A PROJECT REPORT

on

DRIVER DROWSINESS DETECTION

Submitted in partial fulfillment of the requirements for the award of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

Submitted by

B. JAGADEESH	**-20BQ1A1217**
A. UMESH CHANDRA	**-20BQ1A1205**
G. UDAY SANKAR REDDY	**- 20BQ1A1255**
G. SATISH	**-20BQ1A1261**
CH. PRAKASH	**-20BQ1A1234**

Under the esteemed guidance

of **Mr. Md. Shakeel Ahmed**

Associate Professor

DEPARTMENT OF INFORMATION TECHNOLOGY

VASIREDDY VENKATADRI INSTITUTE OF

TECHNOLOGY

NAMBUR (V), PEDAKAKANI (M), GUNTUR-522 508, TEL no: 0873 2118036,

www.vvitguntur.com, approved by AICTE, permanently affiliated to JNTUK

Accredited by NAAC with "A" grade, Accredited by NBA for 3 years

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY:: NAMBUR

BONAFIDE CERTIFICATE

Certified that this project report **"DRIVER DROWSINESS DETECTION"** is the bonafide work of by "**B. JAGADEESH (20BQ1A1217), A. UMESH CHANDRA (20BQ1A1205), G. UDAY SANKAR REDDY (20BQ1A1255), G. SATISH (20BQ1A1261)**, **CH. PRAKASH (20BQ1A1234)"**. who carried out the project under my guidance during the year 2023 towards partial fulfillment of the requirements of the Degree of Bachelor of Technology in Information Technology from Jawaharlal Nehru Technological University, Kakinada. The results embodied in this report have not been submitted to any other University for the award of any degree.

Signature of the Supervisor

MD.SHAKEEL AHMED
Associate Professor, IT.

Signature of the Head of the Department

Dr KALAVATHI ALLA
Professor, IT.

Submitted for Viva voce Examination held on _____

EXTERNAL EXAMINER

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY:: NAMBUR

CERTIFICATE OF AUTHENTICATION

We solemnly declare that this project report **"DRIVER DROWSINESS DETECTION"** is the Bonafide work done purely by us, carried out under the supervision of Mr. Md. Shakeel Ahmed, towards partial fulfillment of the requirements of the Degree of Bachelor of Technology in Information Technology from Jawaharlal Nehru Technological University, Kakinada during the year 2023-24.

It is further certified that this work has not been submitted, either in part of in full, to any other department of the Jawaharlal Nehru Technological University, or any other University, institution or elsewhere, or for publication in any form.

Signature of the Student

B. JAGADEESH – 20BQ1A1217.
A. UMESH CHANDRA – 20BQ1A1205.
G. UDAY SANKAR REDDY -20BQ1A1255.
G. SATISH – 20BQ1A1261.
CH. PRAKASH -20BQ1A1234

ACKNOWLEDGEMENT

We take this opportunity to express our deepest gratitude and appreciation to all those people who made this project work easier with words of encouragement, motivation, discipline, and faith byoffering different places to lookto expand my ideas and help me towards the successful completion of this project work.

First and foremost, we express our deep gratitude to **Mr. Vasireddy VidyaSagar**, Chairman, Vasireddy Venkatadri Institute of Technology for providing necessary facilities throughout the Information Technology program.

We express our sincere thanks to **Dr.Y.Mallikarjuna Reddy**, Principal, Vasireddy Venkatadri Institute of Technology for his constant support and cooperation throughout the Information Technology program.

We express our sincere gratitude to **Dr.A.Kalavathi**, Professor & HOD, Information Technology, Vasireddy Venkatadri Institute of Technology for her constant encouragement, motivation and faith by offering different places to look to expand my ideas. We would like to express our sincere gratitude to our guide **Mr.Md.Shakeel Ahmed** for his insightful advice, motivating suggestions, invaluable guidance, help and support in successful completion of this project.

We would like to take this opportunity to express our thanks to the **teaching and non-teaching** staff in the Department of Information Technology, VVIT for their invaluable help and support.

B.Jagadeesh
A.Umesh Chandra
G.Uday Sankar Reddy
G.Satish
Ch.Prakash

ABSTRACT

The proposed system represents a revolutionary breakthrough in the sphere of road safety optimization, embodying an intelligent driver drowsiness and distraction detection system that redefines the boundaries of technological advancement. Through the seamless integration of state-of-the-art computer vision techniques and intricate analysis of facial landmarks, it establishes a sophisticated framework capable of continuously monitoring and analyzing the driver's subtle eye and mouth movements in real-time. This comprehensive surveillance mechanism enables the system to promptly and effectively trigger a series of well-timed alarms and alerts, serving as a proactive safeguard against any signs of drowsiness or distraction that may compromise driving safety. By mitigating the potential risks associated with driver fatigue and inattention, the system plays a pivotal role in fostering a culture of responsible driving and bolstering overall road safety standards, ensuring the protection and security of all individuals on the road.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Introduction **01**

1.2 Need of the Project **02**

2 SYSTEM ANALYSIS

2.1 Requirement Analysis. **03**

2.1.1 Nonfunctional requirements. **03**

2.2 Hardware requirements. **03**

2.3 Software requirements. **04**

2.4 Existing System. **04**

2.5 Proposed System. **04**

2.6 Modules. **05**

3 SYSTEM DESIGN

3.1 System Architecture. **06**

3.2 UML Diagrams. **07**

3.2.1 Use case diagram. **09**

3.2.2 Activity Diagram. **11**

3.2.3 Sequence diagram. **13**

3.2.4 Class diagram. **14**

4 SYSTEM IMPLEMENTATION

4.1 Technologies Used. **15**

4.1.1 Python. **16**

4.1.2 Pycharm. **17**

4.1.3 OpenCV. **19**

4.1.4 Dlib. **19**

4.1.5 Pygame **20**

4.1.6 Imutils **20**

4.2 Algorithm. **21**

4.2.1 Initialization and User Interface Setup. **21**

4.2.2 Facial Processing and Parameter Collection **21**
4.2.3 Drowsiness and distraction System **21**
4.2.4 Distraction Management **22**
4.2.5 Shutdown and Resource Release **22**
4.3 Sample Code **25**
4.4 Output Screens **27**

5 SYSTEM TESTING

5.1 Purpose of Testing **28**
5.2 Testing Strategies **28**
5.2.1 Unit Testing **29**
5.2.2 Integration Testing **30**
5.2.3 System Testing **30**

6 CONCLUSION AND FUTURE SCOPE

6.1 Conclusion **31**
6.2 Future Scope **31**

APPENDIX 4 . . . . . **32**

LIST OF FIGURES

S. NO	TITLE	PAGE NO
3.1	System Architecture	06
3.2.1	Use Case diagram	09
3.2.2	Activity diagram	11
3.2.3	Sequence Diagram	13
3.2.4	Class diagram	14
4.4.1	Face Recognition	26
4.4.2	Drowsiness Detection	26
4.4.3	Distraction Detection	27
5.1	Types of Testing	29

CHAPTER I

INTRODUCTION

1.1 INTRODUCTION

Driver fatigue and distractions are undeniably significant contributors to the alarming rates of road accidents worldwide. In response to this pressing issue, the proposed system represents a pioneering solution that harnesses the power of cutting-edge computer vision technologies and intricate facial landmarks analysis. By meticulously tracking and analyzing the subtle movements of the driver's eyes and mouth in real time, the system adeptly identifies potential instances of drowsiness and distractions that could compromise the driver's ability to operate the vehicle safely.

Through the integration of advanced algorithms and libraries, the system can accurately assess the driver's physiological state, thereby enabling a comprehensive evaluation of key metrics such as the eye aspect ratio (EAR) and mouth aspect ratio (MAR). These metrics serve as crucial indicators, offering valuable insights into the driver's level of alertness and attentiveness. When deviations from the predetermined safe thresholds are detected, the system promptly initiates a series of strategically timed alarms and alerts, effectively notifying the driver of the imminent risks and prompting immediate corrective actions.

This proactive and preventive approach exhibits promising potential in curbing the incidence of accidents resulting from driver fatigue and distractions, thereby fostering a safer road environment for all stakeholders. By actively addressing the root causes of potential hazards, the system not only enhances the safety of individual drivers but also contributes to the overall well-being of the community at large, emphasizing the crucial role of technology in promoting responsible driving practices and reducing the prevalence of preventable road accidents.

1.2 NEED OF THE PROJECT

- Enhanced Road Safety: The proposed intelligent driver drowsiness and distraction detection system is imperative for mitigating the risks associated with driver fatigue and inattentiveness. By employing sophisticated computer vision techniques and facial landmarks analysis, the system can effectively monitor the driver's eye and mouth movements in real-time, triggering timely alarms and alerts to notify the driver of potential hazards.

- Prompt Alert System: The timely triggering of alarms and alerts in response to identified signs of drowsiness or distraction is crucial in providing the driver with immediate warnings, enabling them to take necessary corrective actions. This swift response mechanism serves as a vital safeguard, reducing the likelihood of accidents and promoting a more vigilant approach to driving safety.

- Real-time Monitoring: The system's ability to monitor the driver's eye and mouth movements in real-time ensures a comprehensive and dynamic assessment of the driver's state. This real-time monitoring capability enables the system to provide timely alerts and notifications, fostering a heightened sense of awareness and attentiveness among drivers.

- Accurate Analysis: Through the utilization of advanced algorithms and precise evaluation metrics, the system can accurately analyze the driver's physiological state. By incorporating key metrics like the eye aspect ratio (EAR) and mouth aspect ratio (MAR), the system can provide a comprehensive understanding of the driver's level of alertness and attentiveness, facilitating informed decision-making in critical driving scenarios.

- Comprehensive Safety Measures: The comprehensive safety measures implemented by the system, including the integration of facial landmarks analysis and real-time monitoring, contribute significantly to the overall enhancement of road safety standards. By addressing the root causes of accidents associated with driver fatigue and distractions, the system establishes a robust framework for fostering a culture of safe and responsible driving practices.

CHAPTER II

SYSTEM ANALYSIS

2.1 REQUIREMENT ANALYSIS

2.1.1 Non-Functional Requirements

Performance Requirements:

Real-Time Responsiveness:

Requirement: The system shall promptly exhibit real-time responsiveness, detecting signs of driver drowsiness and triggering timely alarms and alerts without any noticeable delays.

Rationale: Real-time responsiveness is critical for ensuring that the system can promptly identify and respond to potential instances of driver drowsiness.

Gestures Accuracy:

Requirement: The system shall accurately and reliably detect signs of driver drowsiness with a high level of precision, minimizing false positives and false negatives.

Rationale: Accurate drowsiness detection is essential for preventing unnecessary alarms and ensuring that genuine instances of driver drowsiness are promptly identified and addressed, thereby reducing the risk of accidents.

2.2 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by the software engineers as the starting point for the system design. In hardware requirement we require all those components which will provide the platform for the development of project. The minimum hardware required for the development of this project as follows –

- CPU : INTEL CORE I3 OR HIGHER.
- MEMORY : 4GB RAM AND ABOVE
- HARDWARE : WEBCAM

SOFTWARE REQUIREMENTS

The software requirements are the software specifications of the system. It should include both a definition and specification of requirements. It is a set of what the system should do rather than how it should do it. It is useful in estimating cost, planning team activities, performing tasks and tracking the team's progress throughout the development activity.

- LIBRARIES USED: OPEN CV, IMUTILS, PYGAME
- INTEGRATED DEVELOPMENT ENVIRONMENT: PYCHARM.
- BACK END: PYTHON.

2.3 EXISTING SYSTEM

The existing system utilizes libraries such as scipy, imutils, pygame, dlib, and cv2 to build a real-time driver drowsiness detection system. It processes video frames from a webcam, detects facial landmarks, calculates the eye aspect ratio and triggers alarms when the driver is detected to be drowsy. The system includes features to handle different levels of drowsiness scenarios.

2.4 PROPOSED SYSTEM

The proposed system enhances the capabilities of the existing system by introducing distraction detection in addition to drowsiness detection. It utilizes dlib for face and landmark detection, and calculates the eye and mouth aspect ratios. The system maintains different timers for drowsiness and distraction detection intervals. Alarms are triggered based on predefined thresholds and intervals. Additionally, it provides feedback on the level of drowsiness or distraction detected, offering a more comprehensive solution.

2.5 MODULES

In this project, there are four modules –

1. Face Recognition
2. Detection of Drowsiness
3. Detection of Distraction
4. User Interface and Interaction

Face Recognition:

The module focuses on precise recognition and tracking of the driver's facial features using advanced computer vision techniques. It efficiently identifies essential facial landmarks, enabling real-time monitoring of the driver's eye and mouth movements.

Detection of Drowsiness:

The system incorporates a robust drowsiness detection mechanism that continuously monitors the driver's eye activity, utilizing key metrics such as the eye aspect ratio (EAR). It promptly detects signs of drowsiness, triggering timely alerts to ensure prompt driver intervention.

Detection of Distraction:

A dedicated module is employed to identify potential driver distractions by analyzing specific behavioral patterns and facial movements. It effectively monitors the driver's facial expressions and head motions, allowing for timely intervention through alerts and notifications.

User Interface and Interaction:

The system features an intuitive interface that enables seamless interaction with the driver. It communicates essential information, including real-time monitoring data, alerts, and instructions, ensuring a user-centric approach to enhancing driving safety. Moreover, the system provides intuitive controls and feedback mechanisms for an engaging user experience.

CHAPTER III

SYSTEMDESIGN

3.1 SYSTEM ARCHITECTURE

A System Architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

A System Architecture can consist of system components and the sub-systems developed, that will work together to implement the overall system.

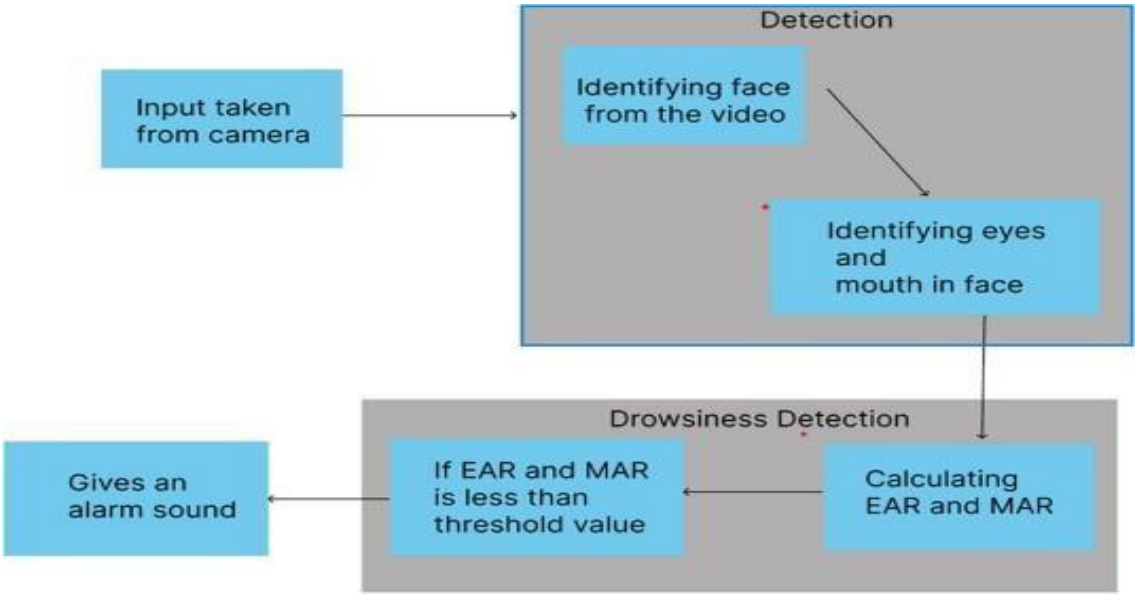


Fig 3.1 System Architecture

The project initiates by setting up the video capture interface using OpenCV, enabling access to the webcam for real-time data processing. Leveraging the dlib and imutils libraries, the system detects facial landmarks and computes essential ratios, such as the eye aspect ratio (EAR) and mouth aspect ratio (MAR), for drowsiness and distraction detection. Audio alerts are triggered via the Pygame mixer library based on predefined thresholds. The integration of these components results in an efficient driver drowsiness and distraction detection system, enhancing road safety using computer vision techniques.

3.2 UML DIAGRAMS

UML is an acronym that stands for **Unified Modeling Language**. Simply put, UML is a modern approach to modeling and documenting software. In fact, it's one of the most popular business process modeling techniques.

It is based on diagrammatic representations of software components. As the old proverb says: "a picture is worth a thousand words". By using visual representations, we are able to better understand possible flaws or errors in software or business processes.

Mainly, UML has been used as a general-purpose modeling language in the field of software engineering. However, it has now found its way into the documentation of several business processes or workflows. For example, activity diagrams, a type of UML diagram, can be used as a replacement for flowcharts. They provide both a more standardized way of modeling workflows as well as a wider range of features to improve readability and efficacy.

Any complex system is best understood by making some kind of diagrams or pictures. These diagrams have a better impact on our understanding. There are two broad categories of diagrams and they are again divided into subcategories −

Structural

Diagrams

Behavioral

Diagrams

Structural Diagrams

The structural diagrams represent the static aspect of the system. These static aspects represent those parts of a diagram, which forms the main structure and are therefore stable.

These static parts are represented by classes, interfaces, components, objects, and nodes. The four structural diagrams are :

Class diagram

Object diagram

Component

diagram

Behavioral Diagrams

Any system can have two aspects, static and dynamic. So, a model is considered as complete when both the aspects are fully covered. Behavioral diagrams basically capture the dynamic aspect of a system. UML has the following four types of behavioral diagrams

Use case diagram

Sequence diagram

Collaboration

diagram Statechart

diagram Activity

diagram

3.2.1 USE CASE DIAGRAM

A use case diagram at the simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

While a use case itself might drill into a lot of detail about every possibility, a use-case diagram can help provide a high-level view of the system. It has been said before that "Use case diagrams are the blueprints for your system". They provide a simplified and graphical representation of what the system must actually do.

Due to their simplistic nature, use case diagrams can be a good communication tool for stakeholders. The drawings attempt to mimic the real world and provide a view for the stakeholder to understand how the system is going to be designed.

The purpose of the use case diagram is simply to provide the high-level view of the system and convey the requirements in laypeople's terms for the stakeholders. Additional diagrams and documentation can be used to provide a complete functional and technical view of the system.

Use case diagrams are used to gather the requirements of a system including internal and

external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified.

When the initial task is complete, use case diagrams are modelled to present the outside view. In brief, the purpose of use case diagrams can be said to be as follows –

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.
- Shows the interactions among the requirements are actors.

Use-case diagram:

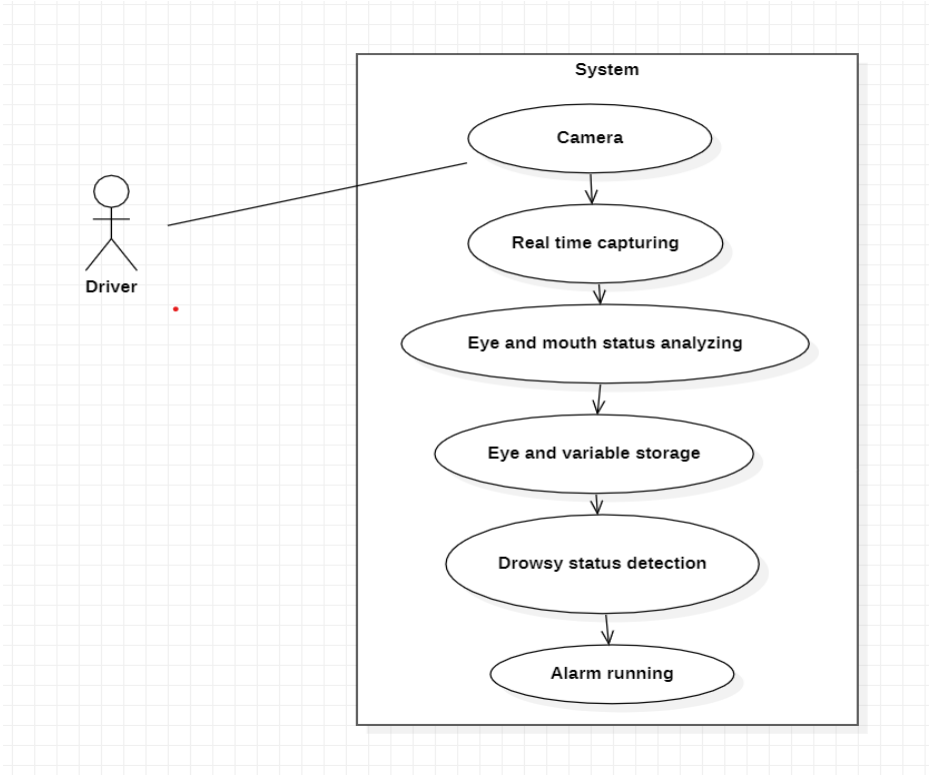


Fig 3.2 Use-case diagram

3.3.2 ACTIVITY DIAGRAM

Activity Diagram is a behavioral diagram that represents the flow of activities in a system or a business process. Activity diagrams are commonly used to model workflows, business processes,

software algorithms, and other dynamic aspects of a system. They provide a visual representation of the sequential and parallel activities that make up a complex process. Here's an overview of key elements and concepts related to Activity Diagrams:

Elements of an Activity Diagram:

- Activity: An activity is a specific task or operation that occurs within a process. Activities are represented by rounded rectangles and are connected by arrows to show the flow of control.

- Action: Actions represent individual steps or computations within an activity. They are typically depicted as rectangles with rounded corners. Actions can be simple, like "print receipt," or complex, involving multiple sub-activities.

- Control Flow: Control flow arrows indicate the sequence in which activities are executed. Arrows connect activities, showing the order of execution. Arrows may have a guard condition that determines under what circumstances they are followed.

- Fork and Join Nodes: Fork nodes (represented by a solid black bar) split a single flow of control into multiple concurrent flows, allowing activities to be executed in parallel. Join nodes (represented by a dashed line) merge multiple flows back into a single control flow.

- Decision Nodes: Decision nodes (diamond shapes) are used to represent points in the process where a decision is made. The outgoing arrows from a decision node have guard conditions that determine which path to follow based on a specific condition.

- Initial Node and Final Node: An initial node (a solid black circle) indicates the starting point of the activity diagram. A final node (a solid circle with a dot inside) represents the end of the process or activity.

- Swimlanes: Swimlanes are used to organize activities into different partitions or responsibilities. Each swimlane typically represents a role, department, or system component responsible for certain activities.

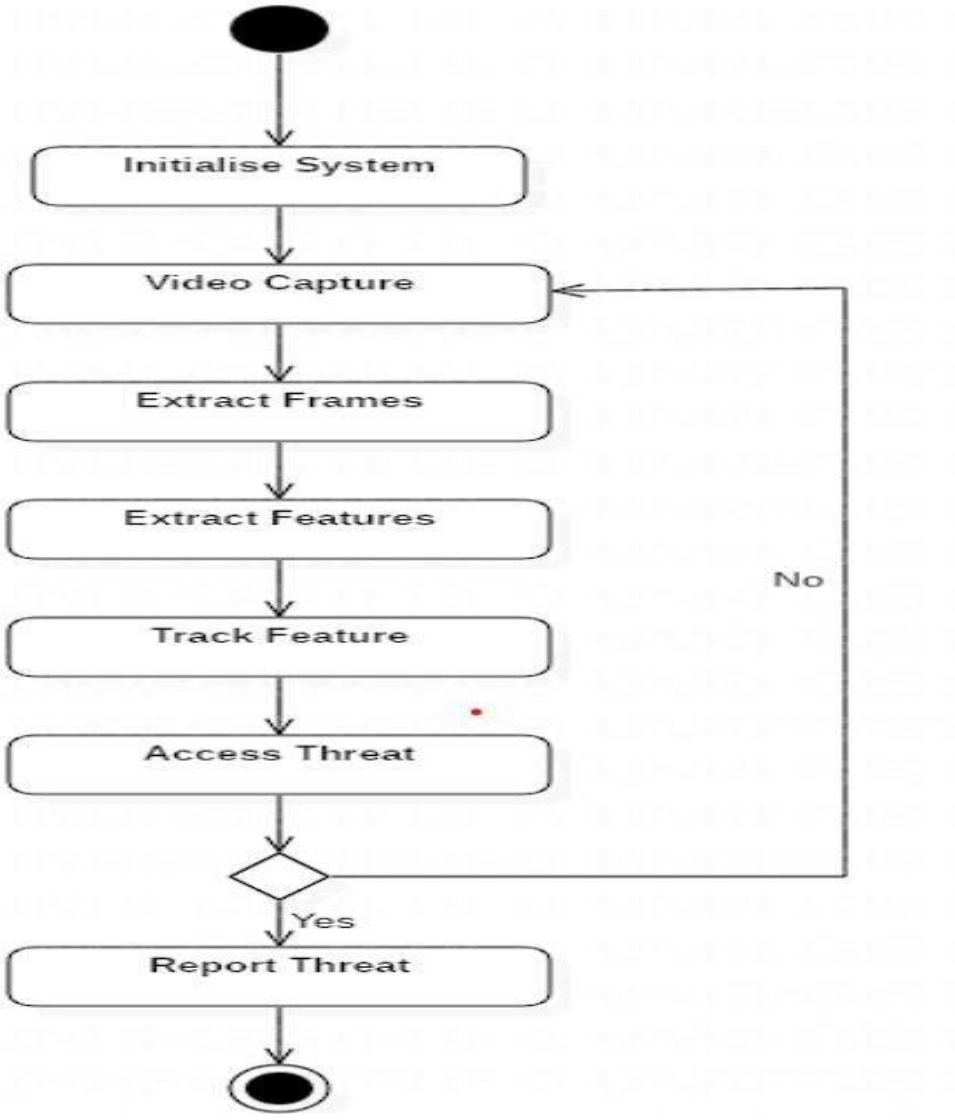


Fig 3.2 Activity diagram

3.2.3 SEQUENCE DIAGRAM

A sequence diagram is a type of Unified Modeling Language (UML) diagram that visually represents the interactions and relationships between objects in a system or software application over time. It is particularly useful for modeling how different components or objects in a system collaborate to achieve a specific task or use case. Here are some key components and concepts of sequence diagrams in UML:

- Lifeline: A lifeline is a vertical dashed line that represents an object or participant

involved in the interaction. It extends from the top to the bottom of the diagram. Each lifeline corresponds to an instance of a class or object.

- Actor: Actors, represented by stick figures, are external entities that interact with the system. They are not part of the system but can initiate interactions with it.

- Messages: Messages are represented by arrows and lines between lifelines. They depict the flow of communication or interaction between objects. Messages can be synchronous (denoted by a solid arrowhead) or asynchronous (denoted by an open arrowhead).

- Activation Bar: An activation bar is a rectangular box drawn on a lifeline to represent the period during which an object is active and processing messages. It shows when an object is actively participating in the interaction.

- Return Message: Return messages indicate the response of an object to a previously received message. They connect to the original message using a dashed line.

- Self Message: A self message is a message that an object sends to itself. It is represented by a loopback arrow to the same lifeline.

- Creating and Destroying Objects: Objects can be created and destroyed during the interaction. Object creation is represented by a message with a "+". Object destruction is shown using a large X.

- Combined Fragments: Combined fragments, enclosed by a box with a specific type, define conditional or looping behaviors in the sequence diagram. Common types include "alt" for alternative flows and "loop" for repetitive actions.

- Optimal Use: Sequence diagrams are best suited for showing interactions in a specific use case or scenario. They are particularly useful for understanding the detailed behavior of a system during a particular task.

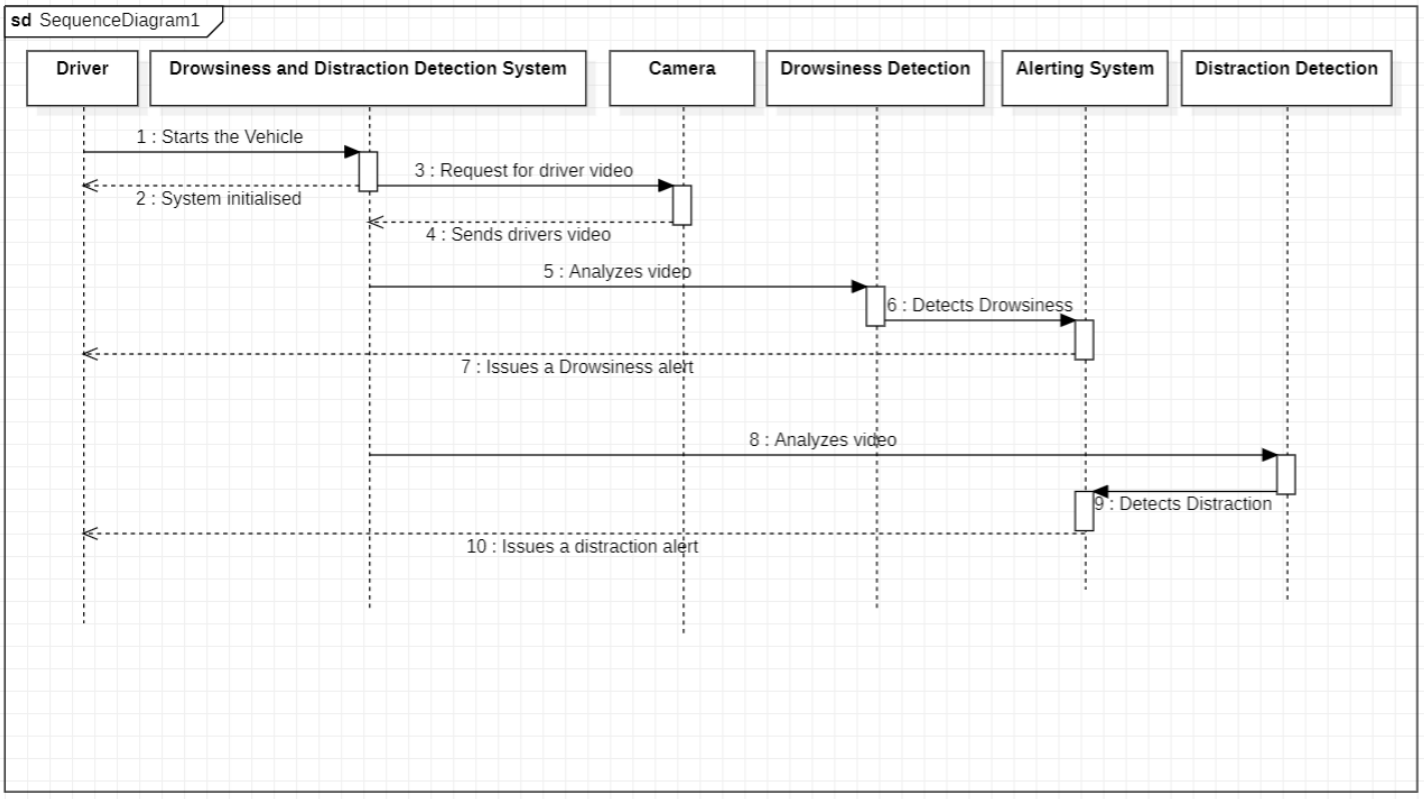


Fig 3.5 Sequence Diagram

3.2.4 CLASS DIAGRAM

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.

It shows the attributes, classes, functions, and relationships to give an overview of the software system. It constitutes class names, attributes, and functions in a separate compartment that helps in software development. Since it is a collection of classes, interfaces, associations, collaborations, and constraints, it is termed as a structural diagram.

The class diagram is made up of three sections:

Upper Section : The upper section encompasses the name of the class. A class is a representation of similar objects that shares the same relationships, attributes, operations, and semantics.

Middle Section : The middle section constitutes the attributes, which describe the quality of the class.

Lower Section : The lower section contain methods or operations. The methods are represented in the form of a list, where each method is written in a single line. It demonstrates how a class interacts with data.

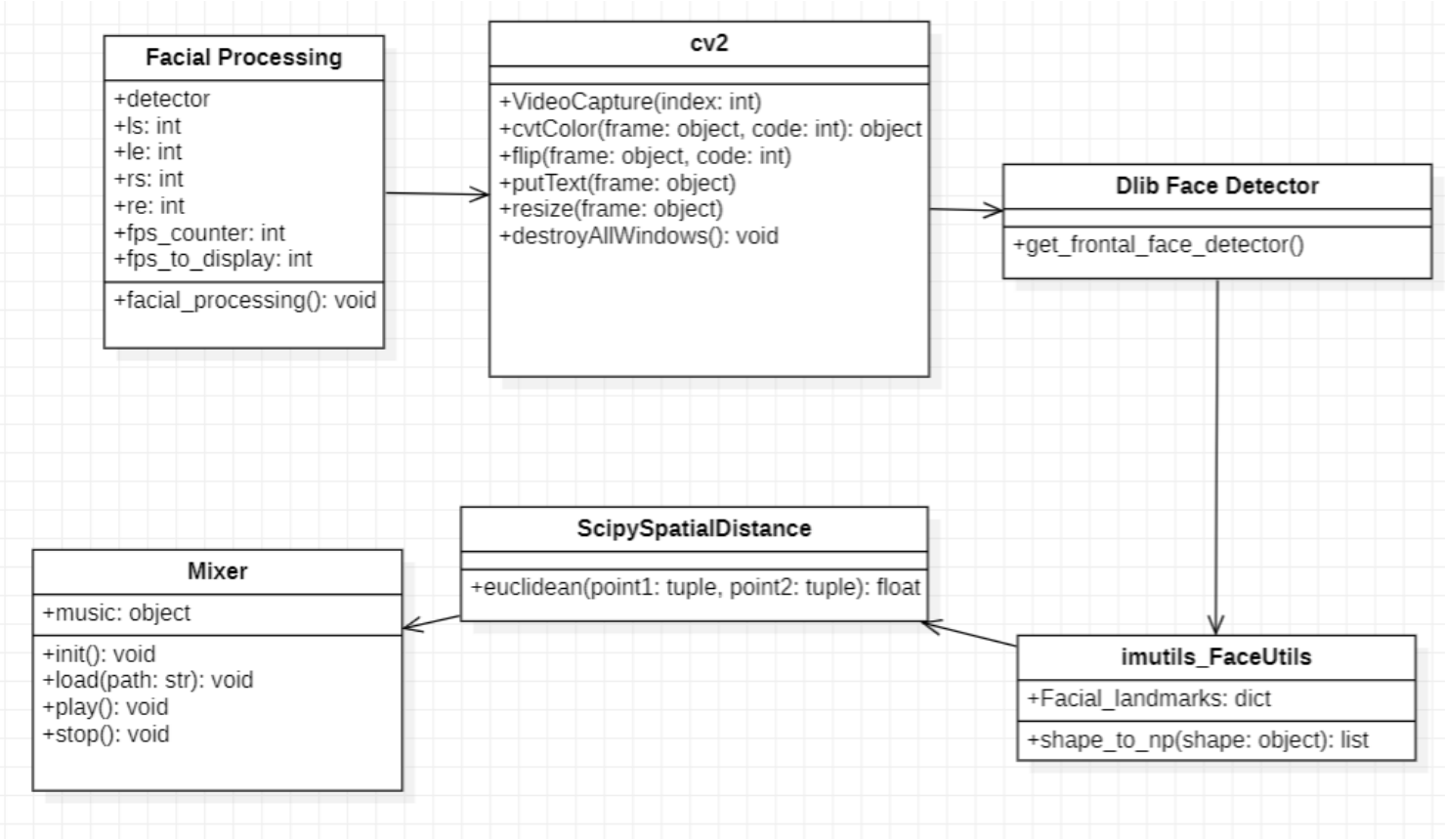


Fig 3.6 Class Diagram

CHAPTER IV

SYSTEM IMPLEMENTATION

4.1 TECHNOLOGIES USED:

4.1.1 PYTHON

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object- oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

The Python 2 language, i.e. Python 2.7.x, was officially discontinued on 1 January 2020 (first planned for 2015) after which security patches and other improvements will not be released for it. With Python 2's end-of-life, only Python 3.5.x and later are supported.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source reference implementation. A non- profit organization, the Python Software Foundation, manages and directs resources

for Python and CPython development.

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)) Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle- detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is sometimes termed the off-side rule, which some other languages share, but in most languages, indentation doesn't have any semantic meaning.

4.1.2 PYCHARM

PyCharm, developed by JetBrains, is a versatile Integrated Development Environment (IDE) exclusively crafted for Python programming. Equipped with a user-friendly interface and a comprehensive set of features, PyCharm offers a seamless development experience for Python developers of all levels.

It supports various Python frameworks, including Django, Flask, and Pyramid, providing developers with a wide array of options for building Python applications.

With its robust code analysis tools, PyCharm facilitates error-checking and ensures the creation of clean, efficient, and error-free Python code. Its intuitive code editor comes with essential features such as syntax highlighting, code completion, and code formatting, enhancing developers' productivity and code readability.

The IDE's intelligent code navigation capabilities enable easy traversal through complex codebases, allowing developers to effortlessly locate and modify code segments.

PyCharm integrates seamlessly with popular version control systems such as Git, Mercurial, and Subversion, enabling efficient code collaboration and management within development teams.

The IDE's comprehensive support for various versions of Python, including Python 2.x and Python 3.x, ensures compatibility with a wide range of Python projects.

PyCharm's extensive plugin ecosystem provides additional functionalities and customizations, allowing developers to tailor the IDE to their specific project requirements.

Featuring a robust debugging tool, PyCharm enables developers to identify and fix issues quickly, thereby streamlining the debugging process and ensuring the smooth functioning of Python applications.

Its unit testing framework allows developers to perform comprehensive testing, ensuring the reliability and stability of their Python code. Overall, PyCharm serves as a reliable and efficient development environment for Python programmers, offering a blend of powerful tools and an intuitive interface to enhance their productivity and coding experience.

4.1.3 OPENCV

OpenCV(Open Source Computer Vision Library) is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such Numpy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV.

Officially launched in 1999 the OpenCV project was initially an Intel Research initiative to advance CPU-intensive applications, part of a series of projects including real-time ray tracing and 3D display walls.The main contributors to the project included a number of optimization experts in Intel Russia, as well as Intel's Performance Library Team.

The first alpha version of OpenCV was released to the public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and five betas were released between 2001 and 2005. The first 1.0 version was released in 2006. A version 1.1 "pre-release" was released in October 2008.

The second major release of the OpenCV was in October 2009. OpenCV 2 includes major changes to the C++ interface, aiming at easier, more type-safe patterns, new functions, and better implementations for existing ones in terms of performance (especially on multi- core systems). Official releases now occur every six months and development is now done by an independent Russian team supported by commercial corporations.

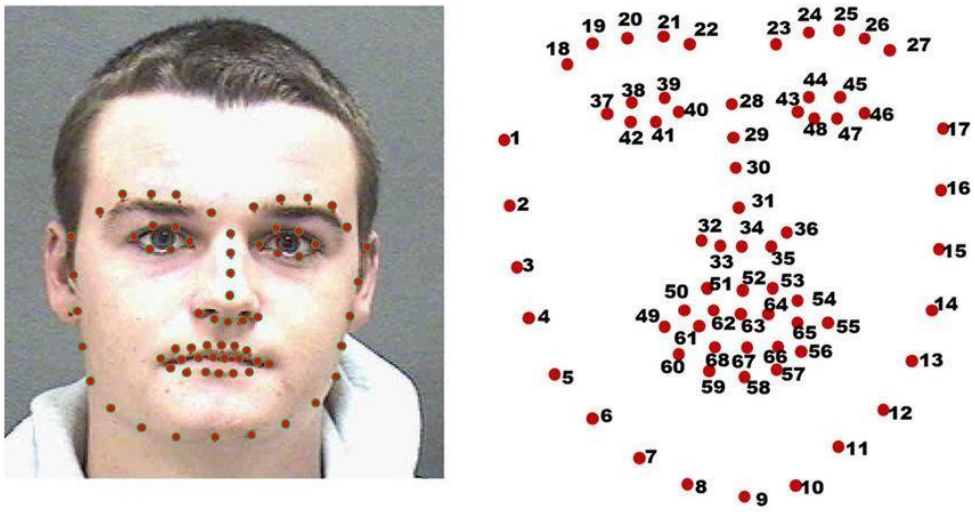
In August 2012, support for OpenCV was taken over by a non-profit foundation OpenCV.org, which maintains a developer and user site.In May 2016, Intel signed an agreement to acquire Itseez, a leading developer of OpenCV.In July 2020, OpenCV announced and began a Kickstarter campaign for the OpenCV AI Kit, a series of hardware modules and additions to OpenCV supporting Spatial AI.In August 2020, OpenCV launched OpenCV.ai – the professional consulting arm. The team of developers provides consulting

services and delivers Computer Vision, Machine Learning, and Artificial intelligence solutions.

OpenCV is written in the programming language C++, as is its primary interface, but it still retains a less comprehensive though extensive older C interface. All newer developments and algorithms appear in the C++ interface. There are language bindings in Python, Java, and MATLAB/Octave. The application programming interface (API) for these interfaces can be found in the online documentation. Wrapper libraries in several languages have been developed to encourage adoption by a wider audience. In version 3.4, JavaScript bindings for a selected subset of OpenCV functions were released as OpenCV.js, to be used for web platforms.

4.1.4 DLIB

The dlib library serves as a fundamental component for facial recognition and landmark detection. It enables the accurate identification of facial features, allowing for precise monitoring of the driver's eye and mouth movements. Its robust capabilities contribute to the efficient computation of essential metrics, such as the eye aspect ratio (EAR) and mouth aspect ratio (MAR), crucial for detecting signs of drowsiness and distraction.



Pygame: Pygame's robust audio functionalities come into play for triggering timely alerts and alarms in response to detected instances of drowsiness or distraction. Its efficient audio management capabilities ensure the seamless integration of audio feedback with the visual monitoring system, effectively alerting the driver to take necessary precautions.

imutils: The imutils library aids in streamlining the implementation of various image processing tasks, enhancing the overall efficiency and performance of the system. Its versatile set of tools enables convenient image manipulation and processing, contributing to the accurate analysis of facial landmarks and the extraction of essential features for drowsiness and distraction detection.

Scipy: The scipy package is an open-source library that provides a collection of mathematical algorithms and functions built on top of the NumPy extension for Python. It is designed to facilitate advanced scientific and technical computing, offering a wide array of tools for tasks such as optimization, integration, interpolation, linear algebra, and more. In the context of the driver drowsiness detection project, the scipy package is specifically utilized for its spatial module, which includes functions for calculating various distance metrics commonly used in scientific and mathematical applications. The distance module aids in the precise computation of distances between facial landmarks, enabling the accurate determination of crucial metrics such as the eye aspect ratio (EAR) and mouth aspect ratio (MAR) for detecting signs of driver drowsiness and distraction.

4.2 ALGORITHM :

4.2.1 Initialization and User Interface Setup

- Initialize the necessary libraries and modules, including scipy, imutils, pygame, dlib, and cv2.
- Setup the video capture interface to access the webcam feed using cv2.VideoCapture(0).
- Initialize the facial processing components, such as the frontal face detector and shape predictor from the dlib library.

4.2.2 Facial Processing and Parameter Collection

1. Real-Time Video Feed Processing

- Continuously capture frames from the webcam and process them for facial landmark detection and analysis.
- Convert the frames to grayscale for efficient processing using cv2.cvtColor().

2. Facial Landmark Detection and Calculation

- Detect facial landmarks using the shape predictor and face detector from dlib.
- Calculate the eye aspect ratio (EAR) and mouth aspect ratio (MAR) based on the detected facial landmarks.
- Utilize the distance module from scipy to precisely compute the required spatial metrics.

4.2.3 Drowsiness and Distraction Detection

1. Eye Drowsiness Detection

- Compare the computed eye aspect ratio with predefined thresholds to determine drowsiness.
- Initiate a warning message if the user's eye aspect ratio falls below the specified drowsiness threshold.
- Activate an alarm sound using Pygame's mixer library if distraction is detected simultaneously.

2. Mouth Drowsiness Detection

- Analyze the computed mouth aspect ratio to identify signs of drowsiness.
- Display a message if the mouth aspect ratio surpasses the predefined drowsiness threshold.
- Trigger an alarm to alert the user using Pygame's mixer library.

4.2.4 Distraction Management

- Track the absence of a frontal face in the video feed to detect potential distraction events.
- Display an alert message on the screen if distraction is detected, prompting the user to focus on the road.
- Play an alarm sound if distraction persists for an extended period.

4.2.5 Shutdown and Resource Release

- Allow the user to terminate the application using the 'q' key.
- Safely close all active components, including the video capture interface and graphical user interface (GUI).
- Release all resources and finalize the application execution.

4.3 SAMPLE CODE

DriverDrowsiness.py

```
from parameters import *
from scipy.spatial import distance
from imutils import face_utils as face
from pygame import mixer
import imutils
import time
import dlib
import cv2

# Some supporting functions for facial processing

def get_max_area_rect(rects):
    if len(rects)==0: return
    areas=[]
    for rect in rects:
        areas.append(rect.area())
    return rects[areas.index(max(areas))]

def get_eye_aspect_ratio(eye):
    vertical_1 = distance.euclidean(eye[1], eye[5])
    vertical_2 = distance.euclidean(eye[2], eye[4])
    horizontal = distance.euclidean(eye[0], eye[3])
    return (vertical_1+vertical_2)/(horizontal*2) #aspect ratio of eye

def get_mouth_aspect_ratio(mouth):
    horizontal=distance.euclidean(mouth[0],mouth[4])
    vertical=0
    for coord in range(1,4):
        vertical+=distance.euclidean(mouth[coord],mouth[8-coord])
    return vertical/(horizontal*3) #mouth aspect ratio
```

```
# Facial processing

def facial_processing():
    mixer.init()
    distracton_initlized = False
    eye_initialized      = False
    mouth_initialized    = False

    detector = dlib.get_frontal_face_detector()
    predictor   = dlib.shape_predictor(shape_predictor_path)

    ls,le = face.FACIAL_LANDMARKS_IDXS["left_eye"]
    rs,re = face.FACIAL_LANDMARKS_IDXS["right_eye"]

    cap=cv2.VideoCapture(0)

    fps_couter=0
    fps_to_display='initializing...'
    fps_timer=time.time()
    while True:
        _ , frame=cap.read()
        fps_couter+=1
        frame = cv2.flip(frame, 1)
        if time.time()-fps_timer>=1.0:
            fps_to_display=fps_couter
            fps_timer=time.time()
            fps_couter=0
        cv2.putText(frame, "FPS :"+str(fps_to_display), (frame.shape[1]-100, frame.shape[0]-10),\
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)


        #frame = imutils.resize(frame, width=900)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        rects = detector(gray, 0)
        rect=get_max_area_rect(rects)

        if rect!=None:

            distracton_initlized=False

            shape = predictor(gray, rect)
            shape = face.shape_to_np(shape)

            leftEye = shape[ls:le]
            rightEye = shape[rs:re]
            leftEAR = get_eye_aspect_ratio(leftEye)
            rightEAR = get_eye_aspect_ratio(rightEye)

            inner_lips=shape[60:68]
```

```
        mar=get_mouth_aspect_ratio(inner_lips)

        eye_aspect_ratio = (leftEAR + rightEAR) / 2.0

        leftEyeHull = cv2.convexHull(leftEye)
        rightEyeHull = cv2.convexHull(rightEye)
        cv2.drawContours(frame, [leftEyeHull], -1, (255, 255, 255), 1)
        cv2.drawContours(frame, [rightEyeHull], -1, (255, 255, 255), 1)
        lipHull = cv2.convexHull(inner_lips)
        cv2.drawContours(frame, [lipHull], -1, (255, 255, 255), 1)

        cv2.putText(frame, "EAR: {:.2f} MAR{:.2f}".format(eye_aspect_ratio,mar), (10,
frame.shape[0]-10),\
            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

        if eye_aspect_ratio < EYE_DROWSINESS_THRESHOLD:

           if not eye_initialized:
              eye_start_time= time.time()
              eye_initialized=True

           if time.time()-eye_start_time >= EYE_DROWSINESS_INTERVAL:
              alarm_type=0
              cv2.putText(frame, "YOU ARE SLEEPY...\nPLEASE TAKE A BREAK!", (10, 20),
                     cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)

              if  not distracton_initlized and not mouth_initialized and not mixer.music.get_busy():
                 mixer.music.load(alarm_paths[alarm_type])
                 mixer.music.play()
        else:
           eye_initialized=False
           if not distracton_initlized and not mouth_initialized and mixer.music.get_busy():
              mixer.music.stop()


        if mar > MOUTH_DROWSINESS_THRESHOLD:

           if not mouth_initialized:
              mouth_start_time= time.time()
              mouth_initialized=True

           if time.time()-mouth_start_time >= MOUTH_DROWSINESS_INTERVAL:
              alarm_type=0
              cv2.putText(frame, "YOU ARE YAWNING...\nDO YOU NEED A BREAK?", (10, 40),
                     cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)

              if not mixer.music.get_busy():
                 mixer.music.load(alarm_paths[alarm_type])
                 mixer.music.play()
        else:
           mouth_initialized=False
```

```
            if not distracton_initlized and not eye_initialized and mixer.music.get_busy():
                mixer.music.stop()


        else:

            alarm_type=1
            if not distracton_initlized:
                distracton_start_time=time.time()
                distracton_initlized=True

            if time.time()- distracton_start_time> DISTRACTION_INTERVAL:

                cv2.putText(frame, "EYES ON ROAD", (10, 20),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)

                if not eye_initialized and not mouth_initialized and not  mixer.music.get_busy():
                    mixer.music.load(alarm_paths[alarm_type])
                    mixer.music.play()

        cv2.imshow("Frame",  frame)
        key = cv2.waitKey(5)&0xFF
        if key == ord("q"):
            break

    cv2.destroyAllWindows()
    cap.release()


if _name=='__main_':
        facial_processing()
```

Parameters.py

```
import os

shape_predictor_path    =
os.path.join("C:/Users/jagad/PycharmProjects/pythonProject/shape_predictor_68_face_landmarks.dat")
alarm_paths             = [os.path.join("C:/Users/jagad/PycharmProjects/pythonProject/mixkit-censorship-
beep-long-1083.wav"),
                    os.path.join("C:/Users/jagad/Downloads/distraction_alert (1).wav"),
                    os.path.join("C:/Users/jagad/Downloads/distraction_alert.wav")]

EYE_DROWSINESS_THRESHOLD    = 0.25
EYE_DROWSINESS_INTERVAL     = 2.0
MOUTH_DROWSINESS_THRESHOLD  = 0.37
MOUTH_DROWSINESS_INTERVAL   = 1.0
DISTRACTION_INTERVAL        = 3.0
```

4.4 OUTPUT SCREENS

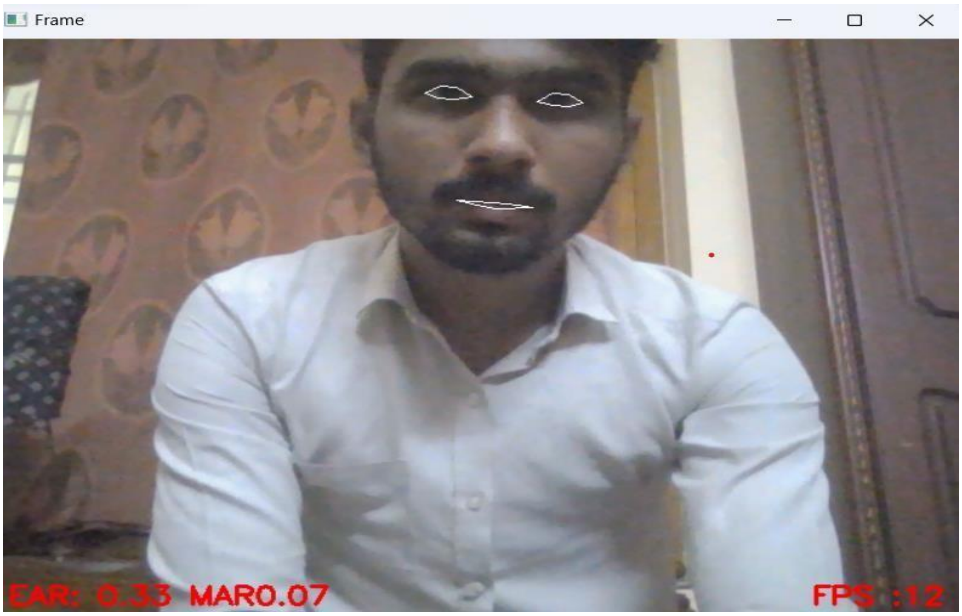


Fig 4.4.1: Face Recognition

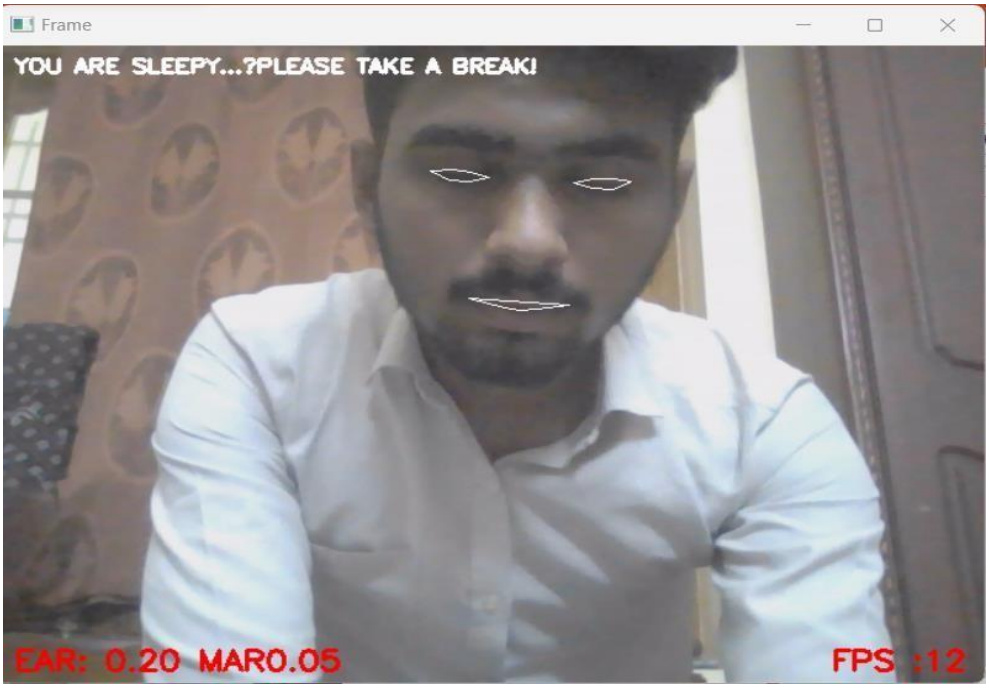


Fig 4.4.2:Detection of Drowsiness

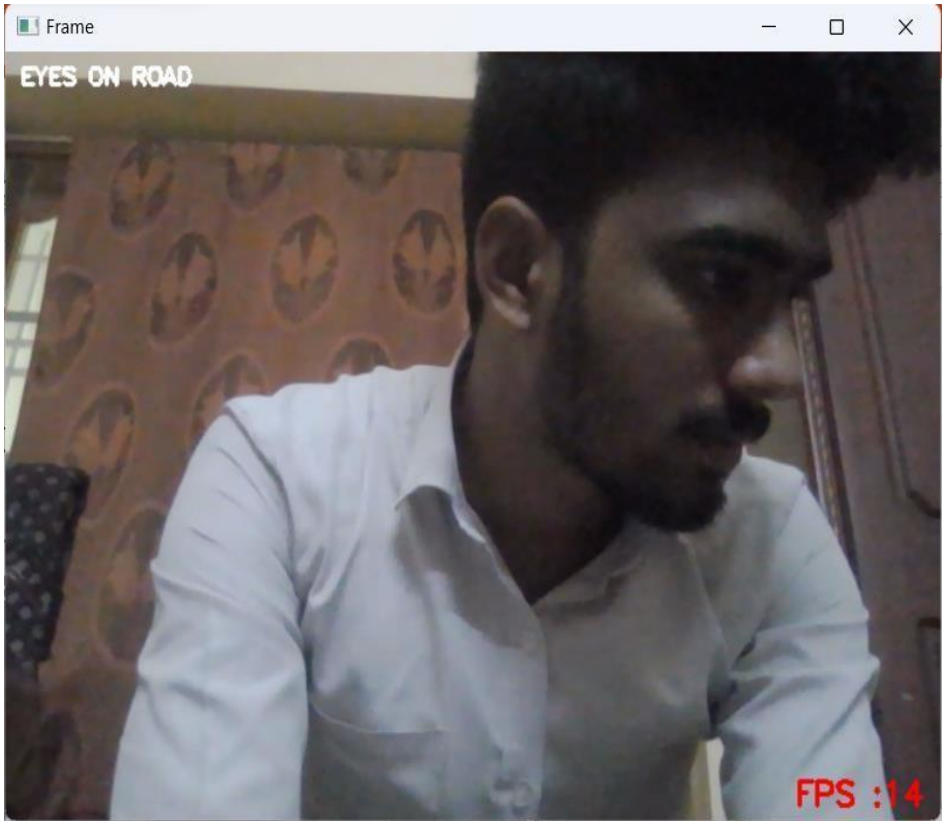


Fig 4.4.3: Detection Of Distraction

Chapter-5

System Testing

5.1 PURPOSE OF TESTING

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. 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 tests; each test type addresses a specific testing requirement. Testing and test design are parts of quality assurance that should also focus on bug prevention. A prevented bug is better than a detected and corrected bug. Testing consumes at least half of the time and work required to produce a functional program. History reveals that even well written programs still have 1-3 bugs per hundred statements. Testing is the process of executing a program with the aim of finding errors. To make our software perform well it should be error-free. If testing is done successfully, it will remove all the errors from the software.

5.2 TESTING STRATEGIES

In order to uncover the errors, present in different phases we have the concept of levels of testing. The Software testing has a prescribed order with the following list of software testing categories arranged in chronological order for our project testing and to generate test cases for output. These are the steps taken to fully test new software in preparation for marketing it:

Types of testing

- ***Unit testing*** performed on each module or block of code during development. Unit Testing is normally done by the programmer who writes the code.
- ***Integration testing*** done before, during and after integration of a new module into the main software package. This involves testing of each individual code module. One piece of software can contain several modules which are often created by several different programmers. It is crucial to test each module's effect on the entire program model.

- ***System testing*** done by a professional testing agent on the completed software product before it is introduced to the market.

- ***Acceptance testing*** - beta testing of the product done by the actual end users.

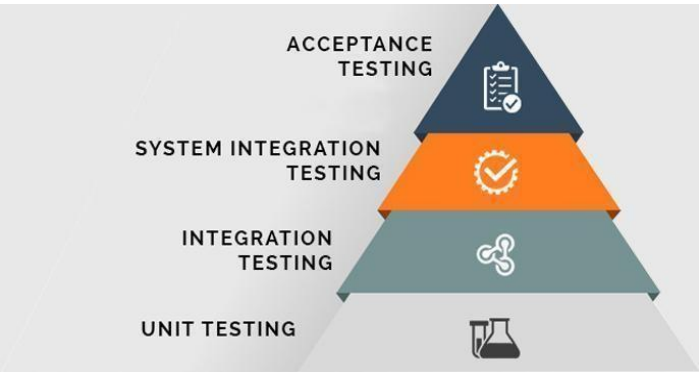


Fig 5.1 Types of Testing

5.2.1 UNIT TESTING

Test Cases:

1. **Drowsiness Detection Unit Testing:**
   - Verify the accuracy of the eye aspect ratio (EAR) calculation during drowsiness detection.
   - Test the system's ability to accurately identify drowsiness based on predefined EAR thresholds.

2. **Distraction Detection Unit Testing:**
   - Validate the functionality of distraction detection based on the absence of a frontal face in the video feed.
   - Test the system's response time and accuracy in alerting users about potential distractions.

3. **Audio Alarm Unit Testing:**
   - Ensure the audio alarms are triggered accurately for both drowsiness and distraction events.
   - Test the playback functionality and sound quality of the alarm system.

5.2.2 INTEGRATION TESTING

Test Cases:

1. **Drowsiness and Distraction Management Integration:**
   - Verify the seamless integration of drowsiness and distraction detection mechanisms.
   - Assess the system's ability to manage and alert users about both drowsiness and distraction simultaneously.

2. Real-Time Video Feed Processing and Parameter Calculation:

- Ensure the integration of the video feed processing with the facial landmark detection system for accurate parameter calculation.
- Assess the responsiveness and accuracy of the real-time parameter calculation during video feed processing.

3. Alarm Triggering and User Interface Interaction:

- Validate the proper synchronization between the alarm triggering system and the user interface.
- Test the system's responsiveness and reliability in providing timely alerts and feedback to the user.

5.2.3 SYSTEM *TESTING*

Test Cases:

1. End-to-End Scenario Testing:

- Execute common usage scenarios, including system startup, gesture recognition, and user interactions.

2. Performance Testing:

- Evaluate the system's responsiveness and speed in accurately recognizing and responding to facial cues and gestures.

3. User Experience and Satisfaction Testing:

- Solicit feedback from users with diverse levels of technical proficiency to assess the overall user experience and satisfaction with the system's functionality.

4. Error Handling and Exception Testing:

- Simulate scenarios with unrecognized facial patterns and unexpected user input to verify the system's robustness and error-handling capabilities.

5. Stress Testing:

- Subject the system to extended periods of operation to assess its stability and performance under prolonged use, ensuring it maintains consistent functionality.

6. Integration with External Applications:

- Verify the seamless integration of the system with external software,such as existing alert systems,to ensure smooth infteroperanility and effective communication between systems

CHAPTER VI

CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

This real-time facial monitoring system represents a pivotal advancement in ensuring the safety and well-being of individuals engaged in critical activities such as driving and healthcare. By harnessing the power of computer vision techniques and sophisticated facial landmark detection, this system effectively addresses the pressing concerns surrounding drowsiness and distraction, providing timely alerts and interventions to minimize the risk of accidents and errors. Its user-friendly nature allows for easy customization, enabling the integration of advanced features such as internet connectivity and the ability to discern subtle changes in speech patterns, thereby further enhancing its capabilities and potential impact. As the system continues to evolve and adapt, it has the capacity to revolutionize the way we approach safety and vigilance in various domains, contributing to a more secure and responsive environment for all.

6.2 FUTURE SCOPE

- **Adaptive Alarms :** Customize alarm responses based on the severity of drowsiness or distraction. For instance, the system can issue a gentle alert for mild drowsiness and a more urgent warning for extreme cases.
- **Multi-Modal Sensing:** Integrate additional sensors, such as heart rate monitors or EEG sensors, to complement facial monitoring. Combining multiple sources of data can enhance the system's accuracy and reliability in detecting drowsiness and distraction.
- **Integration with Wearables:** Extend the system's functionality to wearables, such as smart glasses or augmented reality (AR) headsets. Users can benefit from real-time alerts displayed in their field of vision.
- **Alert Preferences:** Allow users to customize their alert preferences, including the choice of alarms, alarm volume, and visual cues. Personalized settings can make the system more user-friendly.

APPENDIX

REFERENCES

Papers and articles

1]. Ameratunga.S, Bailey.J, Connor.J, Civil.I, Dunn.R, Jackson.R, Norton.R, and Robinson.E, ―Driver sleepiness and risk of serious injury to car occupants: Population control study.‖ British Medical Journal, vol. 324, 2002, pp. 1125–1129.

[2]. Bronte.S, Bergasa.L, Delgado.B, Garcia.I, Hernandez.N and Sevillano.M, ―Vision- based drowsiness detector for a realistic driving simulator, ‖ in IEEE Intelligent Transportations Systems Conference (ITSC), 2010.

[3]. Distante. A, D'Orazio. T, Guaragnella. C and Leo .M,
―A visual approach for driver inattention detection,
‖Pattern Recogn., vol.40, no. 8, 2007, pp. 2341–2355.

[4]. Bradski. G, Kaehler. A, -Learning OpenCV, O'Reilly, 2008.

[5]. Igarashi.K, Itou.K, Itakura.F, Miyajima.C, Ozawa.T,
Takeda.K and Wakita.T, ―Driver identification using
driving behavior signals, ‖ IEICE - Trans. Inf. Syst., vol. E89- D, 2006.

[6]. Nakano.T, Suzuki.M, Yamamoto.N, Yamamoto.O and Yamamoto.S, ―Measurement of driver's consciousness
by image processing a method for presuming driver's drowsiness by eye-blinks coping with individual differences.
‖ Systems, Man

`