

**A**  
**FIELD BASED PROJECT REPORT**  
**on**  
**DRIVER DROWSINESS DETECTION SYSTEM USING**  
**OPENCV AND KERAS**

**BACHELOR OF TECHNOLOGY**  
**in**  
**INFORMATION TECHNOLOGY**

**Submitted by**  
**(BATCH : 20)**  
**G.TANUJA [227Y1A12C3]**  
**M.N.S.VARSHA[227Y1A1292]**

**Under the Guidance**  
**of**

**Mr. CH.V.V.N.RAJU**  
**Associative Professor**



**DEPARTMENT OF INFORMATION TECHNOLOGY**  
**MARRI LAXMAN REDDY INSTITUTE OF TECHNOLOGY AND**  
**MANAGEMENT**  
**(AUTONOMOUS)**  
**JUNE 2024**



**MARRI LAXMAN REDDY**  
**INSTITUTE OF TECHNOLOGY AND MANAGEMENT**

(AN AUTONOMOUS INSTITUTION)

(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section 2(f) & 12(B) of the UGC act, 1956

## **CERTIFICATE**

---

This is to certify that the field based project report titled “**Driver Drowsiness Detection System using opencv and keras**” is being submitted by **G. Tanuja(227Y1A12C3)** and **M.N.S. Varsha(227Y1A1292)** in **II B.Tech II Semester Information Technology** is a record bonafide work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree.

**Internal Guide**

**(Mr.CH.V.V.N.Raju)**

**HOD**

**(Dr.M.NagaLakshmi)**

**Principal**

**(Dr.R.Murali Prasad)**



# **MARRI LAXMAN REDDY** **INSTITUTE OF TECHNOLOGY AND MANAGEMENT**

(AN AUTONOMOUS INSTITUTION)

(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section 2(f) & 12(B) of the UGC act, 1956

## **DECLARATION**

---

We hereby declare that the Field Based Project Report entitled, “**Driver Drowsiness Detection System using opencv and keras**” submitted for the B.Tech degree is entirely our work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree.

**Date:**

**G.Tanuja**

**(227Y1A12C3)**

**M.N.S.Varsha**

**(227Y1A1292)**



## **MARRI LAXMAN REDDY INSTITUTE OF TECHNOLOGY AND MANAGEMENT**

**(AN AUTONOMOUS INSTITUTION)**

(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section 2(f) & 12(B) of the UGC act, 1956

### **ACKNOWLEDGEMENT**

---

We are happy to express our deep sense of gratitude to the principal of the college **Dr.R.Murali Prasad**, Professor, Marri Laxman Reddy Institute of Technology & Management, for having provided us with adequate facilities to pursue our project.

We would like to thank **Dr.M.Nagalakshmi**, Professor and Head, Department of Information Technology, Marri Laxman Reddy Institute of Technology & Management, for having provided the freedom to use all the facilities available in the department, especially the laboratories and the library.

We are very grateful to our project guide Mr.CH.V.V.N.Raju, Assoc. Prof., Department of Information Technology, Marri Laxman Reddy Institute of Technology & Management, for his extensive patience and guidance throughout our project work.

We sincerely thank our seniors and all the teaching and non-teaching staff of the Department of Information Technology for their timely suggestions, healthy criticism and motivation during the course of this work.

We would also like to thank our classmates for always being there whenever we needed help or moral support. With great respect and obedience, We thank our parents and sister who were the backbone behind our deeds.

Finally, We express our immense gratitude with pleasure to the other individuals who have either directly or indirectly contributed to our need at right time for the development and success of this work.



# **MARRI LAXMAN REDDY** **INSTITUTE OF TECHNOLOGY AND MANAGEMENT**

(AN AUTONOMOUS INSTITUTION)

(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

Accredited by NBA and NAAC with 'A' Grade & Recognized Under Section 2(f) & 12(B) of the UGC act, 1956

## **CONTENTS**

<b>S.NO</b>	<b>TITLE</b>	<b>PAGE NO</b>
	<b>ABSTRACT</b>	<b>8</b>
	<b>LIST OF FIGURES</b>	<b>9</b>
	<b>LIST OF TABLES</b>	<b>10</b>
	<b>SYMBOLS AND ABBREVIATIONS</b>	<b>11</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>12</b>
<b>1.1</b>	Motivation	12
<b>1.2</b>	Problem	12
<b>1.3</b>	Solution	12
<b>1.4</b>	Scope	12
<b>1.5</b>	Problem Definition	12
<b>1.6</b>	Objective	13
<b>1.7</b>	Limitations	13
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>14</b>
<b>2.1</b>	Deep Neural Network for Face Recognition System	14
<b>2.2</b>	Behaviour based Data Dispatcher	14
<b>2.3</b>	Driver Fatigue Detection Based Eye Tracking	15

<b>2.4</b>	Driver Drowsiness Detection System based on feature representation learning using various Deep Network	15
<b>3</b>	<b>SYSTEM ANALYSIS</b>	<b>16</b>
<b>3.1</b>	Existing System	16
<b>3.2</b>	Disadvantages of Existing System	16
<b>3.3</b>	Proposed System	16
<b>3.4</b>	Advantages of Proposed System	17
<b>3.5</b>	System Study	17
<b>4</b>	<b>REQUIREMENT ANALYSIS</b>	<b>19</b>
<b>4.1</b>	Requirement Specification	19
<b>5</b>	<b>SYSTEM DESIGN</b>	<b>20</b>
<b>5.1</b>	System Architecture	20
<b>5.2</b>	UML Diagrams	21
<b>6</b>	<b>IMPLEMENTATION</b>	<b>25</b>
<b>6.1</b>	Modules	25
<b>6.2</b>	System Specification	26
<b>7</b>	<b>SOFTWARE ENVIRONMENT</b>	<b>27</b>
<b>7.1</b>	What is Python	27
<b>7.2</b>	What is Django	42
<b>8</b>	<b>TESTING AND VALIDATION</b>	<b>53</b>
<b>8.1</b>	System Test	53

<b>8.2</b>	Test Cases	<b>57</b>
<b>9</b>	<b>CONCLUSION AND FURTHER ENHANCEMENT</b>	<b>64</b>
<b>10</b>	<b>REFERENCES</b>	<b>65</b>

## ABSTRACT

---

The advancement in computer vision has assisted drivers in the form of automatic self-driving cars etc. The misadventure are caused by driver's fatigue and drowsiness about 20%. It poses a serious problem for which several approaches were proposed. However, they are not suitable for real-time processing. The major challenges faced by these methods are robustness to handle variation in human face and lightning conditions. We aim to implement an intelligent processing system that can reduce road accidents drastically. This approach enables us to identify driver's face characteristics like eye closure percentage, eye-mouth aspect ratios, blink rate, yawning, head movement, etc. In this system, the driver is continuously monitored by using a webcam. The driver's face and the eye are detected using haar cascade classifiers. Eye images are extracted and fed to Custom designed Convolutional Neural Network for classifying whether both left and right eye are closed. Based on the classification, the eye closure score is calculated. If the driver is found to be drowsy, an alarm will be triggered.



## LIST OF FIGURES

---

FIG. NO	FIG. NAME	PAGE NO.
1	System Architecture	20
2	Data Flow Diagram	21
3	UML Diagrams	21
4	Use Case Diagrams	22
5	Class Diagram	22
6	Sequence Diagram	23
7	Activity Diagram	24

## LIST OF TABLES

---

<b>TABLEN.</b>	<b>TABLE TITLE</b>	<b>PAGENO.</b>
1	Requirement Specification	26
	Hardware Requirements	26
	Software Requirements	26

## **SYMBOLS & ABBREVIATIONS**

---

<b>IDE</b>	:	Integrated Development Environment
<b>IOT</b>	:	Internet Of Things
<b>I/O</b>	:	Input and Output
<b>LED</b>	:	Light Emitting Diode
<b>SMS</b>	:	Simple Message Service
<b>UML</b>	:	Unified Modeling Language
<b>URL</b>	:	Uniform Resource Locator
<b>USB</b>	:	Universal Serial Bus

# **INTRODUCTION**

Many safety connected driving supporter schemes decreased the danger of four-wheeler accidents, and investigations depicted weariness to be a major reason of four-wheeler accidents. A car organization announced an idea that whole deadly accidents (17%) would be attributed to weary drivers. Many revisions showed by Volkswagen AG specify that 5-25% of all accidents are produced by the sleeping of driver. The lack of concentration damage steering actions and decrease response period, and revisions illustrated that sleepiness raises threat of crashes demand for a dependable intelligent driver sleepiness sensing system. The aim is to create an intelligent processing scheme to avoid road accidents. This can be done by period of time monitoring the drowsiness and warning driver of inattention to prevent accidents. Based on the literature survey, the driver's drowsiness can be detected based on three factors such as physiological, behavioral, and vehicle-based measurements. But these approaches pose some disadvantages in certain real-time scenarios.

## **1.1 Motivation**

Driver drowsiness is a leading cause of road accidents. An automated detection system can alert drivers before they fall asleep, enhancing road safety.

## **1.2 Problem**

Traditional drowsiness detection methods are unreliable in real-time. There is a need for an automated system that can detect drowsiness promptly and accurately.

## **1.3 Solution**

A driver drowsiness detection system using OpenCV and Keras can analyze facial features and eye movements to detect drowsiness. It will alert the driver when drowsiness is detected, preventing potential accidents.

## **1.4 Scope**

- Develop a real-time video processing application using OpenCV.
- Train a neural network with Keras to recognize drowsiness from facial landmarks.
- Integrate the detection system with an alert mechanism.

## **1.5 Problem Definition**

The project aims to detect driver drowsiness in real-time by recognizing signs like frequent blinking and prolonged eye closure and alerting the driver promptly.

## **1.6 Objective**

Develop a reliable driver drowsiness detection system that:

- Detects drowsiness in real-time.
- Minimizes false positives and negatives.
- Is user-friendly and easily integrable into vehicles.

## **1.7 Limitations**

- Performance may vary with lighting conditions and driver movements.
- Less effective for drivers wearing glasses.
- High computational requirements.
- Extensive and diverse training data needed for accuracy.

# **LITERATURE SURVEY**

## **1) Deep Neural Network for Human Face Recognition**

**AUTHORS:** [Priya Gupta](#) , [Nidhi Saxena](#)

Face recognition (FR), the process of identifying people through facial images, has numerous practical applications in the area of biometrics, information security, access control, law enforcement, smart cards and surveillance system. Convolutional Neural Networks (CovNets), a type of deep networks has been proved to be successful for FR. For real-time systems, some preprocessing steps like sampling needs to be done before using to CovNets. But then also complete images (all the pixel values) are passed as input to CovNets and all the steps (feature selection, feature extraction, training) are performed by the network. This is the reason that implementing CovNets are sometimes complex and time consuming. CovNets are at the nascent stage and the accuracies obtained are very high, so they have a long way to go. The paper proposes a new way of using a deep neural network (another type of deep network) for face recognition. In this approach, instead of providing raw pixel values as input, only the extracted facial features are provided. This lowers the complexity of while providing the accuracy of 97.05% on Yale faces dataset.

## **2) Behaviour Based Data Dispatcher**

**AUTHORS:** [Mohan Sai Singamsetti](#), [Mona Teja Kurakula](#)

Human life is a complex social structure. It is not possible for the humans to navigate without reading the other persons. They do it by identifying the faces. The state of response can be decided based on the mood of the opposite person. Whereas a person's mood can be figured out by observing his emotion (Facial Gesture). The aim of the project is to construct a "Facial emotion Recognition" model using DCNN (Deep convolutional neural network) in real time. The model is constructed using DCNN as it is proven that DCNN work with greater accuracy than CNN (convolutional neural network). The facial expression of humans is very dynamic in nature it changes in split seconds whether it may be Happy, Sad, Angry, Fear, Surprise, Disgust and Neutral etc. This project is to predict the emotion of the person in real time. Our brains have neural networks which are responsible for all kinds of thinking (decision making, understanding). This model tries to develop these decisions making and classification skills by training the machine. It can classify and predict the multiple faces and different emotions at the very same time. In order to obtain higher accuracy, we take the models which are trained over thousands of datasets.

### **3) Driver Fatigue Detection Based on Eye Tracking**

**AUTHORS : Mandalapu Sarada Devi**

The International statistics shows that a large number of road accidents are caused by driver fatigue. Therefore, a system that can detect oncoming driver fatigue and issue timely warning could help in preventing many accidents, and consequently save money and reduce personal suffering. The authors have made an attempt to design a system that uses video camera that points directly towards the driver's face in order to detect fatigue. If the fatigue is detected a warning signal is issued to alert the driver. The authors have worked on the video files recorded by the camera. Video file is converted into frames. Once the eyes are located from each frame, by measuring the distances between the intensity changes in the eye area one can determine whether the eyes are open or closed. If the eyes are found closed for 5 consecutive frames, the system draws the conclusion that the driver is falling asleep and issues a warning signal. The algorithm is proposed, implemented, tested, and found working satisfactorily.

### **4) Driver Drowsiness Detection System Based on Feature Representation Learning Using Various Deep Networks**

**AUTHORS : Sanghyuk Park, Fei Pan, Sunghun Kang, Chang D. Yoo**

Statistics have shown that 20% of all road accidents are fatigue-related, and drowsy detection is a car safety algorithm that can alert a snoozing driver in hopes of preventing an accident. This paper proposes a deep architecture referred to as deep drowsiness detection (DDD) network for learning effective features and detecting drowsiness given a RGB input video of a driver. The DDD network consists of three deep networks for attaining global robustness to background and environmental variations and learning local facial movements and head gestures important for reliable detection. The outputs of the three networks are integrated and fed to a softmax classifier for drowsiness detection. Experimental results show that DDD achieves 73.06% detection accuracy on NTHU-drowsy driver detection benchmark dataset.

## **SYSTEM ANALYSIS**

### **EXISTING SYSTEM:**

driving supporter schemes decreased the danger of four-wheeler accidents, and investigations depicted weariness to be a major reason of four wheeler accidents. A car organization announced an idea that whole deadly accidents (17%) would be attributed to weary drivers. Many revisions showed by Volkswagen AG specify that 5-25% of all accidents are produced by the sleeping of driver. The lack of concentration damage steering actions and decrease response period, and revisions illustrated that sleepiness raises threat of crashes demand for a dependable intelligent driver sleepiness sensing system. The aim is to create an intelligent processing scheme to avoid road accidents. This can be done by period of time monitoring the drowsiness and warning driver of inattention to prevent accidents.

### **DISADVANTAGES OF EXISTING SYSTEM:**

- It is not suitable for real-time processing.
- The existing system uses the orientation of facial characteristics for drowsy detection.
- based on three factors such as physiological, behavioral, and vehicle-based measurements. But these approaches pose some disadvantages in certain real time scenarios.

**Algorithm:** Learning Vector Quantization (LVQ), Support Vector Machines (SVM)

### **PROPOSED SYSTEM:**

Our proposed system will provide a solution for monitoring driver's drowsiness. The cons of the existing system in extracting only selected hand-crafted features is overcome by using custom-designed CNN by giving an input driver image. Now the driver will be continuously monitored by a webcam. The video captured is converted into a sequence of frames. For each frame, the face and eye are detected using predefined classifiers available in opencv called haar cascade classifiers. Eye images are extracted and sent to a series of 2D CNN layers (5x5, 3x3 kernel valid padding), max-pooling layers(2x2) and finally, the fully connected dense layer classifies whether



eyes are closed or not. A score is calculated based on eye closure. If both eyes are closed consecutively in 15 frames then the system predicts as drowsy and an alarm sound is triggered to alert the car operator. The categorization of driver drowsiness is done correctly and the normalization issues in the existing model are eliminated by using customdesigned CNN.

### **ADVANTAGES OF PROPOSED SYSTEM:**

- If the eyes are both closed, we increase the score and when eyes are open, we decrease the score. We are drafting the outcome to display the actual time condition of the driver.
- approach enables us to identify driver's face characteristics like eye closure percentage, eye-mouth aspect ratios, blink rate, yawning, head movement.

**Algorithm:** Convolutional Neural Network; Data Augmentation Deep Learning

## **SYSTEM STUDY**

### **FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

**Three key considerations involved in the feasibility analysis are,**

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

### **ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because

most of the technologies used are freely available. Only the customized products had to be purchased.

## **TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

## **SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

## **REQUIREMENT ANALYSIS**

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

### **REQUIREMENT SPECIFICATION**

#### **Functional Requirements**

- Graphical User interface with the User.

#### **Software Requirements**

For developing the application the following are the Software Requirements:

- Python
- Django

#### **Operating Systems supported**

- Windows 10 64 bit OS

#### **Technologies and Languages used to Develop**

- Python

#### **Debugger and Emulator**

- Any Browser (Particularly Chrome)

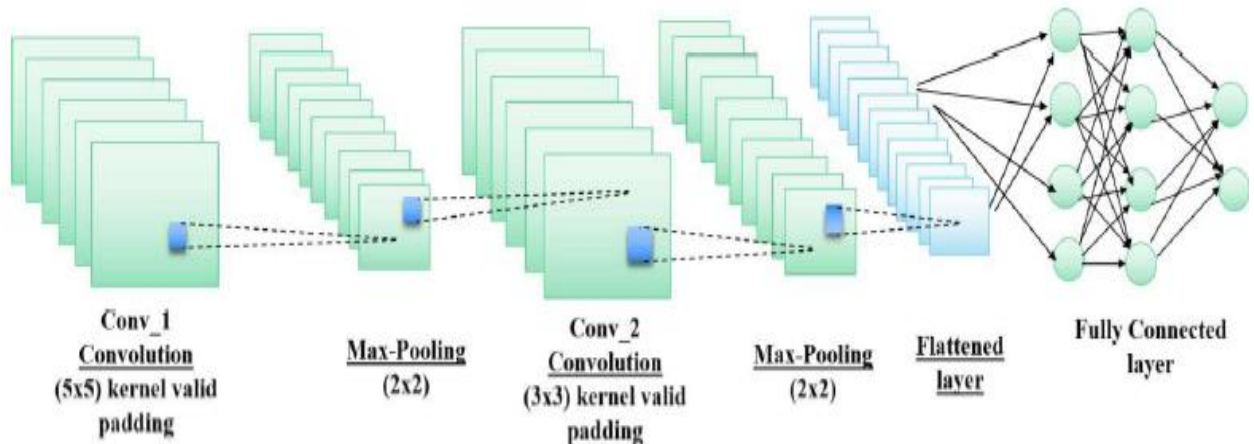
#### **Hardware Requirements**

For developing the application the following are the Hardware Requirements:

- Processor: Intel i9
- RAM: 32 GB
- Space on Hard Disk: minimum 1 TB

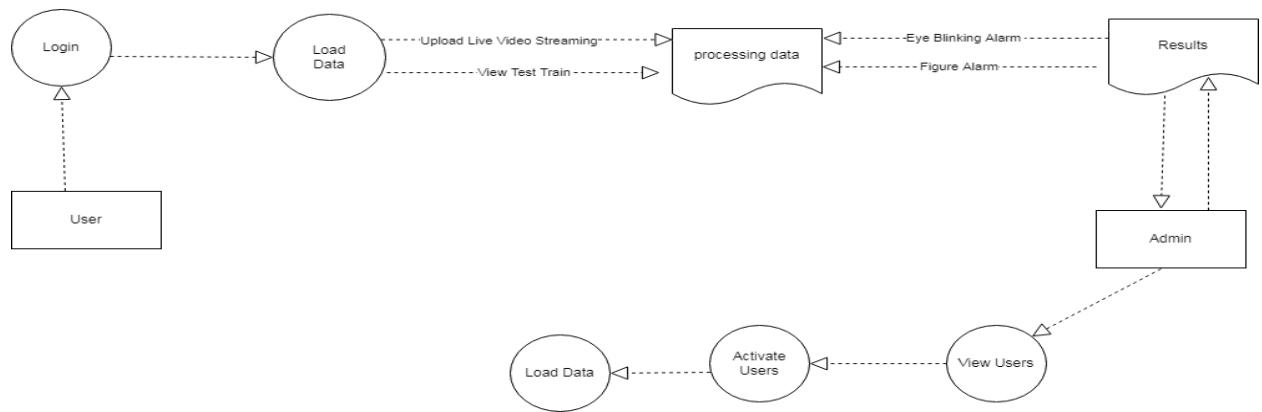
## **SYSTEM DESIGN**

### **SYSTEM ARCHITECTURE:**



### **DATA FLOW DIAGRAM:**

- The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
- The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
- DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
- DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



## **UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

## **GOALS:**

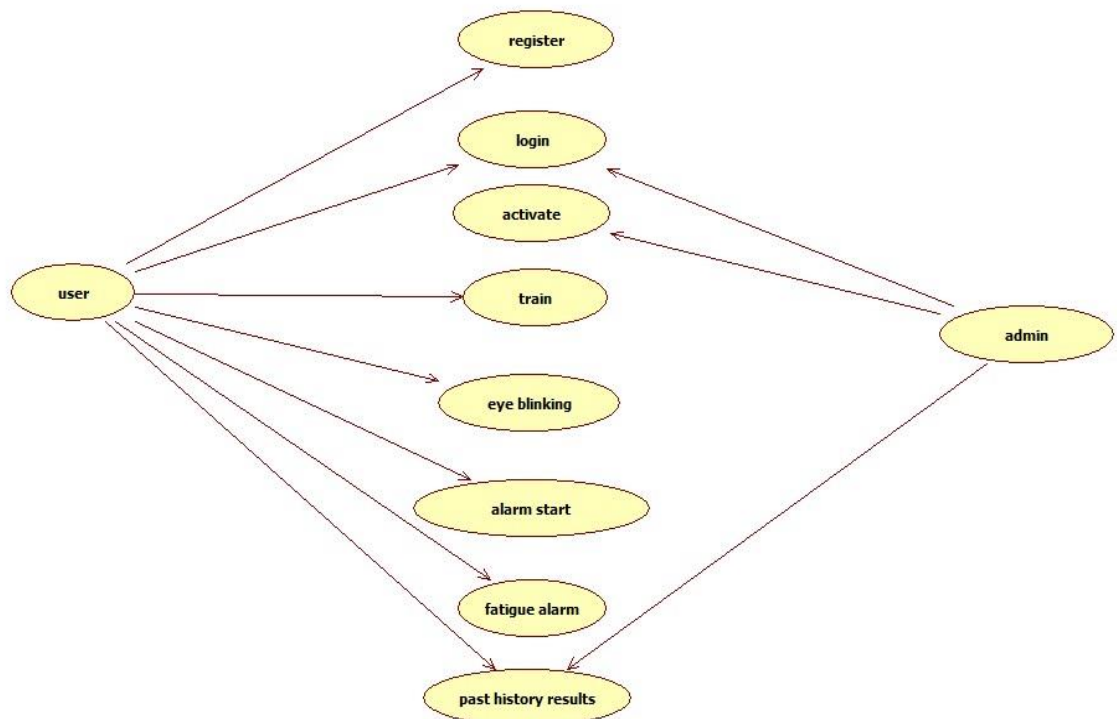
The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.

- Provide a formal basis for understanding the modeling language.
- Encourage the growth of OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

### **USE CASE DIAGRAM:**

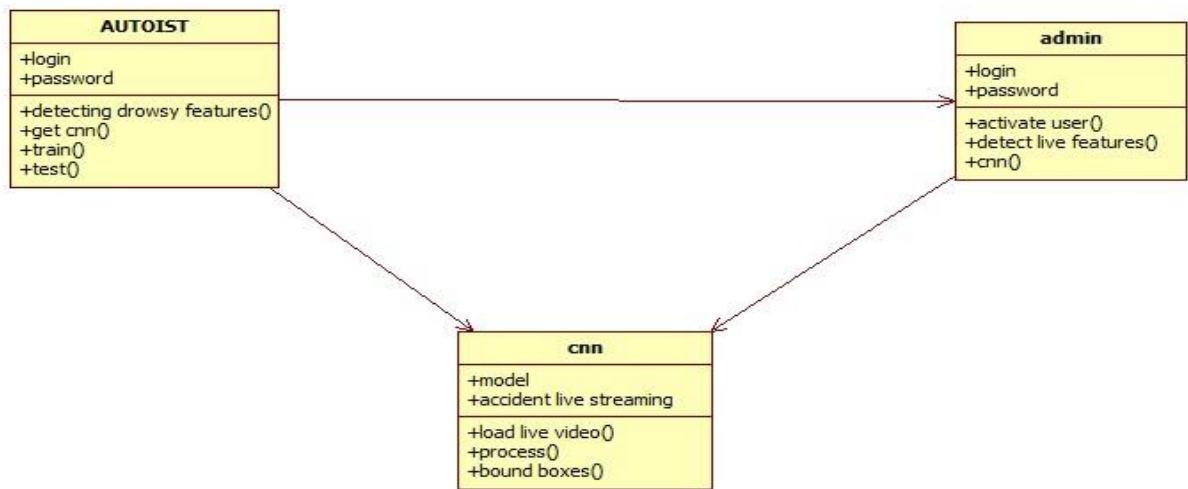
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



### **CLASS DIAGRAM:**

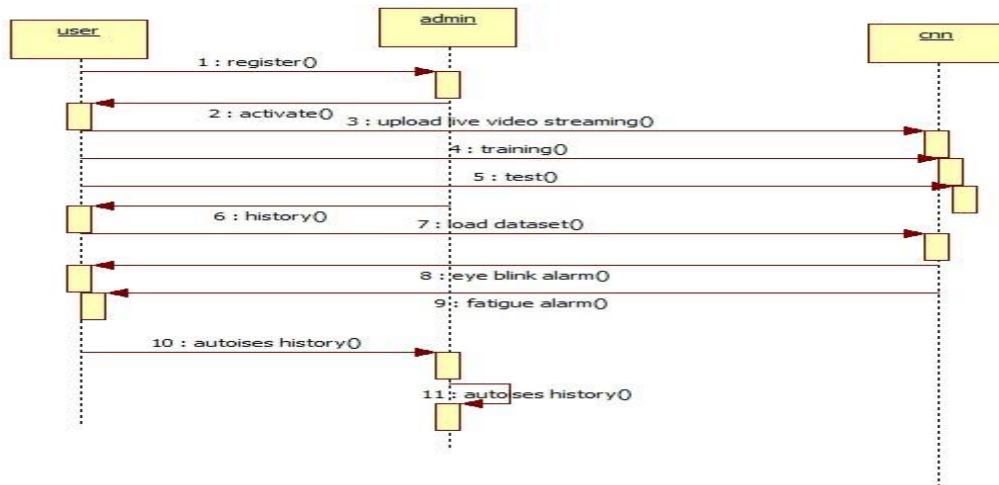
In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing

the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



## **SEQUENCE DIAGRAM:**

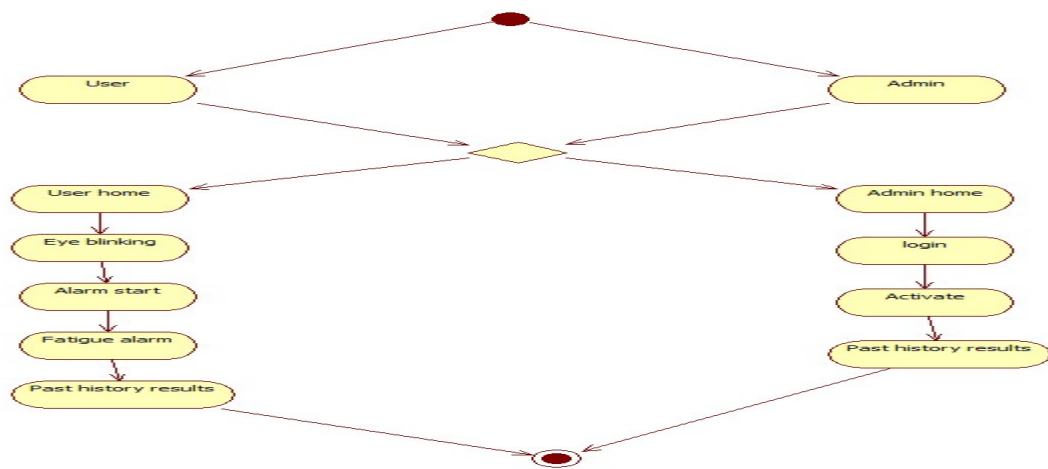
A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



## **ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-

by-step workflows of components in a system. An activity diagram shows the overall flow of control.





# **IMPLEMENTATION**

## **MODULES:**

- User
- Admin
- Data Preprocessing
- Machine Learning

## **MODULES DESCRIPTION:**

### **User:**

The User can register the first. While registering he required a valid user email and mobile for further communications. Once the user register then admin can activate the user. Once admin activated the user then user can login into our system. User can upload the dataset based on our dataset column matched. For algorithm execution data must be in float format. Here we took Driver Drowsiness dataset for testing purpose. User can also add the new data for existing dataset based on our Django application. User can click the Classification in the web page so that the data calculated Precision, Recall, Accuracy, Support and F1-Score based on the algorithms. User can click Prediction in the web page so that user can write the review after predict the review That will display results depends upon review like postive,negative or neutral

### **Admin:**

Admin can login with his login details. Admin can activate the registered users. Once he activate then only the user can login into our system. Admin can view the overall data in the browser. Admin can click the Results in the web page so calculated Precision, Recall, Support Accuracy and F1-Score based on the algorithms is displayed. All algorithms execution complete then admin can see the overall accuracy in web page.

### **Data Preprocessing:**

A dataset can be viewed as a collection of data objects, which are often also called as a records, points, vectors, patterns, events, cases, samples, observations, or entities. Data objects are described by a number of features that capture the basic characteristics of an object, such as the mass of a physical object or the time at which an event occurred, etc. Features are often called as variables, characteristics, fields, attributes, or

dimensions. The data preprocessing in this forecast uses techniques like removal of noise in the data, the expulsion of missing information, modifying default values if relevant and grouping of attributes for prediction at various levels.

### **Machine learning:**

Based on the split criterion, the cleansed data is split into 60% training and 40% test, then the dataset is subjected to Two machine learning classifiers such as Deep Learning, Convolutional Neural Networks . The accuracy Recall, Support, Precision and F1-Score of the classifiers was calculated and displayed in my results. The classifier which bags up the highest accuracy could be determined as the best classifier.

## **SYSTEM SPECIFICATION**

### **HARDWARE REQUIREMENTS**

- **System** : Intel i3
- **Hard Disk** : 1 TB.
- **Monitor** : 14' Colour Monitor.
- **Mouse** : Optical Mouse.
- **Ram** : 4GB.

### **SOFTWARE REQUIREMENTS**

- **Operating system** : Windows 10.
- **Coding Language** : Python.
- **Front-End** : Html. CSS
- **Designing** : Html,css,javascript.
- **Data Base** : SQLite.

# SOFTWARE ENVIRONMENT

## PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An [interpreted language](#), Python has a design philosophy that emphasizes code [readability](#) (notably using [whitespace](#) indentation to delimit [code blocks](#) rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer [lines of code](#) than might be used in languages such as [C++](#) or [Java](#). It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many [operating systems](#). [CPython](#), the [reference implementation](#) of Python, is [open source](#) software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit [Python Software Foundation](#). Python features a [dynamic type](#) system and automatic [memory management](#). It supports multiple [programming paradigms](#), including [object-oriented](#), [imperative](#), [functional](#) and [procedural](#), and has a large and comprehensive [standard library](#).

### Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt –

```
$ python
```

```
Python 2.4.3 (#1, Nov 11 2010, 13:34:43)
```

```
[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>>
```

Type the following text at the Python prompt and press the Enter –

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in `print ("Hello, Python!")`; However in Python version 2.4.3, this produces the following result –

Hello, Python!

## **Script Mode Programming**

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension `.py`. Type the following source code in a `test.py` file –

Live Demo

```
print "Hello, Python!"
```

We assume that you have Python interpreter set in `PATH` variable. Now, try to run this program as follows –

```
$ python test.py
```

This produces the following result –

Hello, Python!

Let us try another way to execute a Python script. Here is the modified `test.py` file –

Live Demo

```
#!/usr/bin/python
```

```
print "Hello, Python!"
```

We assume that you have Python interpreter available in `/usr/bin` directory. Now, try to run this program as follows –

```
$ chmod +x test.py      # This is to make file executable
```

```
$ ./test.py
```

This produces the following result –

Hello, Python!

## **Python Identifiers**

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (\_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, \$, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers –

Class names start with an uppercase letter. All other identifiers start with a lowercase letter.

Starting an identifier with a single leading underscore indicates that the identifier is private.

Starting an identifier with two leading underscores indicates a strongly private identifier.

If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

## **Reserved Words**

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and    exec    not

assert finally    or

break for    pass

class    from    print

continue    global    raise

def if    return

del import try

elif in while

else is with

except lambda yield

## **Lines and Indentation**

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

if True:

    print "True"

else:

    print "False"

However, the following block generates an error –

if True:

print "Answer"

print "True"

else:

print "Answer"

print "False"

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks –

Note – Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces.

```
#!/usr/bin/python

import sys

try:

    # open file stream

    file = open(file_name, "w")

except IOError:

    print "There was an error writing to", file_name

    sys.exit()

print "Enter '", file_finish,

print "' When finished"

while file_text != file_finish:

    file_text = raw_input("Enter text: ")

    if file_text == file_finish:

        # close the file

        file.close

        break

    file.write(file_text)

    file.write("\n")
```

```

file.close()

file_name = raw_input("Enter filename: ")

if len(file_name) == 0:

    print "Next time please enter something"

    sys.exit()

try:

    file = open(file_name, "r")

except IOError:

    print "There was an error reading file"

    sys.exit()

file_text = file.read()

file.close()

print file_text

```

### **Multi-Line Statements**

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example –

```

total = item_one + \

    item_two + \

    item_three

```

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example –



```
days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
```

### Quotation in Python

Python accepts single ('), double (") and triple ("" or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal –

```
word = 'word'
```

```
sentence = "This is a sentence."
```

```
paragraph = """This is a paragraph. It is made up of multiple lines and sentences."""
```

### Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

### Live Demo

```
#!/usr/bin/python
```

```
# First comment
```

```
print "Hello, Python!" # second comment
```

This produces the following result –

```
Hello, Python!
```

You can type a comment on the same line after a statement or expression –

```
name = "Madisetti" # This is again comment
```

You can comment multiple lines as follows –

```
# This is a comment.
```

```
# This is a comment, too.
```

```
# This is a comment, too.
```

```
# I said that already.
```

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comments:

```
This is a multiline comment.
```

### Using Blank Lines

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

In an interactive interpreter session, you must enter an empty physical line to terminate a multiline statement.

### Waiting for the User

The following line of the program displays the prompt, the statement saying “Press the enter key to exit”, and waits for the user to take action –

```
#!/usr/bin/python
```

```
raw_input("\n\nPress the enter key to exit.")
```

Here, "\n\n" is used to create two new lines before displaying the actual line. Once the user presses the key, the program ends. This is a nice trick to keep a console window open until the user is done with an application.

### Multiple Statements on a Single Line

The semicolon ( ; ) allows multiple statements on the single line given that neither statement starts a new code block. Here is a sample snip using the semicolon.

```
import sys; x = 'foo'; sys.stdout.write(x + '\n')
```

### Multiple Statement Groups as Suites

A group of individual statements, which make a single code block are called suites in Python. Compound or complex statements, such as if, while, def, and class require a header line and a suite.

Header lines begin the statement (with the keyword) and terminate with a colon ( : ) and are followed by one or more lines which make up the suite. For example –

```
if expression :
```

```
    suite
```

```
elif expression :
```

```
    suite
```

```
else :
```

```
    suite
```

## **Command Line Arguments**

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h –

```
$ python -h
```

```
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
```

Options and arguments (and corresponding environment variables):

-c cmd : program passed in as string (terminates option list)

-d : debug output from parser (also PYTHONDEBUG=x)

-E : ignore environment variables (such as PYTHONPATH)

-h: print this help message and exit

You can also program your script in such a way that it should accept various options. Command Line Arguments is an advanced topic and should be studied a bit later once you have gone through rest of the Python concepts.

## Python Lists

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example –

```
list1 = ['physics', 'chemistry', 1997, 2000];
```

```
list2 = [1, 2, 3, 4, 5 ];
```

```
list3 = ["a", "b", "c", "d"]
```

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example –

```
tup1 = ('physics', 'chemistry', 1997, 2000);
```

```
tup2 = (1, 2, 3, 4, 5 );
```

```
tup3 = "a", "b", "c", "d";
```

The empty tuple is written as two parentheses containing nothing –

```
tup1 = ();
```

To write a tuple containing a single value you have to include a comma, even though there is only one value –

```
tup1 = (50,);
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

### Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example –

#### Live Demo

```
#!/usr/bin/python

tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5, 6, 7 );

print "tup1[0]: ", tup1[0];

print "tup2[1:5]: ", tup2[1:5];
```

When the above code is executed, it produces the following result –

```
tup1[0]:  physics
tup2[1:5]:  [2, 3, 4, 5]
```

### Updating Tuples

#### Accessing Values in Dictionary

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example –

#### Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

print "dict['Name']: ", dict['Name']

print "dict['Age']: ", dict['Age']
```

When the above code is executed, it produces the following result –

```
dict['Name']:  Zara
```

```
dict['Age']:  7
```

If we attempt to access a data item with a key, which is not part of the dictionary, we get an error as follows –

Live Demo

```
#!/usr/bin/python
```

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
```

```
print "dict['Alice']: ", dict['Alice']
```

When the above code is executed, it produces the following result –

```
dict['Alice']:
```

Traceback (most recent call last):

```
File "test.py", line 4, in <module>
```

```
    print "dict['Alice']: ", dict['Alice'];
```

```
KeyError: 'Alice'
```

## Updating Dictionary

You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example –

Live Demo

```
#!/usr/bin/python
```

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
```

```
dict['Age'] = 8; # update existing entry
```

```
dict['School'] = "DPS School"; # Add new entry
```

```
print "dict['Age']: ", dict['Age']

print "dict['School']: ", dict['School']
```

When the above code is executed, it produces the following result –

```
dict['Age']: 8

dict['School']: DPS School
```

### Delete Dictionary Elements

You can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the del statement. Following is a simple example –

### Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

del dict['Name']; # remove entry with key 'Name'

dict.clear();      # remove all entries in dict

del dict ;         # delete entire dictionary

print "dict['Age']: ", dict['Age']

print "dict['School']: ", dict['School']
```

This produces the following result. Note that an exception is raised because after del dict dictionary does not exist any more –

```
dict['Age']:
```

Traceback (most recent call last):

File "test.py", line 8, in <module>

```
print "dict['Age']: ", dict['Age'];
```

TypeError: 'type' object is unsubscriptable

Note – del() method is discussed in subsequent section.

## Properties of Dictionary Keys

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys –

(a) More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins. For example –

Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'}

print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result –

```
dict['Name']: Manni
```

(b) Keys must be immutable. Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed. Following is a simple example –

Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7}

print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result –



Traceback (most recent call last):

File "test.py", line 3, in <module>

```
dict = {'Name': 'Zara', 'Age': 7};
```

TypeError: unhashable type: 'list'

Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates –

Live Demo

```
#!/usr/bin/python
```

```
tup1 = (12, 34.56);
```

```
tup2 = ('abc', 'xyz');
```

```
# Following action is not valid for tuples
```

```
# tup1[0] = 100;
```

```
# So let's create a new tuple as follows
```

```
tup3 = tup1 + tup2;
```

```
print tup3;
```

When the above code is executed, it produces the following result –

```
(12, 34.56, 'abc', 'xyz')
```

## Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the del statement. For example –

Live Demo

```
#!/usr/bin/python

tup = ('physics', 'chemistry', 1997, 2000);

print tup;

del tup;

print "After deleting tup : ";

print tup;
```

This produces the following result. Note an exception raised, this is because after del tup tuple does not exist any more –

```
('physics', 'chemistry', 1997, 2000)
```

After deleting tup :

Traceback (most recent call last):

File "test.py", line 9, in <module>

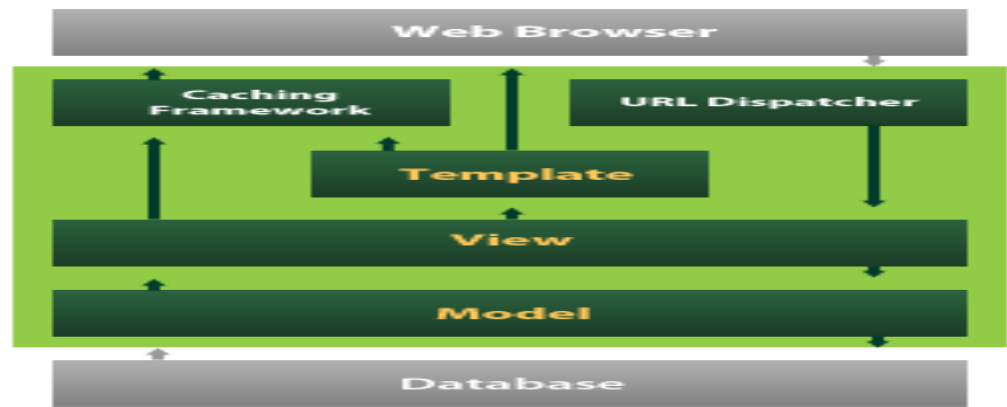
```
print tup;
```

NameError: name 'tup' is not defined

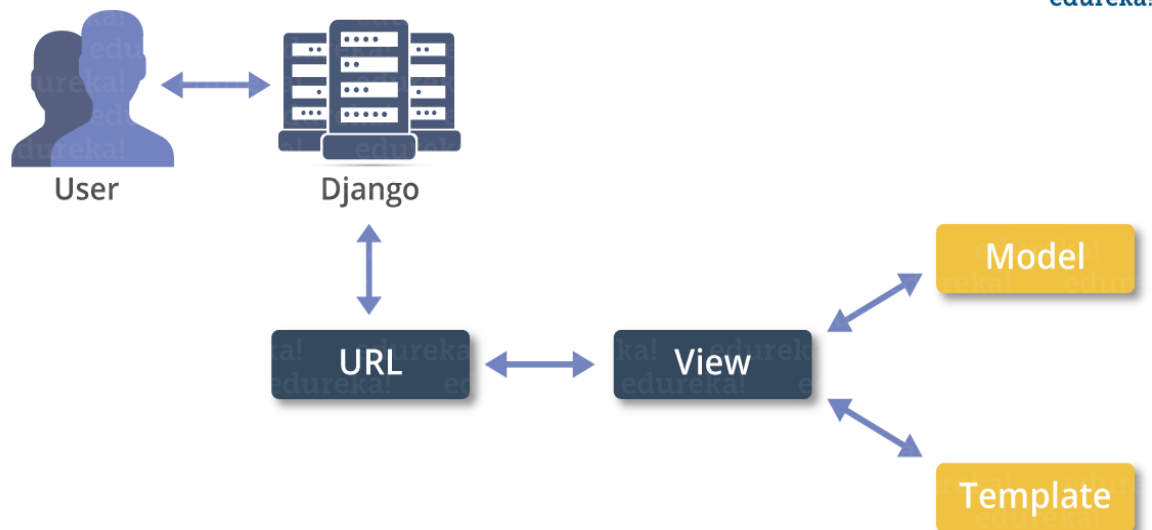
## **DJANGO**

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes [reusability](#) and "pluggability" of components, rapid development, and the principle of [don't repeat yourself](#). Python is used throughout, even for settings files and data models.



Django also provides an optional administrative [create, read, update and delete](#) interface that is generated dynamically through [introspection](#) and configured via admin models



## Create a Project

Whether you are on Windows or Linux, just get a terminal or a cmd prompt and navigate to the place you want your project to be created, then use this code –

```
$ django-admin startproject myproject
```

This will create a "myproject" folder with the following structure –

```
myproject/
```

```
    manage.py
```

myproject/

\_\_init\_\_.py

settings.py

urls.py

wsgi.py

## The Project Structure

The “myproject” folder is just your project container, it actually contains two elements –

manage.py – This file is kind of your project local django-admin for interacting with your project via command line (start the development server, sync db...). To get a full list of command accessible via manage.py you can use the code –

```
$ python manage.py help
```

The “myproject” subfolder – This folder is the actual python package of your project. It contains four files –

\_\_init\_\_.py – Just for python, treat this folder as package.

settings.py – As the name indicates, your project settings.

urls.py – All links of your project and the function to call. A kind of ToC of your project.

wsgi.py – If you need to deploy your project over WSGI.

## Setting Up Your Project

Your project is set up in the subfolder myproject/settings.py. Following are some important options you might need to set –

DEBUG = True

This option lets you set if your project is in debug mode or not. Debug mode lets you get more information about your project's error. Never set it to ‘True’ for a live project.

However, this has to be set to 'True' if you want the Django light server to serve static files. Do it only in the development mode.

```
DATABASES = {  
  
    'default': {  
  
        'ENGINE': 'django.db.backends.sqlite3',  
  
        'NAME': 'database.sql',  
  
        'USER': '',  
  
        'PASSWORD': '',  
  
        'HOST': '',  
  
        'PORT': '',  
  
    }  
}
```

Database is set in the 'Database' dictionary. The example above is for SQLite engine. As stated earlier, Django also supports –

MySQL (django.db.backends.mysql)

PostgreSQL (django.db.backends.postgresql\_psycopg2)

Oracle (django.db.backends.oracle) and NoSQL DB

MongoDB (django\_mongodb\_engine)

Before setting any new engine, make sure you have the correct db driver installed.

You can also set others options like: TIME\_ZONE, LANGUAGE\_CODE, TEMPLATE...

Now that your project is created and configured make sure it's working –

```
$ python manage.py runserver
```

You will get something like the following on running the above code –

Validating models...

0 errors found

September 03, 2015 - 11:41:50

Django version 1.6.11, using settings 'myproject.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CONTROL-C.

A project is a sum of many applications. Every application has an objective and can be reused into another project, like the contact form on a website can be an application, and can be reused for others. See it as a module of your project.

## Create an Application

We assume you are in your project folder. In our main “myproject” folder, the same folder then manage.py –

```
$ python manage.py startapp myapp
```

You just created myapp application and like project, Django create a “myapp” folder with the application structure –

myapp/

\_\_init\_\_.py

admin.py

models.py

tests.py

views.py

\_\_init\_\_.py – Just to make sure python handles this folder as a package.

admin.py – This file helps you make the app modifiable in the admin interface.

models.py – This is where all the application models are stored.

tests.py – This is where your unit tests are.

views.py – This is where your application views are.

### Get the Project to Know About Your Application

At this stage we have our "myapp" application, now we need to register it with our Django project "myproject". To do so, update INSTALLED\_APPS tuple in the settings.py file of your project (add your app name) –

```
INSTALLED_APPS = (  
  
    'django.contrib.admin',  
  
    'django.contrib.auth',  
  
    'django.contrib.contenttypes',  
  
    'django.contrib.sessions',  
  
    'django.contrib.messages',  
  
    'django.contrib.staticfiles',  
  
    'myapp',  
  
)
```

Creating forms in Django, is really similar to creating a model. Here again, we just need to inherit from Django class and the class attributes will be the form fields. Let's add a forms.py file in myapp folder to contain our app forms. We will create a login form.

myapp/forms.py

```
#-*- coding: utf-8 -*-
```

```
from django import forms
```

```
class LoginForm(forms.Form):
```

```
    user = forms.CharField(max_length = 100)
```

```
password = forms.CharField(widget = forms.PasswordInput())
```

As seen above, the field type can take "widget" argument for html rendering; in our case, we want the password to be hidden, not displayed. Many others widget are present in Django: DateInput for dates, CheckboxInput for checkboxes, etc.

## Using Form in a View

There are two kinds of HTTP requests, GET and POST. In Django, the request object passed as parameter to your view has an attribute called "method" where the type of the request is set, and all data passed via POST can be accessed via the request.POST dictionary.

Let's create a login view in our myapp/views.py –

```
#-*- coding: utf-8 -*-
```

```
from myapp.forms import LoginForm
```

```
def login(request):
```

```
    username = "not logged in"
```

```
    if request.method == "POST":
```

```
        #Get the posted form
```

```
        MyLoginForm = LoginForm(request.POST)
```

```
        if MyLoginForm.is_valid():
```

```
            username = MyLoginForm.cleaned_data['username']
```

```
        else:
```

```
            MyLoginForm = Loginform()
```

```
    return render(request, 'loggedin.html', {"username" : username})
```

The view will display the result of the login form posted through the loggedin.html. To test it, we will first need the login form template. Let's call it login.html.



```

<html>

<body>

    <form name = "form" action = "{% url "myapp.views.login" %}"

        method = "POST" >{% csrf_token %}

        <div style = "max-width:470px;">

            <center>

                <input type = "text" style = "margin-left:20%;"

                    placeholder = "Identifiant" name = "username" />

            </center>

        </div>

        <br>

        <div style = "max-width:470px;">

            <center>

                <input type = "password" style = "margin-left:20%;"

                    placeholder = "password" name = "password" />

            </center>

        </div>

        <br>

        <div style = "max-width:470px;">

            <center>

                <button style = "border:0px; background-color:#4285F4; margin-
top:8%;

                    height:35px; width:80%;margin-left:19%;" type = "submit"

```

```

        value = "Login" >

        <strong>Login</strong>

    </button>

</center>

</div>

</form>

</body>

</html>

```

The template will display a login form and post the result to our login view above. You have probably noticed the tag in the template, which is just to prevent Cross-site Request Forgery (CSRF) attack on your site.

```
{% csrf_token %}
```

Once we have the login template, we need the loggedin.html template that will be rendered after form treatment.

```

<html>

    <body>

        You are : <strong>{{ username }}</strong>

    </body>

</html>

```

Now, we just need our pair of URLs to get started: myapp/urls.py

```

from django.conf.urls import patterns, url

from django.views.generic import TemplateView

urlpatterns = patterns('myapp.views',

```

```
url(r'^connection/', TemplateView.as_view(template_name = 'login.html')),  
  
url(r'^login/', 'login', name = 'login'))
```

When accessing "/myapp/connection", we will get the following login.html template rendered –

### Setting Up Sessions

In Django, enabling session is done in your project settings.py, by adding some lines to the MIDDLEWARE\_CLASSES and the INSTALLED\_APPS options. This should be done while creating the project, but it's always good to know, so MIDDLEWARE\_CLASSES should have –

```
'django.contrib.sessions.middleware.SessionMiddleware'
```

And INSTALLED\_APPS should have –

```
'django.contrib.sessions'
```

By default, Django saves session information in database (django\_session table or collection), but you can configure the engine to store information using other ways like: in file or in cache.

When session is enabled, every request (first argument of any view in Django) has a session (dict) attribute.

Let's create a simple sample to see how to create and save sessions. We have built a simple login system before (see Django form processing chapter and Django Cookies Handling chapter). Let us save the username in a cookie so, if not signed out, when accessing our login page you won't see the login form. Basically, let's make our login system we used in Django Cookies handling more secure, by saving cookies server side.

For this, first let's change our login view to save our username cookie server side –

```
def login(request):  
  
    username = 'not logged in'
```

```

if request.method == 'POST':

    MyLoginForm = LoginForm(request.POST)

    if MyLoginForm.is_valid():

        username = MyLoginForm.cleaned_data['username']

        request.session['username'] = username

    else:

        MyLoginForm = LoginForm()

        return render(request, 'loggedin.html', {"username" : username})

```

Then let us create formView view for the login form, where we won't display the form if cookie is set –

```

def formView(request):

    if request.session.has_key('username'):

        username = request.session['username']

        return render(request, 'loggedin.html', {"username" : username})

    else:

        return render(request, 'login.html', {})

```

Now let us change the url.py file to change the url so it pairs with our new view –

```

from django.conf.urls import patterns, url

from django.views.generic import TemplateView

urlpatterns = patterns('myapp.views',

    url(r'^connection/', 'formView', name = 'loginform'),

    url(r'^login/', 'login', name = 'login'))

```

When accessing /myapp/connection, you will get to see the following page

# TESTING AND VALIDATION

## SYSTEM TEST

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 satisfactory, as shown by successful 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.

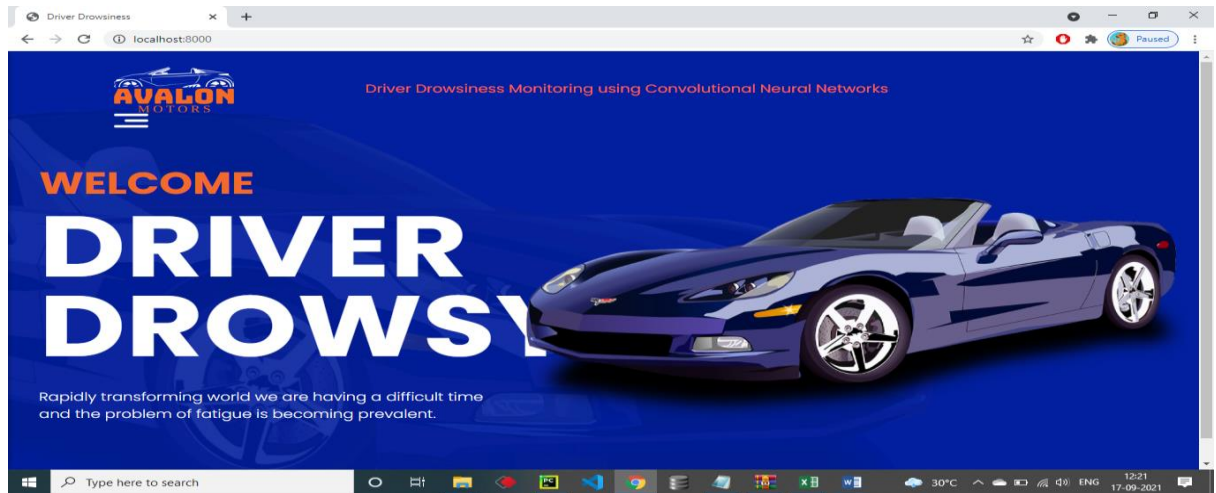


## TEST CASES

S.NO	Test Case	Excepted Result	Result	Remarks(IF Fails)
1.	User Register	If User registration successfully.	Pass	If already user email exist then it fails.
2.	User Login	If Username and password is correct then it will getting valid page.	Pass	Un Register Users will not logged in.
3.	User View User	Show our dataset	Pass	If Data set Not Available fail.
4.	View Fast History Results	The Four Alarm Score Should be Displayed.	Pass	The Four Alarm Score Not Displaying fail
5.	User Prediction	Display Review with true results	Pass	Results not True Fail
6.	Show Detection process	Display Detection process	Pass	Results Not True Fail
7.	Show Eye Blink Process	Display Eye Blink Process	Pass	If Results not Displayed Fail.
8.	Admin login	Admin can login with his login credential. If success he get his home page	Pass	Invalid login details will not allowed here
9.	Admin can activate the register users	Admin can activate the register user id	Pass	If user id not found then it won't login
10.	Results	For our Four models the accuracy and F1 Score	Pass	If Accuracy And F1 Score Not Displayed fail

## SCREEN SHOTS

### Home Page:



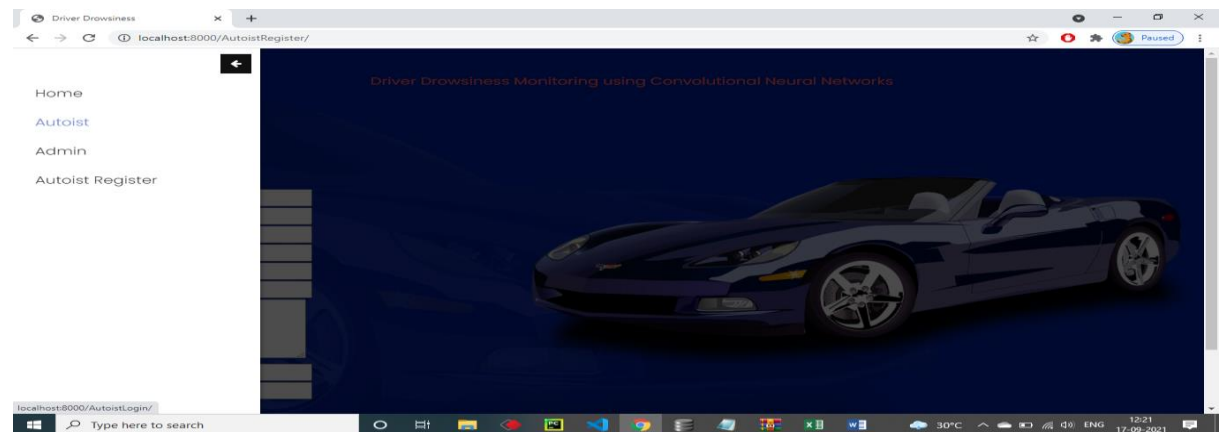
- Main interface showcasing system functionalities.
- Navigation links to autoist registration, system settings, and admin login.
- Overview of real-time monitoring and alert systems.

### Autoist Register:



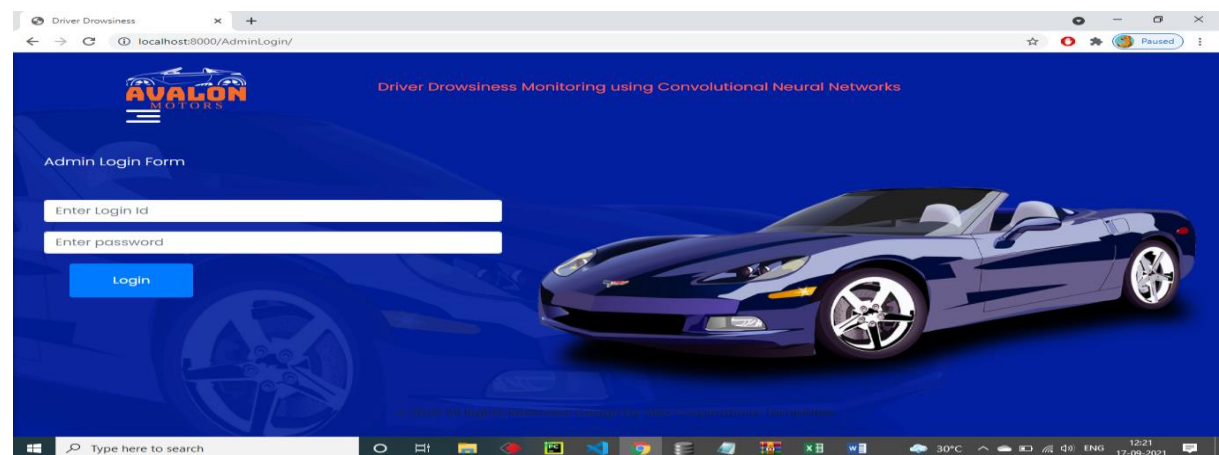
- Registration form for autoists/drivers.
- Fields for personal details, contact information, and driver identification.
- Validation checks to ensure accurate registration.

## System menu:



- Dashboard providing access to system functionalities.
- Options for autoist management, drowsiness detection analytics, and system settings.
- Navigation links for administrators to monitor alerts and generate reports.

## Admin Login:



- Secure portal for administrator access.
- Authentication process ensuring authorized entry.
- Gateway to administrative tools and system controls

## Admin Home Page:



- Control center for system management.
- Tools for managing autoist profiles and reviewing drowsiness detection reports.
- Configuration settings and analytics generation for system optimization.

## View Registered Autoist:



- List of registered autoists with relevant details (name, contact information, registration date).
- Options for admin actions (edit, delete, view details).

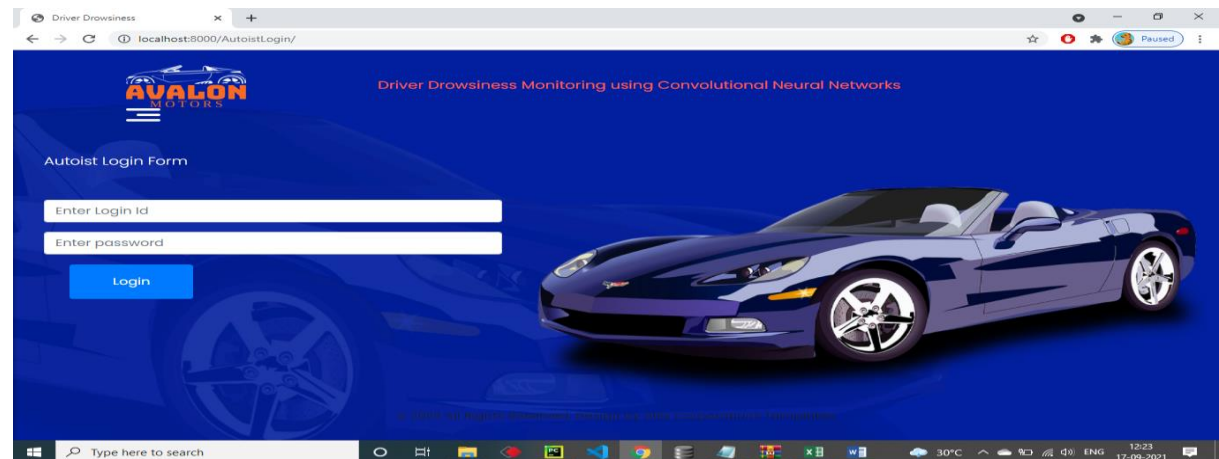
## Drowsiness Detections:



S.No	Name	Login User	Email	Vehicle Number	Latitude	Longitude	Fatigue	Date
1	alex	alex	lx160cm@gmail.com	API0BD8008	17.384	78.4564	Fatigue	Sept. 11, 2021
2	alex	alex	lx160cm@gmail.com	API0BD8008	17.384	78.4564	Fatigue	Sept. 11, 2021
3	alex	alex	lx160cm@gmail.com	API0BD8008	17.384	78.4564	Fatigue	Sept. 11, 2021
4	alex	alex	lx160cm@gmail.com	API0BD8008	17.384	78.4564	Fatigue	Sept. 11, 2021
5	alex	alex	lx160cm@gmail.com	API0BD8008	17.384	78.4564	Fatigue	Sept. 11, 2021
6	alex	alex	lx160cm@gmail.com	API0BD8008	17.384	78.4564	Fatigue	Sept. 11, 2021
7	alex	alex	lx160cm@gmail.com	API0BD8008	17.384	78.4564	Fatigue	Sept. 11, 2021
8	alex	alex	lx160cm@gmail.com	API0BD8008	17.384	78.4564	Fatigue	Sept. 17, 2021

- Loading TensorFlow for deep learning operations.
- Real-time monitoring of driver behavior using computer vision.
- Detection of eye blinks as indicators of drowsiness.
- Activation of alarms/alerts upon detection of fatigue signs.
- Display of fatigue and historical data for analysis.

## Autoist Login Form:



Autoist Login Form

Enter Login id

Enter password

Login

- Input fields for username/email and password.
- Login button and any additional security features (e.g., CAPTCHA).

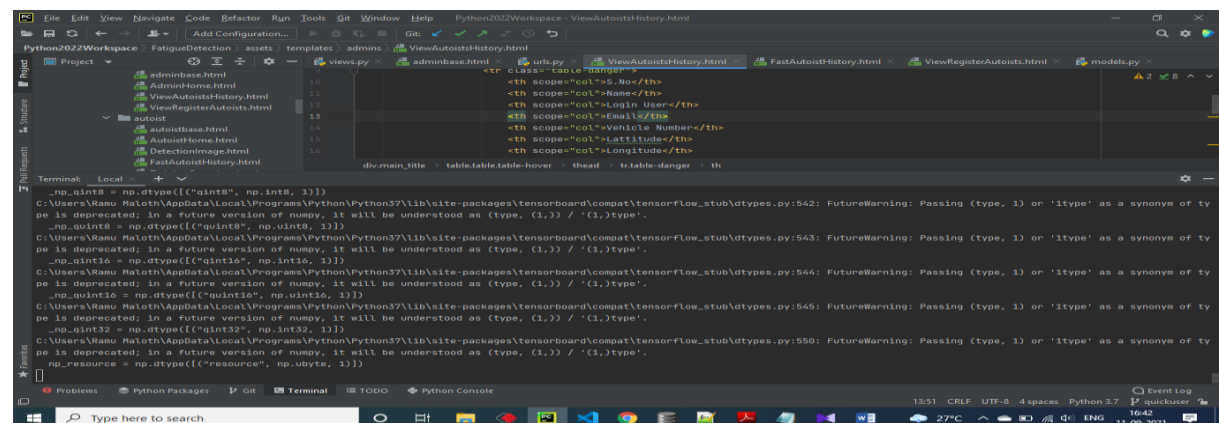


## Autoist Home Page:



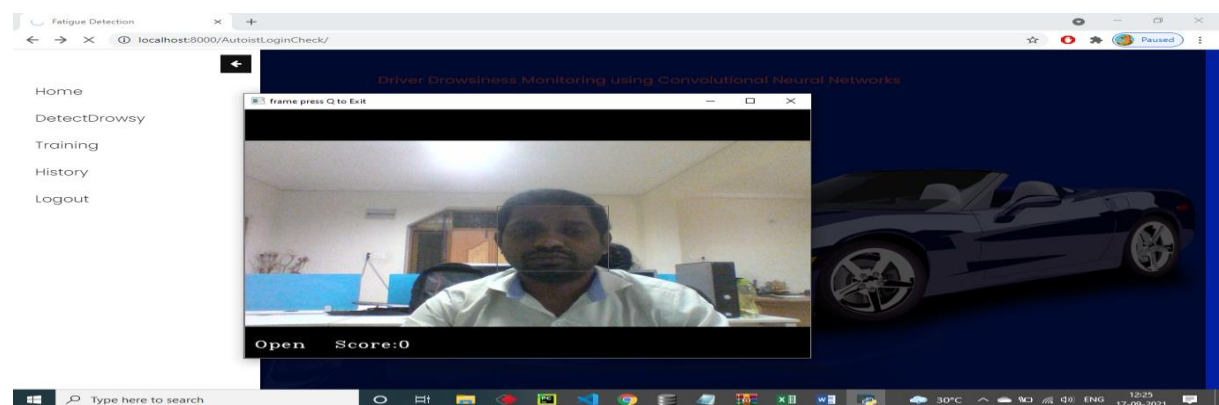
- Features accessible to autoists (e.g., fatigue level monitoring, driving history).
- Navigation menu or tabs for accessing different functionalities.

## Loading TensorFlow:



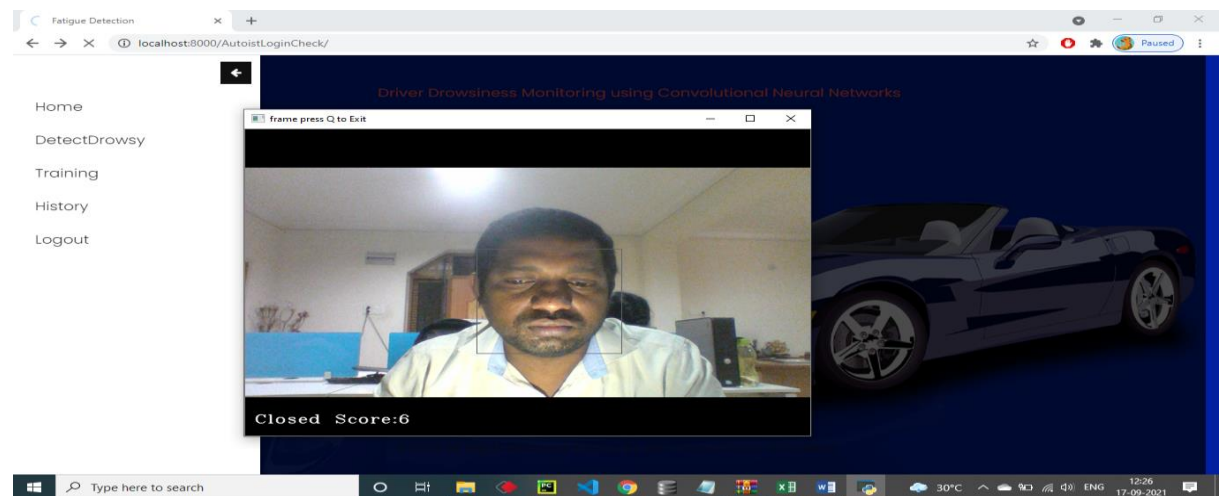
- Progress indicator or message indicating TensorFlow loading.
- Any initialization logs or status messages.

## Detection Process Started:



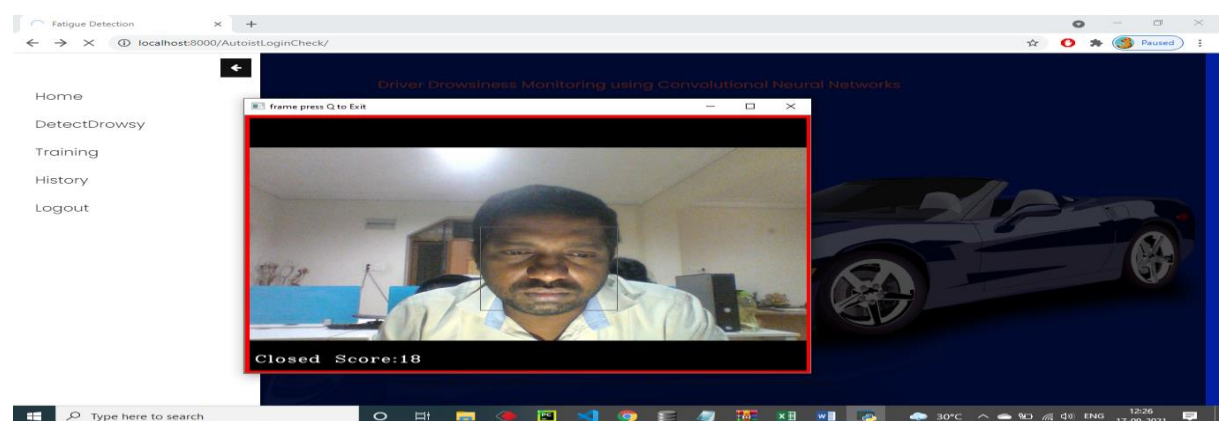
- Real-time video feed or simulation showing driver monitoring.
- Overlay or indicators showing active detection algorithms.

## Eye Blink Started:



- Visual indicators highlighting detected eye blinks.
- Metrics or logs showing frequency and duration of eye blinks.

## Alarm Started:



- Alert message or pop-up notifying about detected drowsiness.
- Options for the driver to acknowledge or respond to the alert.

## FatigueResults:

Driver Drowsiness Monitoring using Convolutional Neural Networks

Driver Drowsiness

Attribute Name	Fields
Logged User ID	alex
Autosist Name	alex
Email Name	ix160cm@gmail.com
Vehicle Number	API08D8008
Latitude	17.384
Longitude	78.4564
Fatigue Status	Fatigue
Date	Sept. 17, 2021, 12:26 p.m.

Open Score:16

- Graphs or charts depicting fatigue levels over time.
- Statistical summaries or trends in drowsiness detection.

## Fast History Results:

Driver Drowsiness Monitoring using Convolutional Neural Networks

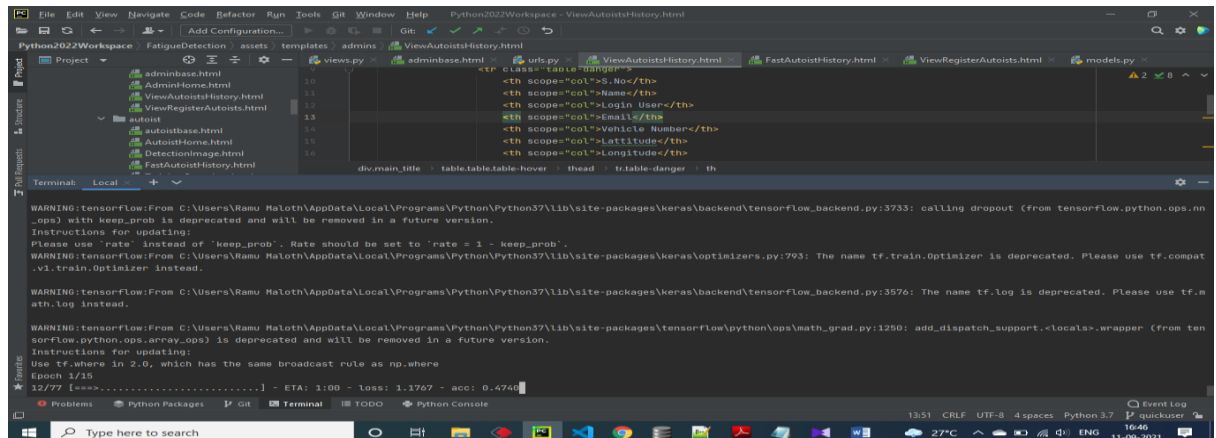
View Fast Detection Results

S.No	Name	Login User	Email	Vehicle Number	Latitude	Longitude	Fatigue	Date
1	alex	alex	ix160cm@gmail.com	API08D8008	17.384	78.4564	Fatigue	Sept. 15, 2021
2	alex	alex	ix160cm@gmail.com	API08D8008	17.384	78.4564	Fatigue	Sept. 15, 2021
3	alex	alex	ix160cm@gmail.com	API08D8008	17.384	78.4564	Fatigue	Sept. 15, 2021
4	alex	alex	ix160cm@gmail.com	API08D8008	17.384	78.4564	Fatigue	Sept. 15, 2021
5	alex	alex	ix160cm@gmail.com	API08D8008	17.384	78.4564	Fatigue	Sept. 15, 2021
6	alex	alex	ix160cm@gmail.com	API08D8008	17.384	78.4564	Fatigue	Sept. 15, 2021
7	alex	alex	ix160cm@gmail.com	API08D8008	17.384	78.4564	Fatigue	Sept. 15, 2021
8	alex	alex	ix160cm@gmail.com	API08D8008	17.384	78.4564	Fatigue	Sept. 17, 2021
9	alex	alex	ix160cm@gmail.com	API08D8008	17.384	78.4564	Fatigue	Sept. 17, 2021

- Displays visual timelines or graphs of recent drowsiness alerts.
- Includes statistical summaries of alert frequency and duration.
- Facilitates comparative analysis with historical data to identify trends and patterns.



## Training Process Started:



The screenshot shows a Python IDE with a project named 'Python2022Workspace'. The file explorer on the left shows a directory structure with files like 'adminbase.html', 'ViewAutoistHistory.html', 'FastAutoistHistory.html', 'ViewRegisterAutoists.html', 'models.py', 'views.py', 'adminbase.html', 'autoistbase.html', 'AutoistHome.html', 'DetectionImage.html', and 'FastAutoistHistory.html'. The main editor shows a table structure in 'ViewAutoistHistory.html' with columns for 'S.No', 'Name', 'Login User', 'Email', 'Vehicle Number', 'Latitude', and 'Longitude'. The terminal at the bottom displays several warnings from TensorFlow and Keras, including deprecation warnings for 'keep\_prob', 'tf.train.Optimizer', 'tf.log', and 'tf.where'. A progress bar at the bottom shows the training status: '12/77 [====...] - ETA: 1:00 - loss: 1.1767 - acc: 0.4740'.

- Training progress bar or status updates.
- Metrics or logs displaying model accuracy and convergence.

## **CONCLUSION**

A model for drowsiness sensing depends on effective CNN architecture, planned to observe drowsiness based on eye closure. The implementation started preparing image datasets for both open and closed eyes. 75% of the data set is used for the custom-designed CNN training and the balance 25% of the dataset is utilized for test purposes. First, the information video is transformed into frames and in each frame, the face and eyes are detected. The enhanced CNN supplied an automated and effective learned characteristics that aid us to categorize the opening or closing of eyes. If the closing of eyes occur in 15 successive frames, an alarm is triggered to alert the driver. The proposed CNN gives a training accuracy of 97% and a testing accuracy of 67%. For future works, extra face characteristics can be added to give more accuracy in detection. We can also combine vehicle driving pattern information obtained using On-Board Diagnostics sensors with the facial features extracted.

## **FURTHER ENHANCEMENT**

The proposed CNN gives a training accuracy of 97% and a testing accuracy of 67%. For future works, extra face characteristics can be added to give more accuracy in detection. We can also combine vehicle driving pattern information obtained using On-Board Diagnostics sensors with the facial features extracted.

## **REFERENCES**

- [1] Dr. Priya Gupta, Nidhi Saxena, Meetika Sharma, Jagriti Tripathi, Deep Neural Network for Human Face Recognition International Journal of Engineering and Manufacturing, vol.8, no.1, pp. 63-71. January 2018.
- [2] Jeyasekar A, Vivek Ravi Iyengar, Based on Behavioural Changes using ResNet , International Journal of Recent Technology and Engineering (IJRTE), vol. 8, no. 3, pp. 25-30, 2019.
- [3]Conference (IACC), Gurgaon, pp. 995-999, 2014.
- [4] Ki Wan Kim, Hyung Gil Hong, Gi Pyo Nam and Kang Ryoung Park, -
- [5] Luigi Celona, Lorenzo Mammana, Simone Bianco, Raimondo Schettini, -Task CNN Framework for Driver Face Berlin, 2018.
- [6] Mandalapu Sarada Devi and Dr. Preeti R Bajaj, Detection Based on Eye Tracking , First International Conference on Emerging Trends in Engineering and Technology, vol.1, pp. 649-652, 2008.
- [7] Sanghyuk Park, Fei Pan, Sunghun Kang and Chang D. Yoo, Driver drowsiness detection system based on feature representation learning Springer International Publishing, Computer Vision ACCV 2016 Workshops.
- [8] Tawsin Uddin Ahmed, Sazzad Hossain, Mohammed Shahadat Hossain, Raihan Ul Islam ,Karl Andersson, Facial Expression Recognition using Convolutional Neural Network with Data Th International Conference on Informatics, Electronics & Vision (ICIEV), Washington, USA,2019,
- [9] Upasana Sinha, Kamal K. Mehta, AK Shrivastava, Real Time Implementation for Monitoring Drowsiness Condition of a Train Driver using Brain Wave Sensor , International Journal of Computer Applications, vol. 139, no.9, pp. 25-30, 2016.
- [10] Xiaoxi Ma, Lap-Pui Chau and Kim-based Two-stream Convolutional Neural Networks for Driver Fatigue IEEE International Conference on Orange Technologies (ICOT), pp. 155-158, 2017.
- [11] Weiwei Zhang, Jinya Su Driver Yawning Detection based on Long , IEEE Symposium Series on Computational Intelligence (SSCI).