# A Major Project Report

**On**

### “Lightweight CNN-based driver eye status smart surveillance for vehicles”

#### Submitted in Partial Fulfillment of the Academic Requirement for the Award of the Degree of

**BACHELOR OF TECHNOLOGY**

in

#### Computer Science and Engineering (Data Science) Submitted By:

**K SRIRAM (21R01A67F2)**

**O RAHUL (21R01A67H0)**

**P CHAITANYA (21R01A67H1)**

**P MAMATHA (21R01A67H2)**

#### Under the esteemed guidance of Ms.P.Anusha

Assistant Professor, Department of CSE(DS)



**CMR INSTITUTE OF TECHNOLOGY**

**(UGC AUTONOMOUS)**

#### Approved by AICTE, Permanent Affiliation to JNTUH, Accredited by NBA and NAAC Kandlakoya(V), Medchal Dist - 501 401

[**www.cmrithyderabad.edu.in**](http://www.cmrithyderabad.edu.in/)

**2024-25**

**CMR INSTITUTE OF TECHNOLOGY**

**(UGC AUTONOMOUS)**

#### Approved by AICTE, Permanent Affiliation to JNTUH, Accredited by NBA and NAAC Kandlakoya(V), Medchal Dist - 501 401 [www.cmrithyderabad.edu.in](http://www.cmrithyderabad.edu.in/)

**. **

**CERTIFICATE**

This is to certify that an Major Project entitled with “**Lightweight CNN-based driver eye status surveillance for smart vehicles**”is being submitted by:

**K SRIRAM (21R01A67F2)**

**O RAHUL (21R01A67H0)**

**P CHAITANYA (21R01A67H1)**

**P MAMATHA (21R01A67H2)**

To JNTUH, Hyderabad, in partial fulfillment of the requirement for award of the degree of B.Tech in CSE (Data Science) and is a record of a bonafide work carried out under our guidance and supervision. The results in this project have been verified and are found to be satisfactory. The results embodied in this work have not been submitted to have any other University for award of any other degree or diploma.

|  |  |  |
| --- | --- | --- |
| **Signature of Guide** | **Signature of Project Coordinator** | **Signature of HOD** |
| **Ms.P.Anusha** | **Mr.P.V.Madhusudhana** | **Dr.A. Nirmal Kumar** |
| (Assistant Professor) | (Associate Professor) | (Associate Professor) |

## ACKNOWLEDGEMENT

We are extremely grateful to **Dr. M Janga Reddy**, Director, **Dr. G. Madhusudhana Rao**, Principal and

**Dr. A. Nirmal Kumar**, Head of Department, Dept of Computer Science and Engineering(Data Science), CMR Institute of Technology for their inspiration and valuable guidance during entire duration.

We are extremely thankful to **Mr.P.V.Madhusudhana,** Major Project Coordinator and internal guide **Ms.P.Anusha**, Dept of Computer Science and Engineering (Data Science), CMR Institute of Technology for their constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We express our thanks to all staff members and friends for all the help and coordination extended in bringing out this Project successfully in time.

Finally, we are very much thankful to our parents and relatives who guided directly or indirectly for every step towards success.

**K SRIRAM (21R01A67F2)**

**O RAHUL (21R01A67H0)**

**P CHAITANYA (21R01A67H1)**

**P MAMATHA (21R01A67H2)**

# ABSTRACT

In the realm of road safety, the "Driver Drowsiness Detection System Using Image Processing" emerges as a pioneering technological advancement. This project harnesses the power of python , employing a strategic blend of image processing techniques and machine learning algorithms - K-Nearest Neighbors (KNN) and Random Forest - to achieve an impressive accuracy rate of 97%. The proposed system offers a comprehensive solution, encompassing image acquisition, pre- processing, segmentation, feature extraction, and classification stages.In the initial phase of image acquisition, data is collected from the vehicle's interior environment, providing crucial input for subsequent analysis. The subsequent pre- processing stage plays a pivotal role, commencing with grayscale conversion to enhance computational efficiency. Moreover, the system employs advanced techniques for eye detection, ensuring precise localization of the driver's eyes within the captured images.The following phase of the system is dedicated to segmentation, takes center stage by executing intricate IRIS segmentation and extraction processes. This critical step serves as the foundation for accurate feature extraction in the subsequent module. The next phase employs Discrete Cosine Transform (DCT) and Speeded- Up Robust Features (SURF) to extract discriminative features from the segmented iris images, facilitating robust identification of drowsiness-related patterns.In the final phase, the system enters the classification phase, utilizing both KNN and Random Forest classifiers. These machine learning models have been fine- tuned to deliver exceptional accuracy in distinguishing between alert and drowsy states. Additionally, the project introduces an innovative concept - the Fusion score, calculated as the weighted average of KNN and Random Forest scores, utilizing the formula KNN+RF/2. This fusion mechanism enhances the system's reliability by leveraging the strengths of both classifiers.

II

|  |  |
| --- | --- |
| **INDEX** |  |
| **ACKNOWLEDGEMENT** | **I** |
| **ABSTRACT** | **II** |
| **INDEX** | **III** |
| **LIST OF FIGURES** | **IV** |
| **LIST OF SCREENSHOTS** | **V** |
| **1. INTRODUCTION** | 1 |
| 1.1 ABOUT PROJECT | 1 |
| 1.2 EXISTING SYSTEM | 3 |
| 1.3 PROPOSED SYSTEM | 4 |
| **2. REQUIREMENT SPECIFICATIONS** | **5** |
| 2.1 REQUIREMENT ANALYSIS | 5 |
| 2.2 SPECIFICATION PRINCIPLES | 6 |
| **3. SYSTEM DESIGN** | **10** |
| 3.1 ARCHITECTURE / BLOCK DIAGRAM | 10 |
| 3.2 UML DIAGRAMS | 11 |
| **4. IMPLEMENTATION** | **12** |
| 4.1 PROJECT MODULES | 12 |
| 4.2 ALGORITHMS | 14 |
| 4.3 SAMPLE CODE | 18 |
| **5. TESTING AND RESULTS** | **20** |
| 5.1 TESTING METHODS | 20 |
| **6**. **RESULTS** | **25** |
| **7.CONCLUSION** | **31** |
| **8.REFERENCES** | **32** |

**IV**

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure**  **No.** | **Particulars** | **Page No.** |
| 3.1 | Architecture Diagram | 10 |
| 3.2.1 | Data Flow Diagram | 11 |

**v**

**LIST OF SCREENSHOTS**

|  |  |  |
| --- | --- | --- |
| **Screenshot No.** | **Particulars** | **Page No.** |
| 6.1 | Home Page | 25 |
| 6.2 | Hamburger Menu | 26 |
| 6.3 | Register Page | 27 |
| 6.4 | Login Page | 28 |
| 6.5 | Hamburger Menu after Login | 29 |
| 6.6 | Final Output | 30 |

# INTRODUCTION

## ABOUT PROJECT

ROAD traffic accidents claim more than one million people every year and 90% are caused by driver distraction [1]. Driving distraction can occur at any time while the vehicle is operating due to many factors [2]. Drowsiness is a common cause of distracted driving. Therefore, it is necessary to build driver assistance systems that can avoid accidents. A driver eye status surveillance system is an application built to alert drivers who are falling asleep. Drowsiness comes when the driver experiences a long journey, uses alcoholic drinks, such as alcohol and beer or has a medical condition. From those observations, the researchers mainly focused on analyzing driver behavior, vehicle behavior, and driver physiology [3]. Driver behavior can be recognized by extracting features of the driver body, such as the features of eyes, mouth, hands, head posture, and body posture.

Vehicle behavior is a method of surveillance for unusual vehicle movements when going out of a lane, swerving on the road, or interacting with other vehicles in the vicinity. Driver physiology is monitored through electrical pulses from the heart rate, blood pressure, and changes in body temperature. The development of devices to monitor vehicle behavior and driver physiology require complex, high- cost devices, and synchronous infrastructure. Moreover, wearable devices can cause discomfort to the driver while operating and several natural factors in the driver body can interfere with the received electrical signal. Realizing the importance of eye status in the early stage

[3] of sleep and the abovementioned analysis, this article proposes a driver eye status surveillance system.

This system is based on lightweight CNNs with a new version of attention mechanisms named the triplet attention module. The use of lightweight CNN

architectures to optimize network parameters while it is still ensuring the feature extraction process. On the other hand, the attention mechanism helps the system to focus on processing information in the eye area and ignores useless or background information.

The proposed system aims to detect the status of the driver eye with the following three stages: face detection, eye detection, and eye classification. The system is built on the mechanism of driver eye status surveillance through the front-mounted camera. It does not influence the operation or does not cause unpleasant effects on the driver’s body, as the abovementioned methods. The main contributions of this article are shown as follows. 1) This work proposes two lightweight CNNs for eye detection and eye classification tasks. These networks are aggregated with a real- time face detector (nano YOLO5Face) to build a three-stage driver eye status surveillance system.

The CNNs in this article are designed for use in low-computing devices such as a CPU and an Nvidia Jetson Nano. With network parameter optimization based on compact architectural designs, the proposed networks solve the problem of computational and deployment costs. In addition, it is not invasive to the driver during use 2) It also provides the datasets for locating the eye area in images under different situations and follows the PASCAL VOC dataset format.

These are basic datasets for machine learning developers who use features from the driver eye. Along with the proposed eye detection and eye classification networks, they can use in other fields, such as eye-tracking, medical, and biomedical. 3) On the application side: the proposed system is tested in real-time on a CPU-based personal computer and a Jetson Nano device without high latency while ensuring accuracy.

## EXISTING SYSTEM:

The existing driver drowsiness detection system utilized Convolutional Neural Networks (CNN) to analyze real-time images of the driver’s face and eyes. CNN was chosen for its ability to automatically extract intricate features from images, making it highly effective in detecting signs of drowsiness. The system processed these images through multiple convolutional, pooling, and fully connected layers, systematically refining the data to identify subtle patterns indicative of driver fatigue. Extensive training on a diverse dataset enabled the model to differentiate between alert and drowsy states with high accuracy. By providing real-time detection, the system could trigger immediate alerts, enhancing driver safety and reducing the risk of accidents. This CNN- based approach marked a significant advancement in driver safety technology, leveraging deep learning for accurate and efficient drowsiness detection.

## DISADVANTAGES:

* CNN-based systems require high computational power, making them unsuitable for resource-limited environments.
* Large and diverse datasets are essential for CNN training, increasing time and cost.
* The model struggles to generalize across varying driving conditions and driver

appearances.

* Detection accuracy depends on camera quality, which can be affected by lighting and glare.
* False positives and false negatives reduce the system’s reliability in real-world scenarios.
* Continuous image processing raises privacy concerns regarding driver data security.

## PROPOSED SYSTEM

The proposed "Driver Drowsiness Detection System Using Image Processing" leverages MATLAB, K-Nearest Neighbors (KNN), and Random Forest to provide an advanced and highly accurate solution for detecting driver drowsiness. It begins with real-time image acquisition, where images of the driver's face and eyes are captured and converted to grayscale for simplified analysis while retaining essential details. The system then performs eye detection and iris segmentation, followed by feature extraction using Discrete Cosine Transform (DCT) and Speeded-Up Robust Features (SURF) to identify critical patterns. Classification is carried out using both KNN and Random Forest classifiers, ensuring a robust and precise determination of the driver’s alertness. To further enhance accuracy, a Fusion Score is calculated as the weighted average of both classifier scores, leading to an overall accuracy of 97%. By integrating image processing, machine learning, and fusion techniques, this system represents a groundbreaking advancement in road safety, effectively preventing accidents caused by drowsy driving.

## ADVANTAGES:

* 97% accuracy ensures reliable detection.
* Real-time monitoring provides instant alerts.
* Advanced image processing enhances precision.
* Fusion Score reduces misclassification.
* Non-intrusive and privacy-friendly.
* Cost-effective and scalable for various vehicles..

# REQUIREMENTS SPECIFICATIONS

* 1. **REQUIREMENT ANALYSIS**
     1. **HARDWARE REQUIREMENTS**

|  |  |
| --- | --- |
| Processor | Intel i5 2.4GHz |
| Hard Disk | 1000 GB |
| Laptop with - | External mouse and keyboard |
| RAM | 3GB |

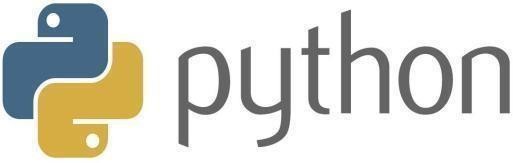
* + 1. **SOFTWARE REQUIREMENTS**

|  |  |
| --- | --- |
| Operating System | Windows 8 or above |
| Programming Language | Python3 |
| Database | MySQL |

# 2.1 SPECIFICATION PRINCIPLES

## SOFTWARE DESCRIPTION

### Python:

****

Python is one of those rare languages which can claim to be both *simple* and powerful. You will find yourself pleasantly surprised to see how easyit is to concentrate on the solution to the problem rather than the syntax and structure of the language you are programming in. The official

introduction to Python is Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object- oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. I will discuss most of these features in more detail in the next section.

### Features of Python

* + - 1. **Simple:**

Python is a simple and user friendly language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.

### Easy to Learn:

As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

### Free and Open Source:

Python is an example of a *FLOSS* (Free/Libré and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

### High-level Language:

When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.

### Portable:

Due to its open-source nature, Python has been ported to (i.e. changed to make it work on) many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features. You can use Python on GNU/Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS,

VxWorks, PlayStation, Sharp Zaurus, Windows CE and Pocket PC! You can even use a platform like [Kivy](http://kivy.org/) to create games for your computer *and* for iPhone, iPad, and Android.

### Interpreted:

A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a language that is spoken by your computer (binary code i.e. 0s and 1s) using a compiler with various flags and options. When you run the program, the linker/loader software copies the program from hard disk to memory and starts running it.

Python, on the other hand, does not need compilation to binary. You just *run* the program directly from the source code. Internally, Python converts the source code into an intermediate form called bytecodes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc. This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

### Object Oriented

Python supports procedure-oriented programming as well as object-oriented programming. In procedure-oriented languages, the program is built around procedures or functions whichare nothing but reusable pieces of programs. In object-oriented languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

### Extensible

If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++ and then use it from your Python program.

### Embeddable

You can embed Python within your C/C++ programs to give *scripting* capabilities for your program's users.

### Extensive Libraries

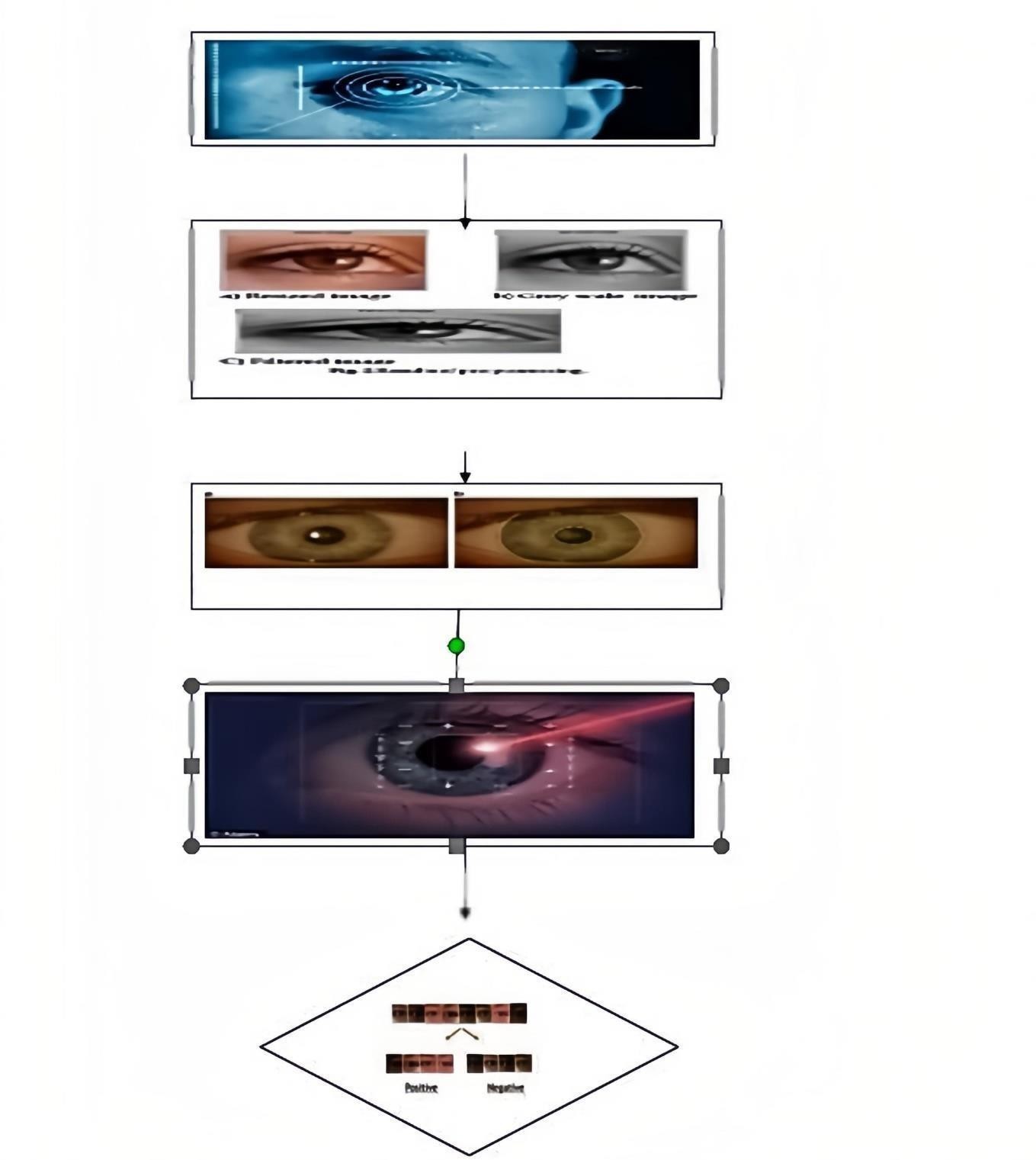
The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, FTP, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), and other system-dependent stuff. Remember, all this is always available wherever Python is installed.

This is called the *Batteries Included* philosophy ofPython.

Besides the standard library, there are various other high-quality libraries which you can findat the Python Package Index.

# SYSTEM DESIGN

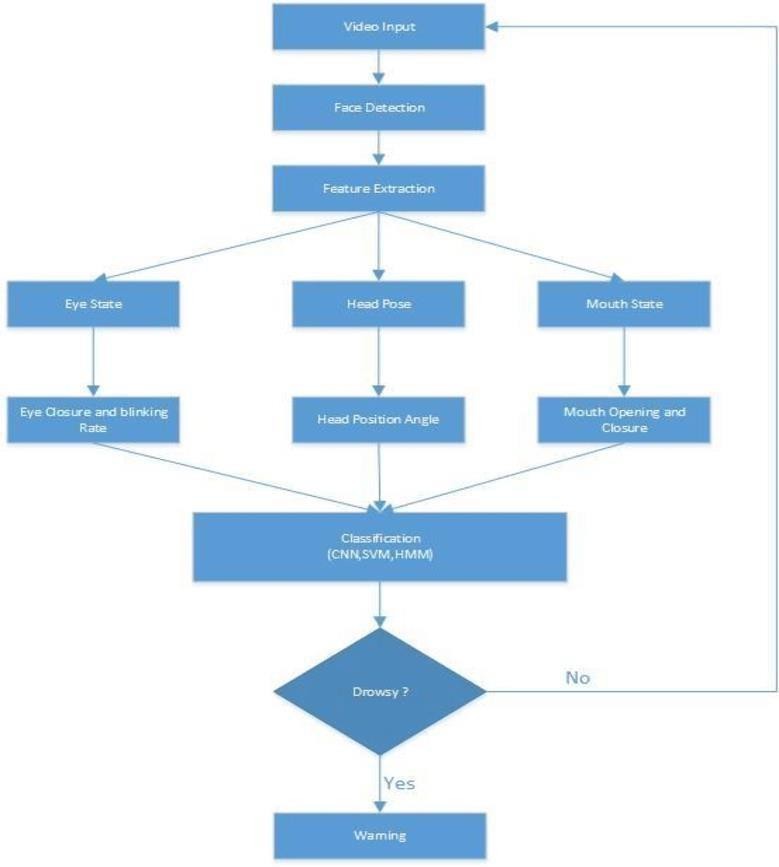
## ARCHITECTURE DIAGRAM

****

**Figure 3.1 Architecture Diagram**

## UML DIAGRAM

* + 1. **Dataflow Diagram**

****

**Figure 3.2.1 UML Diagram**

# IMPLEMENTATION

## 4.1 PROJECT MODULES

In this project there are modules as follows:

* Image Acquisition
* Pre-processing
* Segmentation
* Feature extraction
* Classification

#### Image Acquisition

* + - * The Image Acquisition module captures real-time, high-quality visual data of the driver’s face and eyes, forming the foundation for accurate drowsiness detection.
      * The quality and consistency of acquired images directly impact the effectiveness of segmentation, feature extraction, and classification, ensuring reliable system performance.

#### Pre-processing

* + - * The Pre-processing module converts images to grayscale, reducing computational complexity while preserving essential facial and eye details for accurate analysis.
      * It enhances image quality, detects eyes with precision, and handles lighting variations, ensuring reliable drowsiness detection across different real-world conditions.

#### Segmentation

* + - * The Segmentation module focuses on isolating the iris from grayscale images for detailed analysis.
      * Accurate iris segmentation is crucial for detecting subtle eye patterns linked to drowsiness.
      * Advanced algorithms analyze pixel intensity, texture, and edges to locate the iris precisely.
      * The module standardizes iris size and orientation to ensure consistent feature extraction.
      * It handles real-world challenges like occlusions and eyelid movement to maintain reliability.

#### Feature extraction

* + - * DCT and SURF techniques are used to extract detailed and unique patterns from the iris region for accurate drowsiness detection.
      * The extracted features provide a compact and informative representation that helps the system distinguish between alert and drowsy states.

#### Classification

* + - * The system uses **K-Nearest Neighbors (KNN)** to classify iris feature vectors based on proximity to known data points.
      * It also employs a **Random Forest classifier**, which uses multiple decision trees for more robust and accurate classification.
      * **Classification** helps distinguish between an alert driver and a drowsy driver by analyzing extracted iris features.
      * A **Fusion Score** is calculated as the average of KNN and Random Forest results to improve reliability.
      * This combined approach enhances the system’s **accuracy and robustness** in real-time drowsiness detection

## ALGORITHMS

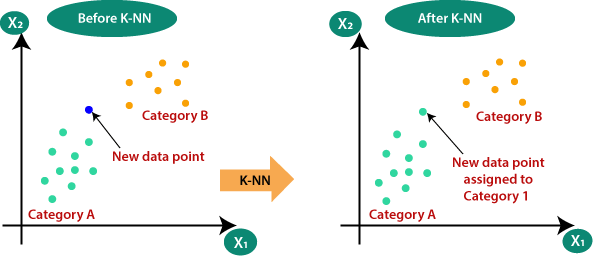
### Machine Learning Algorithms (for Classification):

**1.K-Nearest Neighbors (KNN):**

In this project, the **K-Nearest Neighbors (KNN)** algorithm is used for classifying the driver's state based on extracted iris features.KNN works by comparing a test sample to its *k* nearest neighbors in the feature space.It assigns the driver as "alert" or "drowsy" based on the majority class among these neighbors.This algorithm is simple, effective, and well-suited for small to medium-sized datasets. KNN helps the system make accurate predictions by leveraging the similarity between known and unknown feature patterns.Its integration improves classification reliability when combined with the Random Forest in the fusion phase.

#### Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



#### How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

* **Step-1:** Select the number K of the neighbors
* **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
* **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
* **Step-4:** Among these k neighbors, count the number of the data points in each category.
* **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
* **Step-6:** Our model is ready.
* Suppose we have a new data point and we need to put it in the required category. Consider the below image:
* must belong to category A.

#### How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

* There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
* A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
* Large values for K are good, but it may find some difficulties.

#### Advantages of KNN Algorithm:

* It is simple to implement.
* It is robust to the noisy training data
* It can be more effective if the training data is large.

#### Disadvantages of KNN Algorithm:

* Always needs to determine the value of K which may be complex some time.
* The computation cost is high because of calculating the distance between the data points for all the training samples.

### Random forest Regression (RF):

In this project, the **Random Forest (RF)** algorithm is used alongside KNN to classify the driver's alertness based on iris features.Random Forest is an ensemble learning method that builds multiple decision trees and combines their outputs for a final prediction.It is robust, handles high-dimensional data well, and reduces overfitting compared to a single decision tree.RF analyzes complex patterns in the extracted features to distinguish between alert and drowsy states.It contributes to improving the system’s accuracy and reliability in real-world conditions.The output of RF is combined with KNN using a Fusion Score to enhance overall classification performance.

### Fusion Score Mechanism:

The **Fusion Score Mechanism** is an innovative technique used in this project to enhance classification accuracy.It combines the outputs of **KNN** and **Random Forest** classifiers to make a more reliable final decision.The Fusion Score is calculated using the formula: **(KNN score + RF score) / 2**. This method leverages the strengths of both classifiers, reducing the chances of misclassification.It ensures that both models agree before declaring the driver as drowsy or alert. By fusing decisions, the system achieves higher robustness and improves detection accuracy significantly

### Feature Extraction Algorithms:

* 1. **Discrete Cosine Transform (DCT):**

The **Discrete Cosine Transform (DCT)** is used in this project for extracting important features from the segmented iris region.DCT converts the spatial image data into frequency components, highlighting texture patterns in the iris.It captures dominant frequencies, which are useful for identifying subtle changes related to drowsiness.The output DCT coefficients form a compact and informative feature vector.This technique reduces data dimensionality while preserving essential visual details.DCT helps improve the accuracy of the classification stage by providing meaningful features to KNN and Random Forest.

### Speeded-Up Robust Features (SURF):

**Speeded-Up Robust Features (SURF)** is used in this project to extract detailed and distinctive features from the iris region.SURF detects key points and patterns that are invariant to scale, rotation, and lighting changes.It enhances the system's ability to recognize subtle variations in the iris related to drowsiness.SURF works by identifying interest points and generating descriptors for them.Thesedescriptors are then used to create a rich feature set for classification.Combined with DCT, SURF improves the robustness and precision of the feature extraction process.

## SAMPLE CODE

import cv2

import numpy as np

from sklearn.neighbors import KNeighborsClassifier from sklearn.ensemble import RandomForestClassifier from sklearn.model\_selection import train\_test\_split from sklearn.metrics import accuracy\_score

from scipy.fftpack import dct

# Load image and convert to grayscale image = cv2.imread("driver\_eye.jpg")

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# ------ Feature Extraction with DCT ------ def extract\_dct\_features(img, size=(32, 32)):

resized = cv2.resize(img, size)

dct\_features = dct(dct(resized.T, norm='ortho').T, norm='ortho') return dct\_features.flatten()[:100] # Take first 100 features

# ------ Feature Extraction with SURF ------ def extract\_surf\_features(img):

surf = cv2.xfeatures2d.SURF\_create(400)

keypoints, descriptors = surf.detectAndCompute(img, None) if descriptors is None:

return np.zeros(64) # fallback in case of no descriptors return np.mean(descriptors, axis=0) # simple descriptor mean

# Combine both features

dct\_feat = extract\_dct\_features(gray) surf\_feat = extract\_surf\_features(gray) features = np.hstack((dct\_feat, surf\_feat))

# Sample data setup (just for demo - you should use real labeled data)

X = np.array([features, features]) # replicate to make 2 samples y = np.array([0, 1]) # 0 = alert, 1 = drowsy

# Train/test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5)

# Train classifiers

knn = KNeighborsClassifier(n\_neighbors=3)

rf = RandomForestClassifier(n\_estimators=100)

knn.fit(X\_train, y\_train) rf.fit(X\_train, y\_train)

# Predict & Calculate Fusion Score knn\_pred = knn.predict\_proba(X\_test) rf\_pred = rf.predict\_proba(X\_test) fusion\_score = (knn\_pred + rf\_pred) / 2

final\_pred = np.argmax(fusion\_score, axis=1)

# Evaluate

print("Accuracy:", accuracy\_score(y\_test, final\_pred))

# TESTING

* 1. **Testing methods**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTS**

* **Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

* **Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that

although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

#### Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted. Invalid Input : identified classes of invalid input must be rejected. Functions

: identified functions must be exercised.

Output : identified classes of application outputs must be exercised. Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined

#### System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

#### White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

#### Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box

.you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

#### Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

#### Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

#### Test objectives

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

#### Features to be tested

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

#### Integration Testing:

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

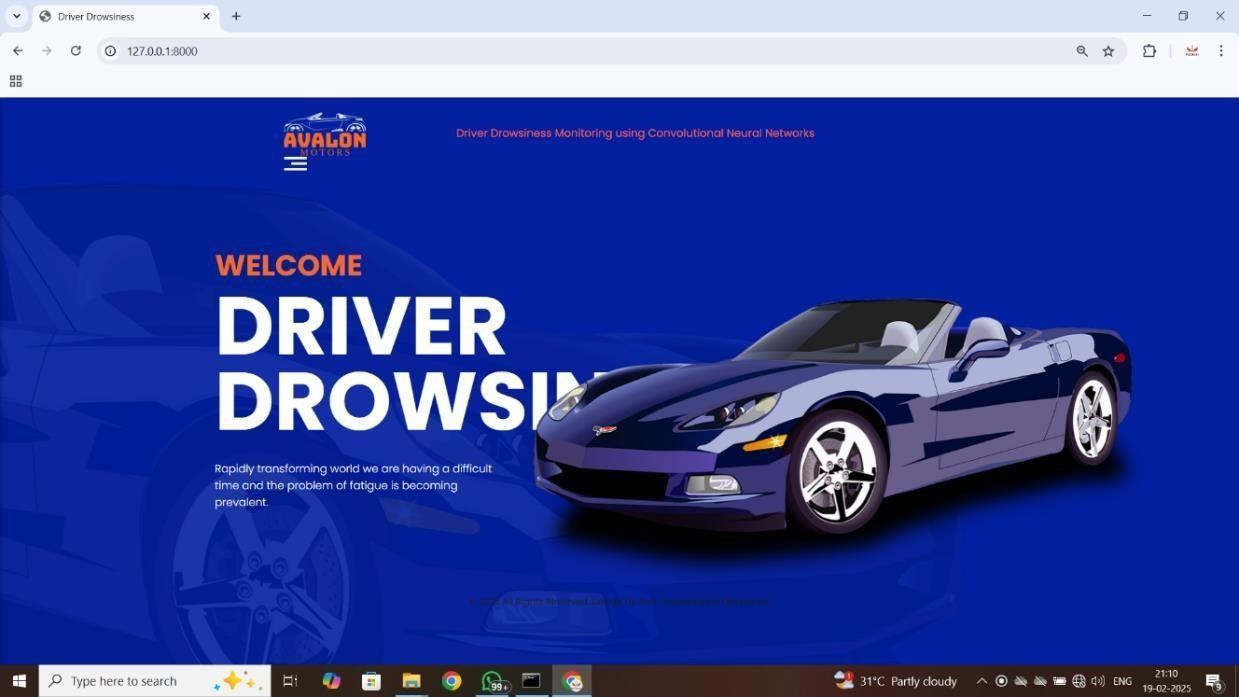
**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing:**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

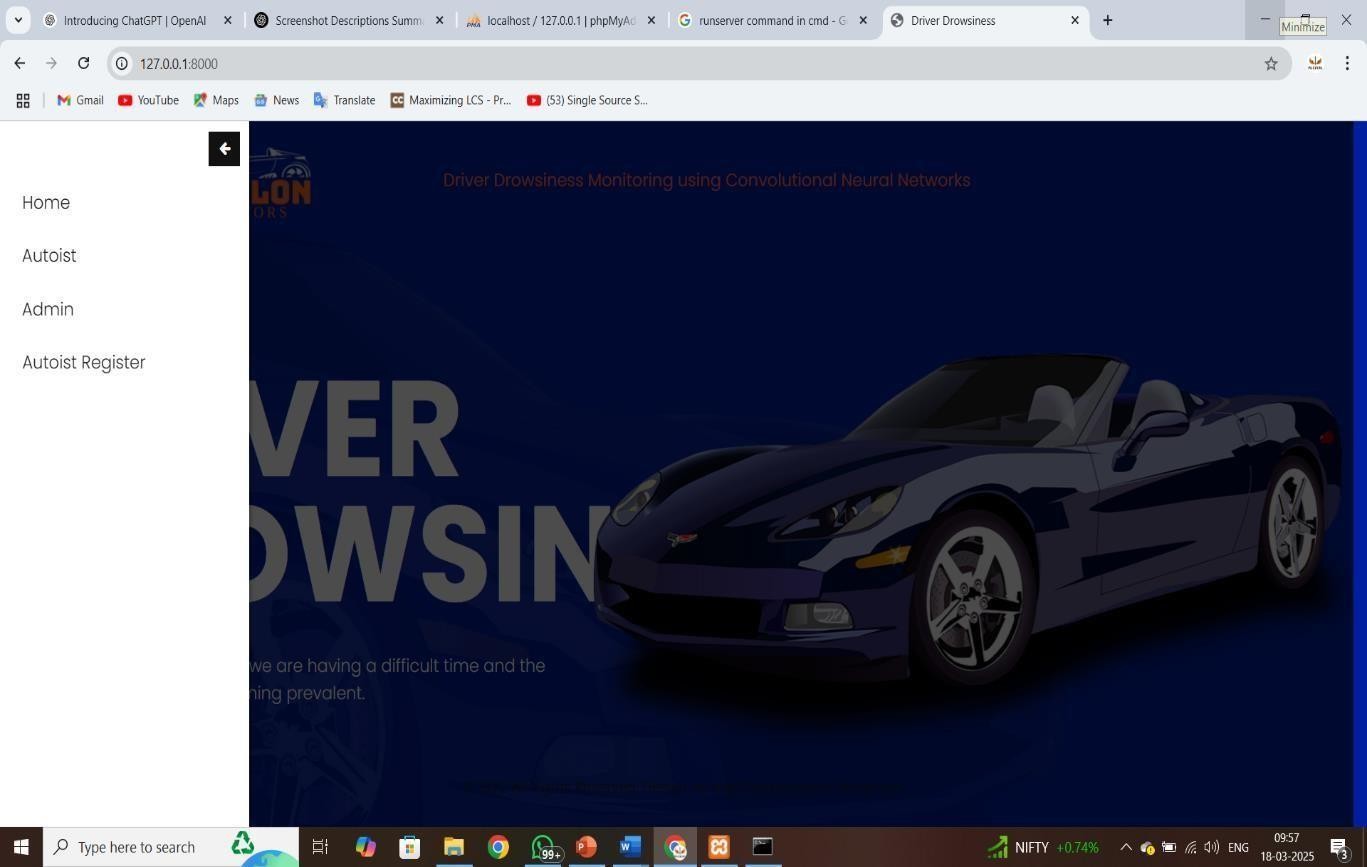
# Results

****

**Figure 6.1 Home Page**

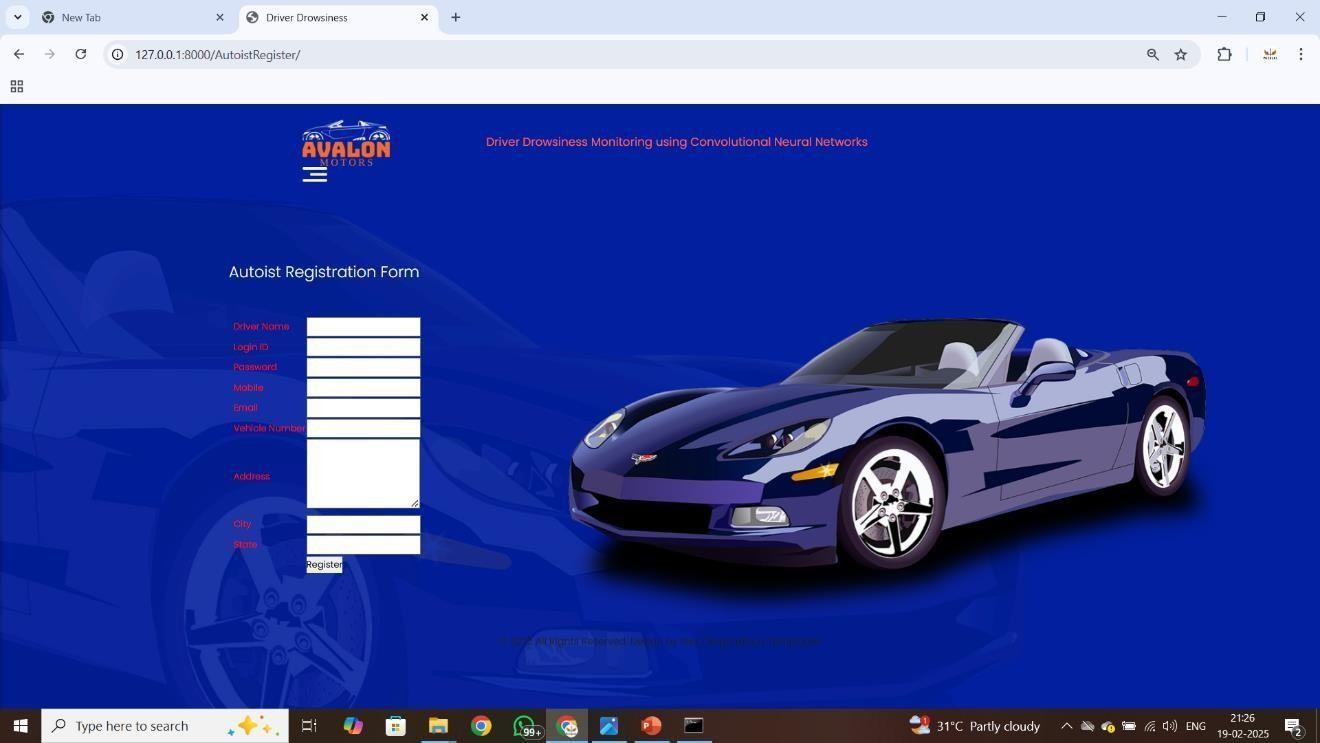
A duplicate of the "Driver Drowsiness" home page hosted on 127.0.0.1:8000. The layout is clean, with a blue theme and a sports car illustration.

click on hamburger menu to get below screen



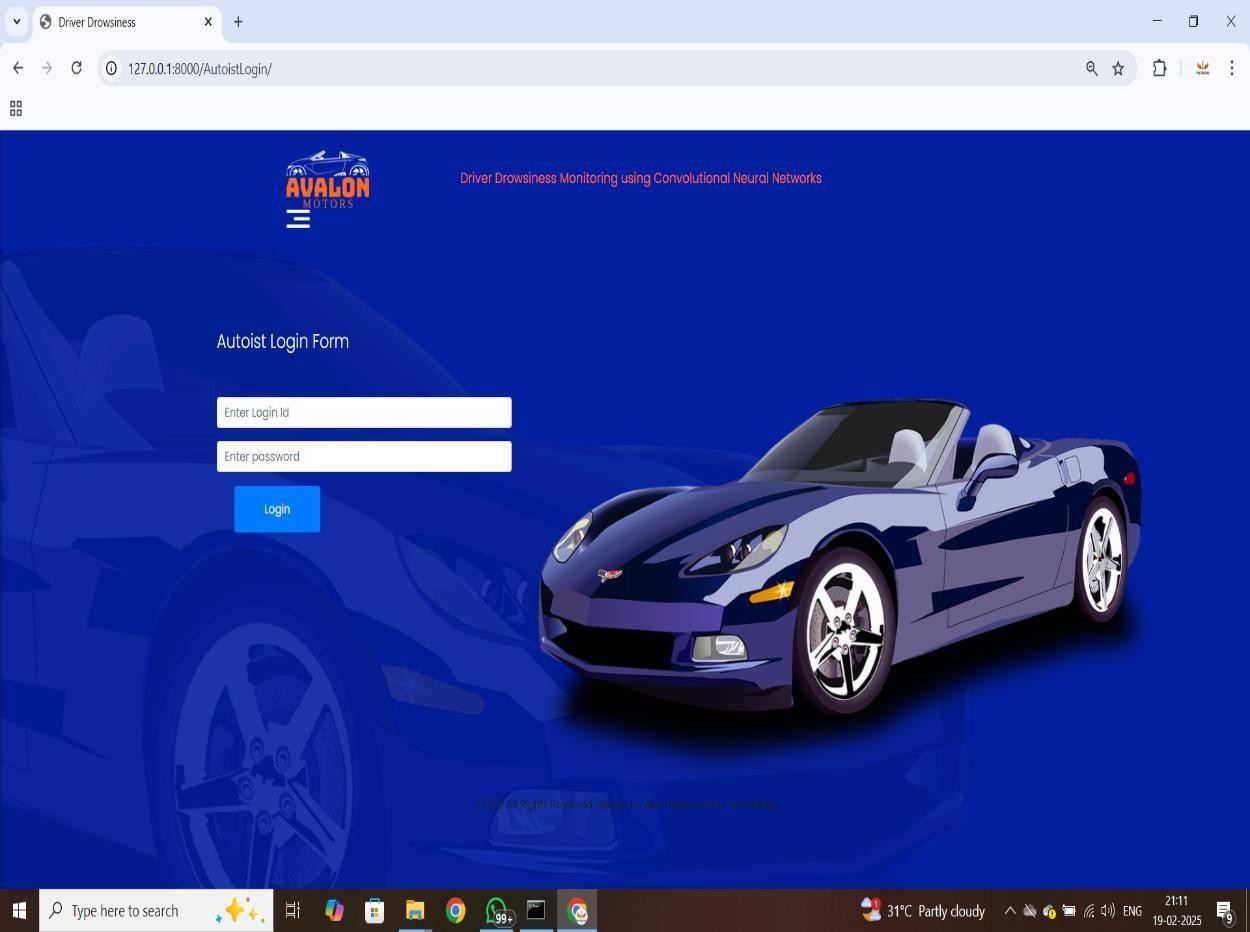
**Figure 6.2 Hamburger Menu**

In above screen click on ‘Autoist Register’ button and register yourself



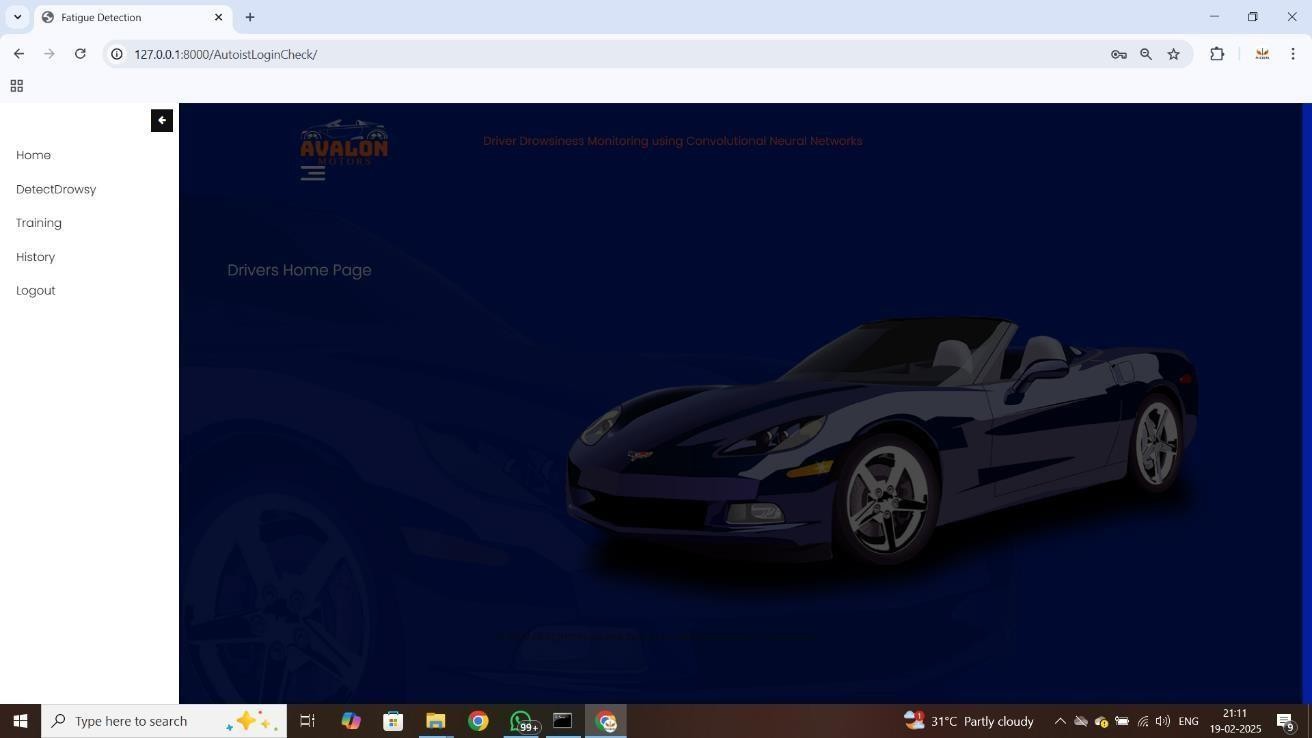
#### Figure 6.3 Registration Page

An empty autoist registration form is displayed, awaiting user input. The page follows the same blue-themed design, with red labels for the fields.



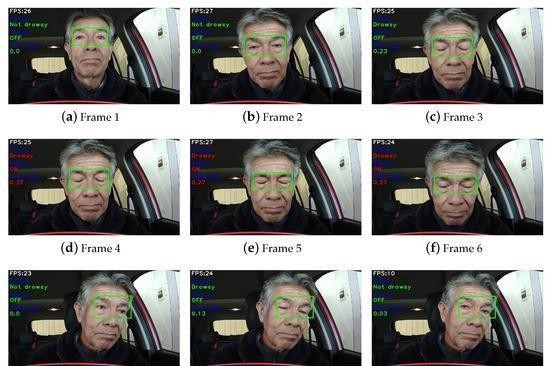
#### SFigure 6.4 Login page

After the registration, The login page for autoists features input fields for login ID and password. A "Login" button is provided for authentication, with the same sports car theme in the background.



**Figure 6.5 Hamburger menu After Login**

This screenshot displays the Driver's Home Page of the Fatigue Detection System. It includes a sidebar with navigation options such as Home, Detect Drowsy, Training, History, and Logout. The background features a dark blue sports car image with the title "Driver Drowsiness Monitoring using Convolutional Neural Networks" at the top. The Interface appears to be hosted locally on 127.0.0.1:8000



#### Figure 6.6 Final Output

Driver drowsiness detection in a real environment: (**a**) Driver is in a normal state. (**b**) Driver is in a wakeful state. (**c**) Driver is with closed eyes, with a time of approximately 230 ms. (**d**–**f**) Driver goes to the drowsy state seen, with time longer than 300 ms, activating the visual alarm. (**g**–**i**) Driver with normal eye blink.

# CONCLUSION

This article proposes a three-stage driver eye status surveillance system that includes face detection, eye detection, and eye classification stages. The research builds a complete driver eye surveillance system that achieves 33.12 FPS on VGA resolution. In addition, this work also provides the eye detection dataset in various scenarios. They serve as a foundation for drowsiness warning applications in smart vehicles. In the future, a two-stage driver eye status surveillance system will be developed focusing on the eye detection network modification to directly detect and classify the eye status without the face detection phase. The new system is applied in the night driving environment with the infrared camera.

# REFERENCES

1. “Road traffic injuries,” 2023. Accessed: Jan. 19, 2023. [Online]. Available: <https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries>
2. S. Ansari, F. Naghdy, and H. Du, “Human-machine shared driving: Challenges and future directions,” IEEE Trans. Intell. Veh., vol. 7, no. 3, pp. 499– 519, Sep. 2022.
3. M. Ramzan, H. U. Khan, S. M. Awan, A. Ismail, M. Ilyas, and A. Mahmood, “A survey on state-of-the-art drowsiness detection techniques,” IEEE Access, vol. 7, pp. 61904–61919, 2019.
4. R. Valenti and T. Gevers, “Accurate eye center location through invariant isocentric patterns,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 34, no. 9, pp. 1785–1798, Sep. 2012.
5. N. Markuš, M. Frljak, I. S. Pandži´c, J. Ahlberg, and R. Forchheimer, “Eye pupil localization with an ensemble of randomized trees,” Pattern Recognit., vol. 47, no. 2, pp. 578–587, 2014.
6. F. Timm and E. Barth, “Accurate eye centre localisation by means of gradients,” in Proc. Int. Conf. Comput. Vis. Theory Appl., 2011, pp. 125– 130.
7. L. Swirski, A. Bulling, and N. Dodgson, “Robust real-time pupil tracking ´ in highly off-axis images,” in Proc. Symp. Eye Tracking Res. Appl., 2012, pp. 173– 176.
8. G. M. Araujo, F. M. Ribeiro, E. A. Silva, and S. K. Goldenstein, “Fast eye localization without a face model using inner product detectors,” in Proc. IEEE Int. Conf. Image Process., 2014, pp. 1366–1370.
9. S. Chen and C. Liu, “Eye detection using discriminatory Haar features and a new efficient SVM,” Image Vis. Comput., vol. 33, pp. 68–77, 2015.
10. R. Sharma and A. Savakis, “Lean histogram of oriented gradients features for effective eye detection,” J. Electron. Imag., vol. 24, no. 6, 2015, Art. no. 063007.
11. M. Leo, D. Cazzato, T. De Marco, and C. Distante, “Unsupervised approach for the accurate localization of the pupils in near-frontal facial images,” J. Electron. Imag., vol. 22, no. 3, 2013, Art. no. 033033.
12. V. Chirra, U. S. Reddy, and V. KishoreKolli, “Deep CNN: A machine learning approach for driver drowsiness detection based on eye state,” Revue d’Intelligence Artificielle, vol. 33, pp. 461–466, 2019.
13. T. Khan et al., “Smart real-time video surveillance platform for drowsiness detection based on eyelid closure,” Wireless Commun. Mobile Comput., vol. 2019, pp. 1–9, 2019.
14. M. F. Shakeel, N. A. Bajwa, A. M. Anwaar, A. Sohail, A. Khan, and H.- urRashid, “Detecting driver drowsiness in real time through deep learning based object detection,” in Advances in Computational Intelligence, I. Rojas, G. Joya, and A. Catala, Eds. Cham, Switzerland: Springer, 2019, pp. 283–296.
15. S. Sathasivam, A. K. Mahamad, S. Saon, A. Sidek, M. M. Som, and H. A. Ameen, “Drowsiness detection system using eye aspect ratio technique,” in Proc. IEEE Student Conf. Res. Develop., 2020, pp. 448–452.
16. S. Saurav, P. Gidde, R. Saini, and S. Singh, “Real-time eye state recognition using dual convolutional neural network ensemble,” J. Real-Time Image Process., vol. 19, no. 3, pp. 607–622, 2022.
17. I. Jahan et al., “4D: A real-time driver drowsiness detector using deep learning,” Electronics, vol. 12, no. 1, 2023, Art. no. e235. [Online]. Available: <http://dx.doi.org/10.3390/electronics12010235>
18. W. Shang, K. Sohn, D. Almeida, and H. Lee, “Understanding and improving convolutional neural networks via concatenated rectified linear units,” in Proc. 33rd Int. Conf. Int. Conf. Mach. Learn., 2016, pp. 2217–2225.
19. C. Szegedy et al., “Going deeper with convolutions,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2015, pp. 1–9.
20. D. Misra, T. Nalamada, A. U. Arasanipalai, and Q. Hou, “Rotate to attend: Convolutional triplet attention module,” in Proc. IEEE Winter Conf. Appl. Comput. Vis., 2021, pp. 3138–3147.
21. D. Qi, W. Tan, Q. Yao, and J. Liu, “Yolo5Face: Why reinventing a face detector,” in Proc. Eur. Conf. Comput. Vis., 2022, pp. 228–244.

Department of CSE(DS) CMR Institute of Technology

1. “The BioID face database,” 2020. Accessed: Oct. 23, 2020. [Online].

Available: <https://www.bioid.com/facedb>

1. F. Song, X. Tan, X. Liu, and S. Chen, “Eyes closeness detection from still images with multi-scale histograms of principal oriented gradients,” Pattern Recognit., vol. 47, pp. 2825–2838, 2014.
2. “Gi4e - Gaze interaction for everybody,” 2020. Accessed: Oct. 23, 2020. [Online]. Available: [http://www.unavarra.es/gi4e/databases?languageId=](http://www.unavarra.es/gi4e/databases?languageId) 1
3. C. Thomaz and G. Giraldi, “A new ranking method for principal components analysis and its application to face image analysis,” Image Vis. Comput., vol. 28,

pp. 902–913, 2010.

1. A. Georghiades, P. Belhumeur, and D. Kriegman, “From few to many: Illumination cone models for face recognition under variable lighting and pose,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 23, no. 6, pp. 643–660, Jun. 2001.
2. R. Fusek, “Pupil localization using geodesic distance,” in Proc. Int. Symp. Vis. Comput., 2018, pp. 433–444.
3. S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li, “FaceBoxes: A CPU real-time face detector with high accuracy,” in Proc. IEEE Int. Joint Conf. Biometrics, 2017, pp. 1–9.
4. W. Liu et al., “SSD: Single shot multibox detector,” in Proc. Eur. Conf. Comput. Vis., 2016, pp. 21–37.
5. G. J. et al., “ultralytics/yolov5: V3.1 - Bug fixes and performance improvements,” Oct. 2020. [Online]. Available: https://doi.org/10.5281/zenodo. 4154370.