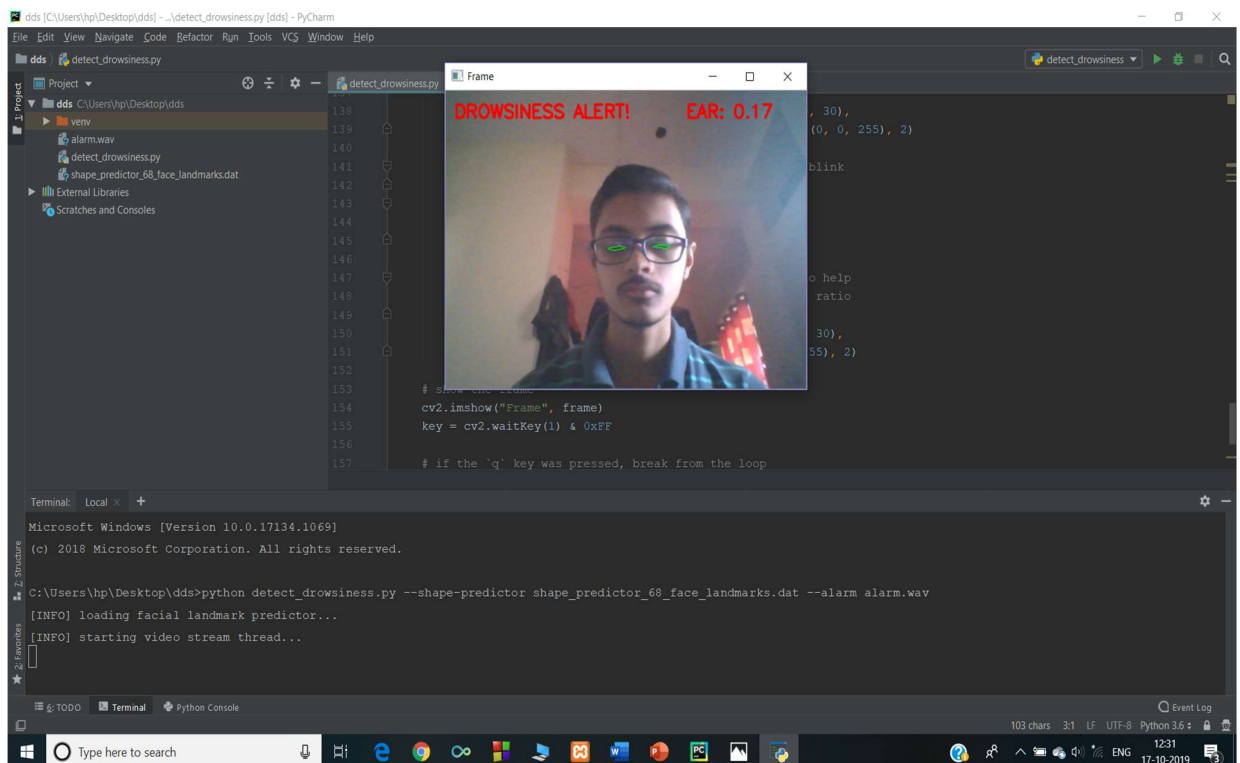
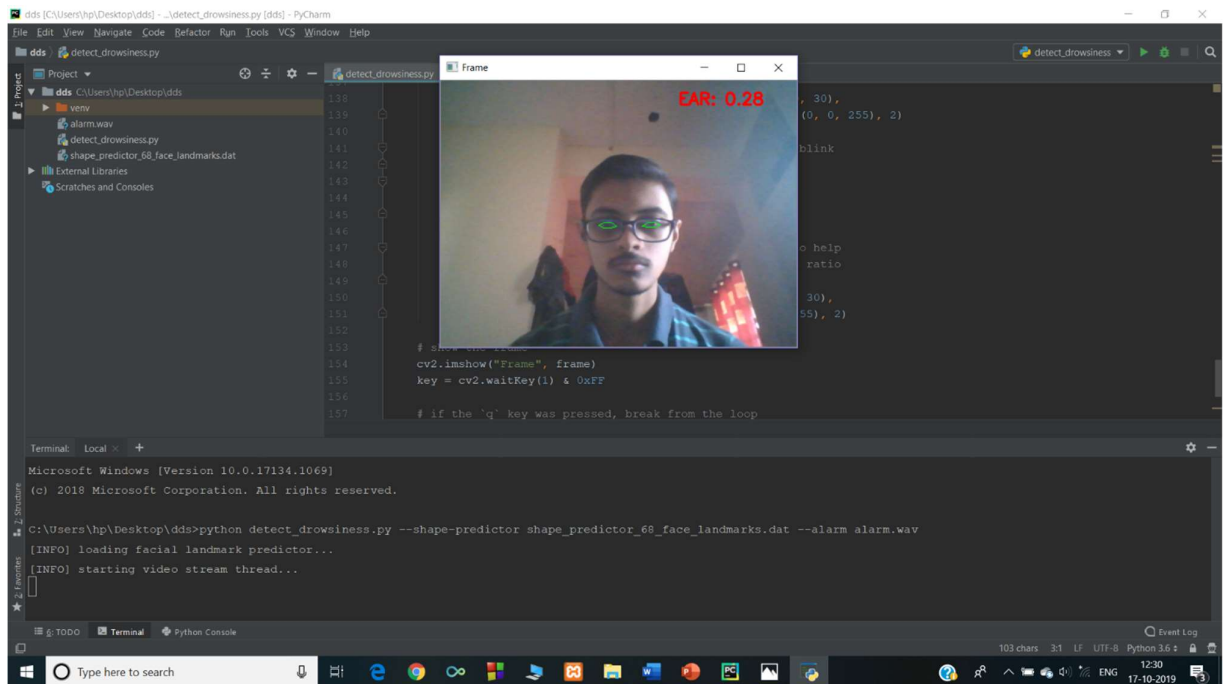


The program is then executed on the terminal using the following command -

```
python detect_drowsiness.py --shape-predictor
shape_predictor_68_face_landmarks.dat --alarm alarm.wav
```



## 5.2 Output Snapshots



## **CHAPTER 6 - CONCLUSION & FUTURE SCOPE**

### **6.1 CONCLUSION**

To sum up, we successfully developed a Driver Drowsiness Detection System in OpenCV using YOLO algorithm. The project if deployed could help us achieve the goal of preventing fatal road accidents and promoting safe driving. The need of the hour is that this system is developed into a more comprehensive Advanced Driver Assistant System so that it can be deployed as soon as possible.

### **6.2 FUTURE SCOPE**

The project is in its early stages and is merely a prototype. It can be further developed in the future so as to be deployed as a full-fledged Advanced Driver Assistant System (ADAS). The project can expand into diverse field of applications like drunk driving detection, accident avoidance systems, traffic monitoring and vehicle law enforcement. The system could also be integrated with self-driving cars whereby the ADAS could take control of the vehicle if the driver is found to be drowsy. Thus the project is flexible enough to implement such advancements and due to this reason it has a very bright future.

## **CHAPTER 7 – REFERENCES**

- [1]. Stephen H. "*Impairment of driving performance caused by sleep deprivation or alcohol. A comparative study*". Human Factors, 1999, 41(1): 118-128.
- [2]. Lee M, Ranganath S. Pose, "*Invariant face recognition using a 3D deformable model*". Pattern Recognition, 2003, 36(8):1835-1846.
- [3]. S. Vitabile, A. Paola and F. Sorbello, "*Bright Pupil Detection in an Embedded, Real-time Drowsiness Monitoring System*", in 24th IEEE International Conference on Advanced Information Networking and Applications, 2010.
- [4]. B. Bhowmik and C. Kumar, "*Detection and Classification of Eye State in IR Camera for Driver Drowsiness Identification*", in Proceeding of the IEEE International Conference on Signal and Image Processing Applications, 2009.
- [5]. N. Otsu, "*A Threshold Selection Method from Gray-Level Histograms*", IEEE Transactions on Systems, Man and Cybernetics, pp. 62-66, 1979.
- [6]. T. Hong, H. Qin and Q. Sun, "*An Improved Real Time Eye State Identification System in Driver Drowsiness Detection*", in proceeding of the IEEE International Conference on Control and Automation, Guangzhou, CHINA, 2007.
- [7]. Z. Tian et H. Qin, "*Real-time Driver's Eye State Detection*", in Proceedings of the IEEE International Conference on Vehicular Electronics and Safety, October 2005.
- [8]. M. S. Nixon and A. S. Aguado, Feature Extraction and Image Processing, 2nd ed., Jordan Hill, Oxford OX2 8DP, UK, 2008.