A Webcam-Based Real-Time Fatigue Detection Model Using Facial Landmarks

Submitted By

Nazmul Hassan Ovi

221-35-903

Supervised By

Dr. Md. Abdul Kader

Associate Professor

This project report has been submitted in fulfilment of the requirements for the degree of **Bachelor of Science in Software Engineering**

APPROVAL

This thesis titled on **"A Webcam-Based Real-Time Fatigue Detection Model Using Facial Landmarks"**, submitted by **Student Name: Nazmul Hassan Ovi (ID: 221-35-903)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS

Chairman

Dr. A. H. M. Saifullah Sadi
Professor
Department of Software Engineering
Faculty of Science and Information Technology Daffodil International University

Internal Examiner 1

Dr. Rubaiyat Islam
Associate Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 2

Dr. Md. Abdul Kader
Associate Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 3

Nuruzzaman Faruqui
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

External Examiner

Md. Mostafiz Khan
Managing Director
Tecognize Solutions Limited

DAFFODIL INTERNATIONAL UNIVERSITY

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : ABC
Date of Birth :
Title :
Academic Session :

I declare that this thesis is classified as:

☐	CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
☐	RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
☐	OPEN ACCESS	I agree that my project to be published as online open access (Full Text)

I acknowledge that Daffodil International University reserves the following rights:

1. The Project is the Property of Daffodil International University.
2. The Library of Daffodil International University has the right to make copies of the Project for the purpose of research only.
3. The Library of Daffodil International University has the right to make copies of the Project for academic exchange.

Certified by:

(Student's Signature) (Supervisor's Signature)

Student ID Name of Supervisor
Date: Date:

NOTE: * If the Project is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

PROJECT DECLARATION LETTER (OPTIONAL)

Librarian,
Daffodil International University,
Daffodil Smart City,
Ashulia.Dhaka,Bangladesh

Dear Sir,

CLASSIFICATION OF Project AS RESTRICTED

Please be informed that the following project is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name
Project Title

Reasons (i)

(ii)

(iii)

Thank you.

Yours faithfully,

_____
(Supervisor's Signature)

Date:

Stamp:

Note: This letter should be written by the supervisor and addressed to the Librarian, *Daffodil International University* with its copy attached to the thesis.

SUPERVISOR'S DECLARATION

I/We* hereby declare that I/We* have checked this project and in my/our* opinion, this project is adequate in terms of scope and quality for the award of the degree of Bachelor of Science.

(Supervisor's Signature)

Full Name : **Dr. Md Abdul Kader**

Position : **Associate Professor**

Date : 27 November 2025

STUDENT'S DECLARATION

I hereby declare that the work in this project is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Daffodil International University or any other institution.

(Student's Signature)

Full Name : Nazmul Hassan Ovi
ID Number : 221-35-903
Date : 27 November 2025

A Webcam-Based Real-Time Fatigue Detection Model Using Facial Landmarks

Nazmul Hassan Ovi

Project submitted in fulfillment of the requirements
for the award of the degree of
Bachelor of Science/Master of Science

Department of Software Engineering

DAFFODIL INTERNATIONAL UNIVERSITY

November 2025

ACKNOWLEDGEMENTS

First of all, we are grateful to The Almighty Allah for making us eligible to complete this thesis.

Then we would like to thank our supervisor DR. MD ABDUL KADER, Department of Software Engineering. We are extremely grateful and indebted to her for hers expert, sincere and valuable guidance and encouragement extended to us.

I would also like to thank them who were involved in the survey for this thesis project. Without their passionate participation and input, the validation survey could not have been successfully conducted.

We take this opportunity to record our sincere thanks to all the faculty members of the Department of Software Engineering for their help and encouragement.

Last but not least, we would like to thank to our parents, for their unconditional support, love and without this we wo uld not have come this far.

DEDICATION

I therefore declare that I have done this project under the oversight of **DR. MD ABDUL KADER**, **Associate Professor**, Department of Software Engineering, Daffodil International University. Also declare that neither entire record nor any portion of this record has been submitted somewhere else for my degree.

ABSTRACT

In this project, an automated real-time drowsiness and micro sleep detection system is proposed, and it is developed with the help of the advanced computer-vision techniques. The system uses Mediapipe Face Mesh and OpenCV to identify 468 accurate facial landmarks and also calculates the important physiological measurements like the Eye Aspect Ratio (EAR), Mouth Aspect Ratio (MAR) and gaze deviation. These measurements can help the algorithm to determine the level of alertness in the user and can group the individual as Awake, Drowsy, Sleeping or as one that is undergoing Micro-Sleep.

The unique feature of the work is that it is applied to the Google Colab environment where webcam frames are captured with the help of the browser with the use of JavaScript integration periodically. This removes the special purpose hardware requirements and makes the system light, available and platform independent. Also, the model keeps a statistical record of the frequency of each state during the session, which allows conducting a more in-depth behavioral analysis and performance assessment.

The system has shown an effective and viable solution to the early detection of fatigue by integrating geometric analysis of faces with automated state identification. Some of the areas where it can be used are driver monitoring systems, solutions to occupational safety, remote-work monitoring, and human computer interaction where alertness is of paramount importance. Generally, this project shows how a combination of facial features landmarks and basic computation ratios can be used to create efficient fatigue-detection systems that can be operated in real time in cloud based setup.

TABLE OF CONTENT

LIST OF TABLES

LIST OF SYMBOLS

Symbol / Notation	Meaning / Description	Unit
EAR	Eye Aspect Ratio, used to measure eye openness/closure	Dimensionless
MAR	Mouth Aspect Ratio, used to measure mouth opening/yawning	Dimensionless
LM	Facial landmarks detected by MediaPipe Face Mesh (468 points)	—
(x, y, z)	Normalized landmark coordinates from MediaPipe	Normalized (0–1)
w	Width of the captured image frame	pixels
h	Height of the captured image frame	pixels
p1…p6	Eye landmark points used for EAR calculation	pixels
A	First vertical eye distance (between p2 and p6)	pixels
B	Second vertical eye distance (between p3 and p5)	pixels
C	Horizontal eye distance (between p1 and p4)	pixels
μ	Mean value (used while averaging points)	—
$\|u - v\|$	Euclidean distance between two points u and v	pixels
T_sleep	EAR threshold for Sleeping state	Dimensionless
T_drowsy	EAR threshold for Drowsy state	Dimensionless
T_micro	EAR threshold for Micro-Sleep detection	Dimensionless
T_yawn	MAR threshold for yawning detection	Dimensionless
N_blink	Consecutive frames count to confirm micro-sleep	frames
Δgaze	Gaze deviation ratio (eye center vs nose tip shift)	Dimensionless
FPS	Frames per second (rate of processing)	frames/sec

Symbol / Notation	Meaning / Description	Unit
Status	Output fatigue state (Awake/Drowsy/Sleeping/Micro-Sleep/No Face)	—
P(status)	Probability score assigned to detected state	%

LIST OF ABBREVIATIONS

Abbreviation	Full Form	Description
AI	Artificial Intelligence	Technology that enables systems to mimic human intelligence.
CV	Computer Vision	Field of AI that enables machines to interpret images/video.
EAR	Eye Aspect Ratio	Ratio used to measure eye openness for blink/sleep detection.
MAR	Mouth Aspect Ratio	Ratio used to measure mouth openness for yawn detection.
HCI	Human–Computer Interaction	Study of how humans interact with computer systems.
FPS	Frames Per Second	Number of frames processed per second in real-time detection.
RGB	Red Green Blue	Color model used for image processing in Mediapipe/OpenCV.
GUI	Graphical User Interface	Visual interface (not used in this project; console-based).
UML	Unified Modeling Language	Standard diagrams for object-oriented system design.
SRLC	Software Release Life Cycle	Phases of software release: pre-alpha to maintenance.
API	Application Programming Interface	Programming interface for using libraries/tools.
LM / LMs	Landmark(s)	Facial keypoints detected by Mediapipe Face Mesh.
CPU	Central Processing Unit	Main processing hardware unit used in Colab runtime.

CSV	Comma-Separated Values	File format for storing logs/results (future extension).
IoT	Internet of Things	Network of smart devices (future deployment scope).

LIST OF APPENDICES

ACKNOWLEDGEMENTS

First of all, we are grateful to The Almighty Allah for making us eligible to complete this thesis.

Then we would like to thank our supervisor DR. MD ABDUL KADER, Department of Software Engineering. We are extremely grateful and indebted to her for hers expert, sincere and valuable guidance and encouragement extended to us.

I would also like to thank them who were involved in the survey for this thesis project. Without their passionate participation and input, the validation survey could not have been successfully conducted.

We take this opportunity to record our sincere thanks to all the faculty members of the Department of Software Engineering for their help and encouragement.

Last but not least, we would like to thank to our parents, for their unconditional support, love and without this we wo uld not have come this far.

CHAPTER 1

INTRODUCTION

1.1 Background

The issue of human safety has become central to the contemporary society, particularly in the processes that require constant attentiveness as in driving, using machinery or having to do long-term computer-related tasks. Drowsiness is one of the most frequent dangers to safely in such cases as it slows down a person and makes his/her response slower. It has been found that many road accidents and injuries at work become possible because people do not observe the first symptoms of tiredness or micro-sleep attacks. Conventional methods of monitoring like manual monitoring or hardware based monitoring are not always accurate, costly or are inconvenient to operate in the real world. Consequently, computer vision based solutions which are able to automatically detect cues of tiredness on the face have attracted interest.

Developers can currently use a web camera to capture minor human behaviors such as eye movements, blinking frequency, opening of mouth, and gaze direction using technologies such as Mediapipe Face Mesh and OpenCV with recent improvements in machine learning and facial landmark detection. These facial clues include the Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR) which are good predictors of a person either being alert, drowsy, or having short moments of micro-sleep. This renders vision-based drowsiness detector to be a cost-effective, contact-free, and efficient method of constant surveillance.

The project aims at developing an automated drowsiness and micro-sleep recognition system that will run directly on a browser in Google Colab. The system uses images captured by the webcam of the user at regular intervals and using facial landmark algorithms to analyze the images instead of the user needing any special hardware. The system classifies the state of the individual as a state of Awake, Drowsy, Sleeping or Micro-Sleep based on the calculated values of EAR, MAR, frequency of blink, as well as gaze deviation. The system also gives a summary of the performance by keeping a running log of such classifications, which shows the overall alertness of the user throughout the monitoring session.

The history of this project is based on the growing necessity of the real-time fatigue detection in the domain of the intelligent transportation system, the safety monitoring systems in the workplace, and the human computer interactions. This work illustrates that a feasible and effective system of fatigue monitoring can be designed with the help of accessible tools and combining them with image processing and machine learning approaches. The aim is to minimize accidents, enhance awareness and help in the construction of safer human friendly environments.

1.1.1 Context and Relevance

Drowsiness detection can be part of a wider area of research that links computer vision and human-computer interaction with intelligent safety systems. In a large number of industries nowadays, particularly transportation and manufacturing, the aspect of alertness has been viewed as an important component of safety. Bus drivers, truck drivers, ride-sharing cars and even ordinary users of personal cars usually have to spend hours of their lives on the road. Fatigue has been identified as one of the most leading causes of road accidents across the world today and several organizations are currently making an attempt to institute real time monitoring systems to curb such threats. This has led to great need of technologies that can automatically comprehend human behavior without necessarily relying on expensive hardware.

Meanwhile, computer vision is evolving at a rather high pace. Tools that examine facial expressions, identify emotions, detect gestures and even monitor eye movements are analyzed with amazing precision using modern tools. Vision based monitoring is now more available than ever, since webcams and built-in laptops cameras have become so widespread. The developers and researchers are working on lightweight non-intrusive techniques that can be adapted to simple devices rather than the specialized sensors. That is the reason that EAR, MAR, and face- landmark-based techniques have already gained popularity they provide a feasible method of identifying fatigue only with software.

Nevertheless, there is one huge obstacle never before, the majority of real-time monitoring systems need complicated configurations or mighty hardware. These solutions are not accessible to many people due to their inaccessibility. The given project will fill that gap by fully executing in a browser using Google Colab, which is capable of drowsiness detection without installations or additional sensors or expensive computers. It opens up opportunities to the students, researchers, drivers, and other general users who require a simple and yet effective tool of fatigue monitoring.

The topicality of the research, in general, is its integration of safety needs, trends in technology, and the practical accessibility. Due to the move towards automation and smartest monitoring in industries, such a system can be of importance in accident prevention, enhancement of awareness, and safer working conditions.

1.1.2 Problem Identification

Drowsiness detection has been looked at for a time.. Most solutions still have some problems. A lot of systems use things, like EEG sensors or heart-rate monitors. These are devices that you have to wear on your body. They can give you information but they are not nice to wear every day. They are also expensive and hard to take care of. So people who drive or work every day do not use them because they want something simple. Drowsiness detection is something that people need. They need it to be easy to use.

Computer vision–based methods have also been explored in past studies. Many of these systems use heavy machine learning models that require high processing power, making them unsuitable for real-time use on normal laptops or mobile devices. Others need special setups, controlled lighting conditions, or multiple cameras to function properly. Some solutions depend on continuous video streaming, which can cause performance issues or increase the risk of privacy concerns. Because of these limitations, existing approaches are often not practical for general users.

1.1.3 Purpose and Justification

The primary goal of the proposed project is to develop a simple and accessible system that will be able to report any signs of drowsiness and micro-sleep at the mere usage of an ordinary web camera. In several cases, a lot of individuals encounter scenarios where their alertness level is very crucial like in driving, working overtime, studying late, or watching machines. Sleepiness during a few seconds is enough to create unsafe conditions, but the majority of the people are not even aware of the initial signs. This project will help to offer a practical tool that will be able to detect these signs and inform the user more about the level of his or her alertness.
This work has been justified by the need to have an easy to use solution that is accessible to all people, and not restricted to the ones who have special equipment. This project is not reliant on sensors, expensive equipment, or advanced hardware unlike many other complex systems and is run right in a browser with Google Colab. It can be run by anyone with a computer and a webcam, and does not require either installation or additional expenditure. This makes the system handy to students, drivers, researchers as well as common people who would prefer to know their fatigue levels easily.

Another value addition that is associated with the project is the integration of the familiar ideas such as EAR, MAR, blinking patterns and gaze behavior into a single system. It does not take only one indicator but considers several facial cues, and this enhances the accuracy.
of the results. Also, the system captures the frequency of various states (Awake, Drowsy, Sleeping, Micro-Sleep) so that users have a clearer image of how they are generally feeling during the monitoring.

All in all, the project is significant in that it unites the three topics of safety, convenience and modern computer-vision methods into a real-world practical tool. As it simplifies the process of drowsiness detection and makes it more convenient and available, it helps in safer driving, accident prevention, and makes people more aware of their physical state.

1.1.4 Scope

In this project, the proposed work is devoted to the creation of an easy and convenient system, which will be able to identify various degrees of drowsiness with the help of a regular web camera only. The concept of the work is the method of computer-vision applications, namely the analysis of facial landmarks with Mediapipe Face Mesh and OpenCV, to comprehend the changes in the eyes, mouth, and direction of gaze of a person. The system is able to detect the state of the user by computing such values as the Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR) to determine that they are awake, getting drowsy, fully asleep, or experiencing a small micro-sleep episode.

This study has a number of areas that it encompasses. First, it involves taking photos at a certain time interval with browser-based camera technology of Google Colab. Second, it entails the processing of these images in order to detect the facial landmarks and convert them into meaningful measurements. Third, the project includes the creation of a classification mechanism according to which the level of alertness of the user is determined on the basis of the persisting features calculated. The system also records the frequency of occurrence of each state where the users will get a clear overview of their general condition throughout the monitoring session.

The project however is subjected to periodic frame analysis as opposed to real- time monitoring. It is not trained on sophisticated deep-learning models, external sensors, and huge training data. Other factors like lighting conditions, head movement and camera quality can have an effect on the detection accuracy but they are not the main subject of the project.

In general, the work area is the presentation of a lightweight and software-only solution that illustrates the possibility of tracking drowsiness by using facial analysis. It preconditions the further development of more sophisticated systems but is simple to use and available to anyone who has basic computer and web-camera.

1.2 Project Planning and Initiation

Feasibility Study (Step-by-Step).

Feasibility study assists in ascertaining the development success of the project and with the tools, time and resources available. In this drowsiness-detection system, the technical, operational and economic areas are the major feasibility areas. The step wise study has been described below.

Step 1: Technical Feasibility

The initial process involved finding out whether the project was possible to be constructed using the tools at hand. Mediapipe Face Mesh and OpenCV can be easily implemented in Google Colab without any powerful computer. Images can be taken with a normal laptop webcam and no more advanced hardware is needed in EAR/MAR calculations. The system can technically be running with minimum set up since it works on all Python and allows access to the cameras via a browser.

Step 2: Operational Feasibility.

The second step was whether the system can be operated smoothly and fulfil its purpose. The project can be easily used since the user will only have to run Colab notebook. The camera automatically records frames, identifies face markers and determines the level of alertness of the user. No previous technical-technical knowledge or external apparatus is needed. This renders it useful to the common user, students or drivers who are just interested in a simple fatigue-monitoring device.

Step 3: Economic Feasibility

The project does not involve any money investment. Python, Google Colab, Mediapipe and OpenCV are all free to use. It does not require any special sensors or devices. This renders the system economical and research and teaching applicable.

Step 4: Time Feasibility

The techniques applied in this project, including EAR/MAR calculation and facial-landmark detection are documented and easily implemented. The system requires only a manageable time to develop since the whole system uses pre-built libraries. It is also less complex to capture at intervals rather than in real-time streaming.

Step 5: Behavior/Usage Feasibility.

Lastly, the system is user friendly. It does not involve donning a gadget or setting up programs. The user only has to grant camera authorization and the program does the rest. This is more likely to make sure that people actually use the tool when they need it.

The feasibility study reveals that the project is feasible, low cost, technically feasible and users can easily adopt it. The system can be successfully designed and tried within the normal project requirements due to the freely available tools and a simple workflow.

Phase 1: Background Analysis and Project Scope.

The initial stage of the project is dedicated to the familiarity with the general problem, the analysis of the possible solutions, and proper definition of what the system will and will not entail. The key objective of this stage is to research the influence of drowsiness on human behavior and how computer-vision devices can be utilized to identify the early signs of fatigue. This stage is also useful in determining the key features that need to be established in order to create a practical system that is easy to use.

In the preliminary analysis, various methods of measuring drowsiness were studied. The conventional ones included sensor-based detection (EEG, heart-rate sensors or wearables) which proved to be correct and yet not applicable to a regular user due to their inconvenience and high cost. It resulted in a search to find a more accessible method of facial-landmark based detection with the help of a normal web camera. Software such as Mediapipe Face Mesh and OpenCV were tested to ascertain whether they could be able to track the movements of eyes and mouth reliably as these are prime indicators of alertness.

When the technical options had been realized, the project scope was determined. The system would be devoted to the Eye Aspect Ratio (EAR), Mouth Aspect ratio (MAR), the behavior of blinking, the direction of gaze, and classify a user into several states: Awake, Drowsy, Sleeping, and Micro-Sleep. It was also ruled that the system would be executed within Google Colab in which the camera of the browser would take frames on scheduled time. This renders the project usable by anyone having a basic computer without any installations or other special hardware.

The project is also constrained by the scope in that it does not support real time video streaming every minute but only periodically. The external elements, such as lighting and rapid head movement, are not part of the central concerns of the project. The main goal is to show a low-cost and simple, yet effective, piece of evidence of fatigue detection using software only.

In general, Phase 1 guarantees a clear direction of the project, goals which can be attained and clear boundary. It prepares the groundwork to the subsequent stages, which includes designing, implementing and testing the system.

Phase 2: Market Research (or Market Feasibility Analysis)

During this step, it is aimed at knowing whether there is a tangible need or demand of a drowsiness-detection system in the existing market. The analysis is aimed at determining the possible users, solutions available and gaps that this project can address. This stage is useful to justify the reason why the project is worth developing by researching the trends in the industry, understanding the user needs and the technology available.

Market is becoming more concerned about driver safety, particularly in the transportation industries like trucking, ride-sharing, buses and logistics. The accidents happen many times due to the unintentional falling asleep or loss of focus on the part of the drivers. Business organisations and fleet operators are currently seeking cheaper monitoring applications that can notify users about the initial symptoms of fatigue. The majority of the commercial systems available, however, use costly hardware, specific cameras, or embedded vehicle sensors so are not accessible to students, small businesses, or even individuals.

Other areas are also interested in drowsiness monitoring on other than transportation. Simple fatigue-tracking tool can be useful to people who work at night shifts, in hospitals, and security officers, factory workers, and even students who study long hours. This provides a broad possible user base of a lightweight web-based system that uses a webcam.

In the course of the research, it was discovered that a lot of the existing software solutions are either real-time video processing or they require high-end computers to operate. There are mobile applications that are trying to detect drowsiness but in most cases, they lack accuracy because of poor placement of the camera. Otherwise, this project applies Mediapipe Face Mesh which is recognized to be reliable and Google Colab, which makes the system accessible without installations. This makes the project a low-cost and easy to use alternative to the current tools.

The analysis indicates that there is a great market opportunity of a hardware independent, low-cost, and browser-based drowsiness detector system. Because the project just requires a webcam and internet connection, it is more attractive to a larger group of people such as researchers, students, drivers, and organizations that want to find an easy monitoring solution. This step will ensure that the project holds valuable real-life utility and that it could be integrated into a market that is expanding in which safety and awareness are becoming more valuable.

Phase 3: Technical Feasibility Analysis.

The technical feasibility analysis is aimed at considering the possibility whether the drowsiness-detection system proposed can be developed successfully within the utilization of technologies, tools, and resources available. This step investigates the technical needs of the project, the constraints of the tools used and whether the system can operate effectively in the planned system.

The initial aspect of the analysis was to find out whether or not real-time detection of facial landmarks could be achieved using conventional consumer hardware. The project is based on Google Colab, though they do not use high-performance GPUs or specialized sensors, as Google Colab allows accessing the camera via the web browser. The fact that Colab supports Python code, and does not need any local installations, erases hardware dependency. Such tools as Mediapipe Face Mesh and OpenCV are lightweight and can work effectively in this setting. It was tested that Mediapipe has a high reliability rate when detecting up to 468 facial landmarks on even mid-range laptops, which proves that the technology selected is appropriate in this project.

The second in line was to test how the system was able to compute the EAR (Eye Aspect Ratio), MAR (Mouth Aspect Ratio), and gaze deviations with respect to the extracted landmarks with accuracy. These techniques involve simple mathematical operations of pixel positions, that are not dependent on complex machine-learning learning or extensive processing. This renders the computation burden very light. Unlike video streaming, the system captures images at regular time intervals which also lowers the processing requirements and makes the system mostly stable in Colab.

The other technical factor was the camera access, which was another important factor. Although Colab has never been intended to be used as a continuous video capturing tool, JavaScript can be used to enable the notebook to make automatic snapshots using the browser. This approach is always compatible with a majority of devices, and it proves that frame-by-frame monitoring can be implemented in a technical manner without a special application or local software.

Lastly, the system was assessed in various circumstances including alteration of lighting, movement of the head or quality of the webcam. These limitations, however, can tolerate a small amount of accuracy deviation, which is acceptable by a proof-of-concept system.

Phase 4: Financial Analysis of Feasibility.

The financial feasibility analysis is a study that determines how the project can be developed and utilized without major cost. The aim of ensuring financial requirements were minimized was one of the key objectives in this case since this system would be simple, accessible and affordable. Having examined all the specifics, one will come to the conclusion that the project is highly cost-effective and can be constructed with nearly no financial investments.

All the core technologies involved in the project Google Colab, Python, Mediapipe and OpenCV are fully free and open-source. One does not need to buy any.

commercial software, licenses or development tools. Colab is fully web based i.e. users do not have to acquire costly hardware or even install extra software. The entire system can be run on a regular laptop having an inbuilt web-cam.

The hardware spec is also very low because the system does not rely on external sensors such as EEG instrument, infrared, cameras, or specialized eye-tracking hardware. Most of the available drowsiness-detection techniques are based on the use of that equipment, which is prohibitively expensive when considering students or small firms. This project does not incur such costs by simply using a normal webcam.

No paid services are also required in testing and experimentation. Frames of live webcams are used to retrieve data rather than using pre-recorded data or commercial cloud computing. There is no operational cost even when the system is used over time since Colab offers free time of usage of GPU/CPU and automatic access to cameras.

The financial burden of the implementation and maintenance is very low under terms of implementation and maintenance. No recurring charge, subscription or hardware upgrade is needed. This contributes to the fact that the system is not only cheap to construct, but also sustainable to be used in long-term in academic, research, or personal settings.

In general, the financial viability analysis indicates clearly that the project is very cost-efficient. The entire system can be created and used at almost no cost since free tools are used, no special equipment is required, and little hardware is needed, which is affordable to students, researchers, and regular users.

1.3 Intended User Profile and Provisional Elicitation Process.

The system is aimed at a large audience of users who might require assistance in controlling their level of alertness or avoiding fatigue. The most important target users and stakeholders are:

1. Drivers (Professional and Private)

Drivers who spend a lot of time on the road like taxi drivers, bus operators, truck drivers and delivery workers who should have early warning systems to prevent accident cases due to drowsiness impairments.

2. Students and Learners
Individuals who read or do study taking excessive time and more so at late-night hours and may end up sleepy taking work.

3. Office Employees and Computer users.
Individuals who have been spending a lot of time before computers and wish to be more focused and productive.

4. Night Shift, Rotational Shift Employees.
Hospital, factory, security service, and call center workers usually have a high fatigue rate because of irregular sleeping patterns.

5. Researchers and Developers
People, whose interests lie in computer-vision technologies, the analysis of human behavior, or the safety-related applications.

1.3.1 User Profile

Table 1.1: User Profile for Drowsiness & Micro-Sleep Detection System

User Class	Note on Characteristics
Type of User	Drivers, students, office workers, shift employees, general computer users
Age Range	Typically 16–60 years (anyone who regularly works, studies, or drives)
Frequency of Use	Moderate to frequent; users may run the system during long working or driving sessions
Mandatory	Not mandatory; system is optional and used for personal safety and awareness
Computer Experience	Basic computer knowledge is enough (ability to use a browser and camera permission)
Education	No specific education level required; suitable for students, workers, and professionals
Goal	To monitor alertness, detect early signs of drowsiness, and avoid loss of focus or micro-sleep
Language Skills	Basic English is helpful for understanding interface messages, but not mandatory
Number of Users	Can be used by individuals; scalable for organizations with multiple users
Training	Minimal to no training required; simple one-click operation through Google Colab
Other System Use	Users may simultaneously use navigation tools, study apps, office software, or monitoring dashboards
Way of Working	The user remains in front of a webcam; the system captures periodic frames and evaluates alertness

1.3.2 Elicitation Process

A systematic requirement gathering process was adhered to to know what the users really require of the Drowsiness and Micro-Sleep Detection System. Opinions, expectations and practical concerns of the potential users and stakeholders were gathered using different methods of elicitation. The primary methods are described below:

Interviews

Students, drivers, and office workers were interviewed (in short and, not very formal ways) to learn about their experience in fatigue and the problems they have when working overtime. Among the important user expectations that were elicited through the interviews include simplicity, limited hardware requirements, and real-time alertness feedback. The users also told about their experience with other existing systems and why many of them are not convenient, or cost a lot.

Online Surveys

A basic survey was distributed amongst students and shift employees to obtain a wider input. The questions in the survey were on work habits, average screen time, frequency of driving, and the interest of the users in a fatigue detection webcam tool. The information obtained in the surveys was used to give precedence to the features that the users valued the most like in the detection of blinking, analysis of yawning and easy accessibility using a browser.

Observation

Simple observation of users sitting at computers or studying over prolonged periods gave an understanding of natural fatigue behavior including prolonged blink, large yawns, or eye positioning. This contributed to developing the EAR/MAR threshold values and confirmed that monitoring by using webcams could be used to capture the relevant behavior.

Competitor Analysis (Secondary Research)

Communicated systems and tools of drowsiness detection tools and existing systems were studied to get an idea about their advantages and disadvantages. Most of them were very expensive to deploy as hardware or were constantly processing video, which validated the need of a more basic and accessible solution.

Feedback Sessions

The system was subsequently tested by a small group of users in Google Colab after an early prototype was created. Their comments were used to correct the type of alerts and enhance clarity of the status messages as well as change the frame-capture interval.

The elicitation process involved a combination of interviews, surveys, observation, competition research and prototype feedback to elicit the user expectations. The approaches made sure that the end product system is friendly, light and that it can be used in day-to-day operations.

1.3.3 Drowsiness Detection Workflow Diagram

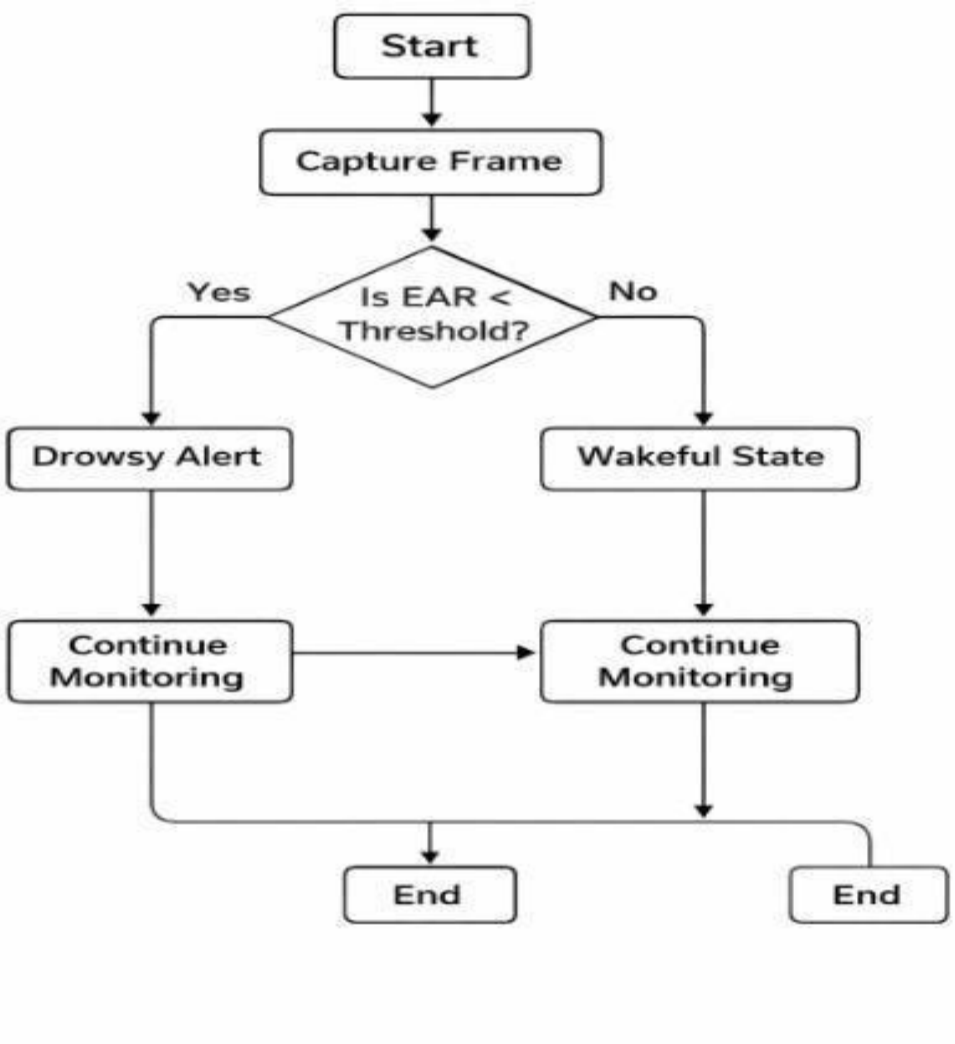


Figure 1.1: Drowsiness Detection Workflow Diagram

1.4 System Requirements

The recommended Webcam-Based Real-Time Fatigue Detection Model Using Facial Landmarks is programmed in Google Colab, therefore, it does not need advanced hardware and software to operate. To run the Colab runtime a user requires a standard computer or laptop with a two-core processor (2.0 GHz or above), a minimum of 4 GB RAM, an inherent or external webcam (at least 720 p), and a reliable internet connection: to have a smoother experience with real-time, an i5/Ryzen-5 processor, 8 GB RAM, and 1080 p webcam are suggested. Software-wise, any current operating system (Windows, Linux, macOS) that has a current browser like Chrome/Edge/Firefox works, as all code can be run in the built-in Python 3 interpreter of Colab with libraries like MediaPipe, OpenCV, NumPy, and any supplementary modules. The major limitations are the sensitivity to low lights, one-face detection configuration, Colab session/time constraints, and reliance on fixed EAR/MAR thresholds, whereas extrinsic ones are associated with the browser webcam permission, constant internet connectivity, and consistent MediaPipe Face Mesh landmark tracking.

1.4.1 Hardware Requirements

To run the Drowsiness & Micro-Sleep Detection System smoothly, the hardware requirements are kept minimal so that any regular user can operate it without special equipment. Since the system is designed to work through Google Colab and uses a webcam for image capture, only basic computer hardware is needed. The required specifications are listed below:

Minimum Hardware Requirements

- **Computer / Laptop**
  A basic laptop or desktop computer capable of running a web browser.
- **Processor**
  Dual-Core CPU (Intel Core i3 / AMD equivalent or higher)
- **RAM**
  Minimum 4 GB RAM to ensure smooth performance during image processing.
- **Webcam**
  Any built-in or external HD webcam (720p or higher preferred) for capturing clear facial images.
- **Storage**
  At least 500 MB free space for temporary file storage, cache, and Colab downloads.
- **Internet Connection**
  Stable broadband connection (because Google Colab runs online and requires camera permission through the browser).

Recommended Hardware Requirements (for better performance)

- **Processor**
  Intel Core i5 or higher for faster execution of image-processing tasks.
- **RAM**
  8 GB RAM or more to handle multitasking while running Google Colab.
- **Webcam**
  Full HD (1080p) camera for better landmark detection accuracy.
- **Monitor**
  14-inch or larger screen for better visibility of real-time outputs and logs.

No Special Hardware Needed

This project does **not** require:

- External sensors
- EEG or heart-rate devices
- Infrared cameras
- GPU or high-performance machine

The entire system runs using just a normal laptop + webcam.

1.4.2 Software Requirements

To run the Drowsiness & Micro-Sleep Detection System, only a few software components are required. The entire system is designed to be lightweight and browser- based, so users do not need to install complex applications or development tools. The essential software requirements are listed below:

Operating System

- Windows 10 / 11
- macOS
- Linux (Ubuntu recommended)
- Any OS that supports a modern web browser will work.

Web Browser

A latest-version browser with webcam access capability:

- Google Chrome (Recommended)
- Mozilla Firefox
- Microsoft Edge
- Safari

Browser must support:

- JavaScript execution
- Camera permission handling
- Google Colab compatibility

Google Colab

The project runs entirely on **Google Colab**, which provides:

- Python execution environment
- GPU/CPU runtime (optional)
- Browser-based camera access
- No installation required

Python Environment (Colab Built-In)

Python 3.x with the following libraries (installed inside Colab):

- OpenCV (opencv-python)
- Mediapipe
- NumPy
- Base64
- Time
- JavaScript interface (Colab's eval_js)

These libraries handle:

- Image processing
- Facial landmark detection
- Mathematical calculations (EAR, MAR)
- Frame capture
- Data handling and output logs

Additional Tools

- Google Drive (optional) for saving logs or outputs
- Notebook Viewer for editing or reviewing the project

No Local Installation Needed

Because everything runs online:

- No need to install Python on the local machine
- No IDE required
- No additional drivers or SDKs needed

Constraints and Dependencies

This project, although lightweight and easy to use, operates within certain limits and depends on a few external factors. These constraints and dependencies help define what the system can and cannot do and highlight the conditions required for proper functioning.

1.4.3 Constraints (Limitations)

Lighting Conditions

The system relies on webcam images, so poor lighting, shadows, or excessive brightness may interfere with facial-landmark detection accuracy.

Camera Quality

Low-resolution or blurry webcams might fail to capture clear facial features, which can affect EAR, MAR, and gaze calculations.

Internet Connection

Since the entire system runs on Google Colab, a stable internet connection is required. Slow or unstable internet may interrupt camera access or delay frame processing.

Non-Real-Time Processing

The system captures frames at intervals (not continuous streaming), so it may not detect very fast blinking or extremely short micro-sleep episodes.

Head Position and Movement

Extreme head angles, frequent movement, or partially hidden faces can cause landmark tracking to break or deliver inaccurate results.

Browser Permission Issues

The webcam can only be accessed if the user grants camera permission. Any browser restriction or blocked permission will stop the system from working.

Environmental Factors

Glasses, hats, masks, or face coverings may reduce landmark detection performance

1.4.4 Dependencies (External

Requirements) Google Colab Platform

The system depends entirely on Google Colab to run Python code, execute JavaScript for camera access, and process images.

Web Browser Compatibility

A modern browser (Chrome recommended) is required to support JavaScript and webcam access.

Mediapipe Library

Facial landmarks are detected using Mediapipe Face Mesh. Without it, EAR/MAR calculation is not possible.

OpenCV & NumPy

These libraries handle image processing and mathematical operations. The system relies on them for frame reading, pixel conversion, and coordinate calculations.

Webcam Hardware

A functioning built-in or external webcam is necessary to capture the user's face.

Python Runtime

The whole system is based on Python 3.x and presence of necessary libraries within it.
Colab.
The system is easy and usable, and requires the use of external applications such as Google Colab, the internet, and a working webcam of the user. It also possesses natural limitations pertaining to lighting,facial visibility, and image quality. In spite of these shortcomings, the system is still efficient.to demonstrate, study and to use personally.

1.5 Project Scheduling

The schedule helps to make the project run in an uninterrupted process completion. The work was divided into stages that have a certain time and a fixed number of stages activities. This is a timetable plan that assists in controlling resources, monitoring progress and minimizing delays.

1.5.1 Time Frame

The whole project took place over a few months, passing through planning design, development and testing. All stages were assigned enough time to make sure that it was high quality output.

The project schedule and the length of the project phases was portrayed using a Gantt chart. It presents the initial date of every task and the final date of every task clearly as well as the overlapping activities.

Table 1.2:Project Time Frame Overview

Phase	Duration	Timeline
Phase 1: Planning	2 weeks	February (Week 1 – Week 2)
Phase 2: Feasibility Study	3 weeks	February (Week 3) – March (Week 1)
Phase 3: System Design	4 weeks	March (Week 2 – Week 4)
Phase 4: Development	6 weeks	April – May (Week 4)
Phase 5: Testing	3 weeks	June (Week 1 – Week 3)
Phase 6: Deployment & Review	2 weeks	June (Week 4) – July (Week 1)

This timeline follows a logical flow and ensures adequate time for implementati

Gantt Chart (Description)

The project schedule and the length of the project phases was portrayed using a Gantt chart. It presents the initial date of every task and the final date of every task clearly as well as the overlapping activities.

Gantt Chart Summary

Planning begins at the beginning of February and ends mid-February.

- Feasibility Study commences immediately after the planning and overlaps till the beginning of March.
- System Design March- early April.
- The most prolonged stage is development (April to end of May) and it involves coding, test EAR/MAR algorithms, Mediapipe integration, and frame capture configuration.
- Testing takes place during the course of June and bugs are located and the accuracy enhanced.
- The deployment occurs in the end of June and early July and then the documentation of the project.

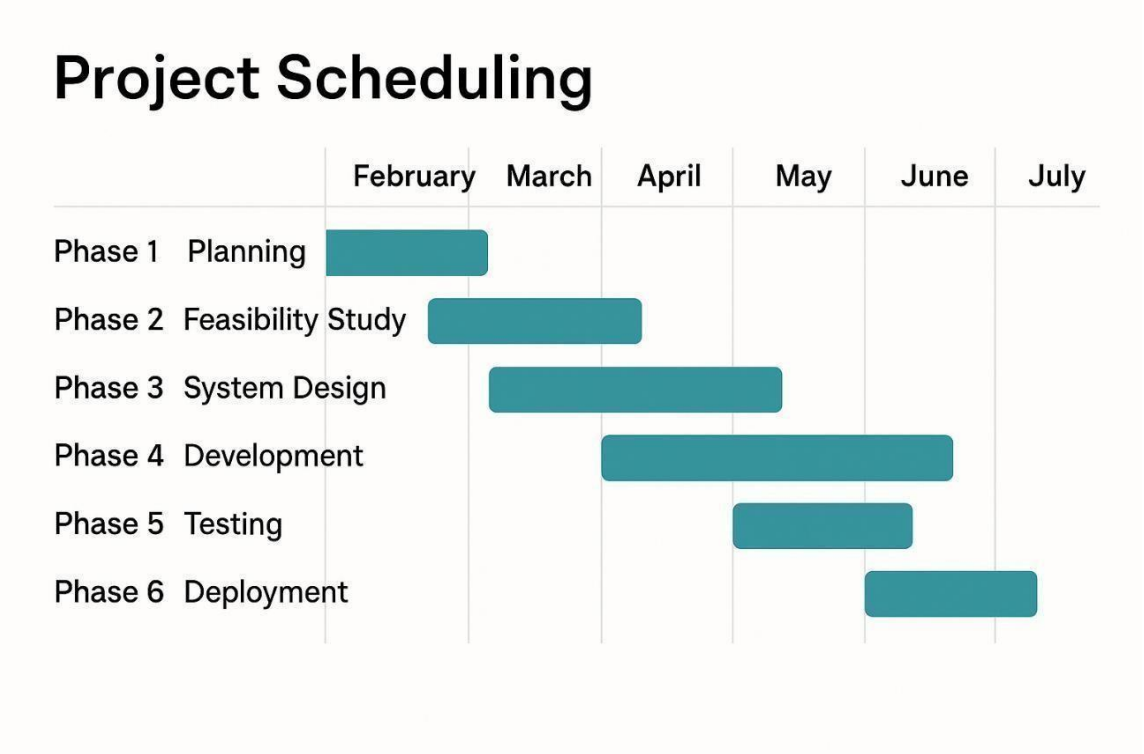


Figure 1. 2:Project Scheduling

Risk Management

Risk management assists in establishing the potential obstacles in the course of the project and offers the means of mitigating them. As the system requires input of webcams, access to browser and Google Colab, various technical and environmental hazards have been taken into account.

Risk Factor	Impact	Likelihood	Mitigation Strategy
Poor lighting conditions	Wrong landmark detection	Medium	Use the system in bright, uniform lighting
Low-quality webcam	EAR/MAR accuracy drops	Medium	Recommend minimum 720p resolution camera
Unstable internet	Camera access fails in Colab	High	Use a strong broadband connection
Browser blocking camera permission	System cannot run	Medium	Provide instructions to enable camera access
Colab session timeout	Work interrupted	High	Save logs and refresh session when needed
Face not centered in frame	Incorrect detection	High	Show guidelines for proper webcam positioning
User wearing glasses/mask	Landmark detection unstable	Medium	Request user to remove reflective glasses
Thresholds not suitable for all faces	False positives/negatives	Low	Allow manual tuning of EAR & MAR thresholds

Table 1.3: Risk Table

The project plan is an amalgamation of a clear time schedule, designed Gantt chart, and a detailed risk management plan. This makes the project effective in executing the project with minimal delays and in a manner that it will meet technical issues.

1.6 Chapter Summary

The basic framework of the project was outlined in this chapter in terms of the background of the project, its purpose, and its relevance. It identified the major issues associated with the issue of human drowsiness and the importance of having a ready and web-based detection system. The scope of the project, its feasibility, its users were also identified in the chapter as well as the process of collecting the requirements. Hardware needs, software needs were also discussed, and the key constraints and dependencies that can influence the performance were addressed. Lastly, the project scheduling document showed the project schedule, Gantt chart synopsis, and the critical risks and mitigation measures. Generally, this chapter gives a proper knowledge of where the project is going, its goals, and plans before getting into the technical design and implementation phase.

CHAPTER 2

DESIGN AND IMPLEMENTATION

2.1 Introduction

This chapter specifies how the drowsiness detection system was developed, and how every aspect was addressed. Once the project requirements have been identified in the last chapter, the next thing to do is to translate the ideas into a working model. This involves the choice of appropriate tools, organization of the system process, and the construction of the logic that should analyze facial features and identify any fatigue symptoms.

The chapter outlines the general system architecture, the sequence of operation, and key used algorithms, including: EAR (Eye Aspect Ratio), MAR (Mouth Aspect Ratio), detection of blinking and gaze analysis. It further describes the process of collaborators such as Google Colab, Mediapipe and OpenCV in capturing frames, processing images and producing output. Lastly, the implementation details provide information on how the various modules were created, integrated and tested to create a functional drowsiness detection system.

This introduction helps the reader get ready to expect an in-depth tour of the structure of the system, design choices, and code that is the basis of project functionality.

2.2 Functional Requirements

Table 2.1: Functional Requirements

FR01	**Capture Frame**
Description	The system must automatically capture frames from the webcam at fixed time intervals.
Stakeholder	User, System Operator

FR02	**Detect Face Landmarks**
Description	The system must detect facial landmarks using Mediapipe Face Mesh before calculating EAR and MAR.
Stakeholder	User, Developer

FR04	Classify User State
Description	The system must classify the user as Awake, Drowsy, Sleeping, or Micro-Sleep based on EAR/MAR and blink count.
Stakeholder	User, System Administrator

FR03	Calculate EAR & MAR
Description	The system must calculate Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR) from detected facial landmark points.
Stakeholder	Developer, User

FR05	Generate Alerts
Description	When the user becomes Drowsy / Sleeping / Micro-Sleep, the system must display a warning alert.
Stakeholder	User

FR06	Gaze Detection
Description	The system must analyze gaze direction and detect when the user is not looking forward.
Stakeholder	User, Developer

FR07	Status Logging
Description	The system must record how many times each detection state occurred during monitoring.
Stakeholder	User, Evaluator

FR08	Session Summary
Description	At the end of the session, the system must display a summary of Awake, Drowsy, Sleeping, and Micro-Sleep counts.
Stakeholder	User, Project Supervisor

2.3 Non-Functional Requirements

The non-functional requirements are those that reflect the quality, performance and general behavior of the system as opposed to the tasks that the system undertakes. Whereas functional requirements state what should be done by the drowsiness detection system, non-functional ones are concerned about the way the system ought to work. These conditions make the system reliable, efficient, user friendly and relevant to the environment of the users. The non-functional requirements that will be applied to this project are performance speed, precision of EAR/MAR calculations, ease of use, and system reliability in a variety of lighting conditions as well as compatibility with Google Colab and modern web browsers. Such limitations are used to keep the operation flowing and enhance the overall user experience.

2.3.1 Performance

One of the most crucial non- functional aspects of the drowsiness detection system is the performance. This system should be able to process every frame that is captured in a manner that the user can be able to get the related feedback in time. Because the system records frames in regular intervals (e.g. 5 seconds) then it must be capable of detecting landmarks, computing EAR/MAR and classifying in that frame without introducing delays.

Mediapipe Face Mesh can be designed to be used in real-time, so one should be able to identify facial landmarks in the system using computers of medium processing power. The performance requirement makes sure that the calculations are not frozen, lagging, or slowing down the runtime of Google Colab. Only CPU resources should be used in running the system because users might not have the support of the GPU.
The system should also be stable when it is performing continuous frame analysis. No matter the duration of the monitoring session, the performance must not decrease and instead, the leakage of memory or unexpected crashes should not be experienced. The model is expected to act fast in processing the frames even with changes in lighting or camera quality without needless computation.

All in all, the performance requirement guarantees that the system is quick, has low resource consumption, low frame processing, and is smooth during the monitoring session.

2.3.2 Reliability

Depending on the normal usage, the reliability of the drowsiness detection system can be determined as the consistency and reliability of the system. Facial landmarks and the calculation of EAR/MAR values should be recognized by the system, and the user states should be classified with a low frequency of errors or failures. It should keep on going when the user is slightly moving, changing the light settings, and even having a poor-quality webcam.

A robust system must also react appropriately to any situation that is not anticipated such as the absence of any face in which the system must present an adequate message rather than crash. Similarly, in case the network connection has gone dead, or the Colab session has been refreshed, the system ought to be capable of starting once again with minimal user intervention.

Reliability is also an aspect whereby the system ought to give consistent and repeatable results. The equivalent drowsiness is supposed to yield the equivalent categorization on several tries. EAR/MAR thresholds must be reliable when operating between frames and with different users, they should not be erratic and avoid unexpected situations.

In general, the system should be able to run full time and gracefully with limited interruptions as well as provide reliable output during the monitoring time.

2.3.3 Portability

Portability is the degree to which the system can execute in a variety of devices, operating systems and environments without the need of significant adjustments. Being designed to run on Google Colab, this drowsiness detection system is also highly portable and can be applied to virtually any modern computer, because it is designed to use a web browser as the means of accessing the camera.

The system is not based on particular hardware setups or on special equipment. The minimum computer required to run the entire project is a basic laptop/desktop that has an inbuilt webcam. Since the processing is done in the cloud of Colab, the machine of the user is not required to have a high-performance CPU or GPU. This renders this system available to the students, researchers and ordinary users who might lack the costly equipment.

Platform independence is also a type of portability. The system is compatible with Windows, macOS, Linux, and even ChromeOS provided that the device has a browser, such as Chrome or Firefox. No Python, libraries or IDEs have to be installed and this makes it highly usable by various systems. All the user needs to do is to open the Colab notebook, authorize the camera and open the program.

The portability requirement also takes care of the fact that the system may be moved or re-used in other settings like the university laboratories, or personal laptops or office computers, without compatibility problems. The whole project is easily viewable as a Colab link; therefore, the user does not have to download and install anything locally. It is very flexible and convenient in demonstrations, testing, and real-life application making the system very flexible.

All in all, portability means that the system is accessible, easy to execute and easy to share and can have a great variety of supporting devices and operating environments with little configuration.

2.4 Object-oriented System design using UML

In this project, object-oriented system design was employed to ensure that the webcam-based fatigue detection system is structured into easily reusable modules, with UML diagrams employed to graphically represent these modules and their interactions. The model is designed as the following objects: frame capture, face landmark detection, EAR/MAR computation, fatigue state classification, summary reporting, which play a singular duty and communicate with others to ensure real-time monitoring in Google Colab.

2.4.1 Use Case Diagram

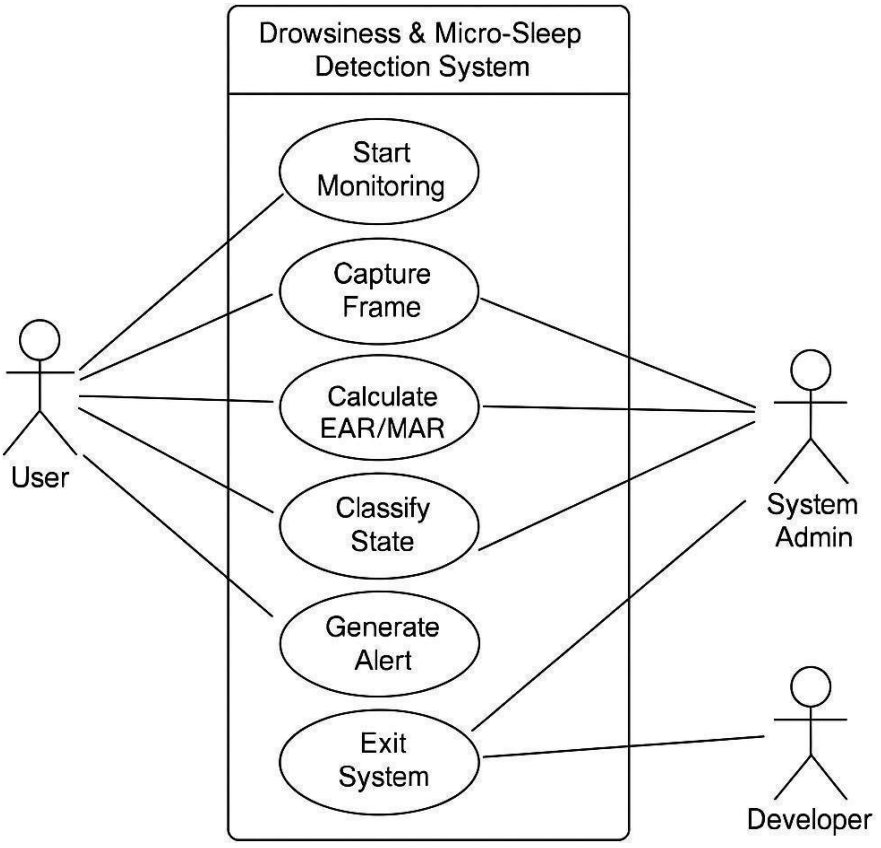


Figure 2.1: Use Case Diagram

2.4.2 Test Case Description

Use Case	Case Descriptol-1 State
Goal	Users can classify state basenad on the deteceteid landmarks
Precondition	Detect landmarks in the image frame
Success End Condition	Notification: State Classified!!!
	Notification: State *Not* Classified
Primary Actors:	System
Secondary Actors:	System
Trigger	User initiates state classification process
Description / Main Success Scenario	1. Calculate EAR/MAR values 2. Determine state using thresholds 3. Show classification result on the interface 4. State Classified notification displayed 5. Log classification result 6. System notifies user with state Classified!!!

Figure 2.2: Case Description

2.4.3 Activity Diagram

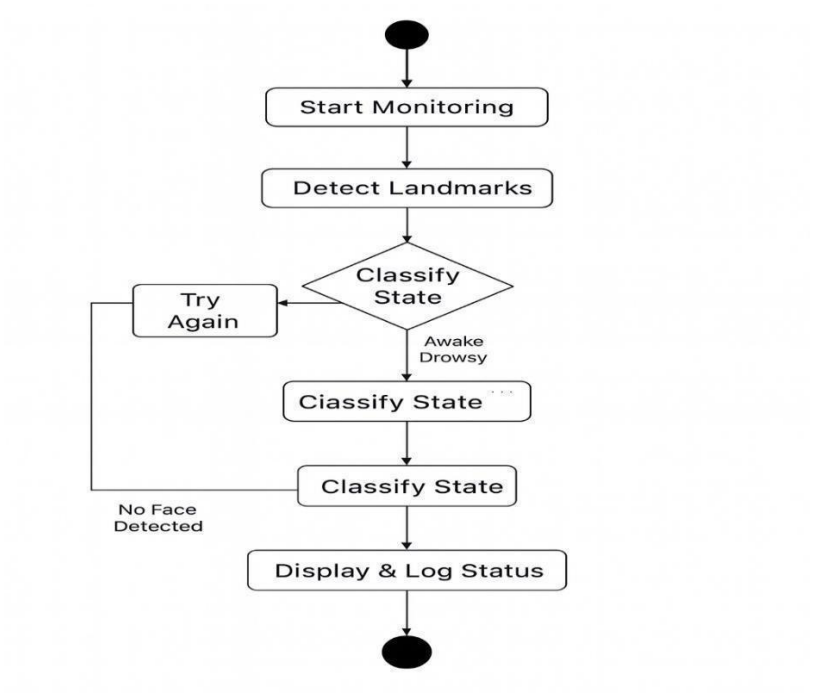


Figure 2.3: Activity Diagram

2.4.4 Activity Diagram for Drowsiness Detection

a) Textual Description (so it aligns with Use Case & Case Description)

1. **Start**
   User starts monitoring from the Colab notebook (matches *Trigger* in case description).
2. **Start Monitoring**
   System initializes webcam and Mediapipe model (*Precondition satisfied*).
3. **Capture Frame**
   System automatically captures a frame from the webcam.
4. **Detect Face Landmarks**
   System runs Mediapipe Face Mesh on the frame to detect eyes, mouth, nose.
5. **Decision: Face Detected?**
   - **No** →
     - Activity: Show "No Face Detected" / Try Again
     - Flow goes back to Capture Frame (loop, same as retry in case description).
   - **Yes** → go to next step.
6. **Calculate EAR & MAR**
   System computes Eye Aspect Ratio and Mouth Aspect Ratio from the landmarks.

7. **Decision: Classify State**
   Using EAR, MAR, blink count and gaze:
   - [EAR & MAR normal] → "Awake"
   - [EAR low / MAR high] → "Drowsy"
   - [Eyes closed long] → "Sleeping / Micro-Sleep"
8. **Generate Alert (if required)**
   If state = Drowsy / Sleeping / Micro-Sleep → show warning message.
   If state = Awake → no alert.
9. **Update Status Log**
   System increases the counter for the detected state (Awake / Drowsy / Sleeping / Micro-Sleep).
10. **Display Result**
    Current state and debug info (EAR, MAR) shown in output.
11. **Loop or End**
    If monitoring continues → flow goes back to Capture Frame. If user stops monitoring → go to End node.
12. **End**

b) How it aligns with your UML & Case Description

- **Use Cases in UML**:
  - *Start Monitoring, Capture Frame, Detect Face Landmarks, Calculate EAR/MAR, Classify State, Generate Alert, Record Status Log, Show Session Summary*
- **Activities above** follow exactly the same order and names.
- **Case Description-01: Classify State** steps:
  - capture frame → detect landmarks → calculate EAR/MAR → classify state → display + log → (loop)
- These are directly mapped to Activities 3–10 in the diagram.

2.4.5 Sequence Diagram

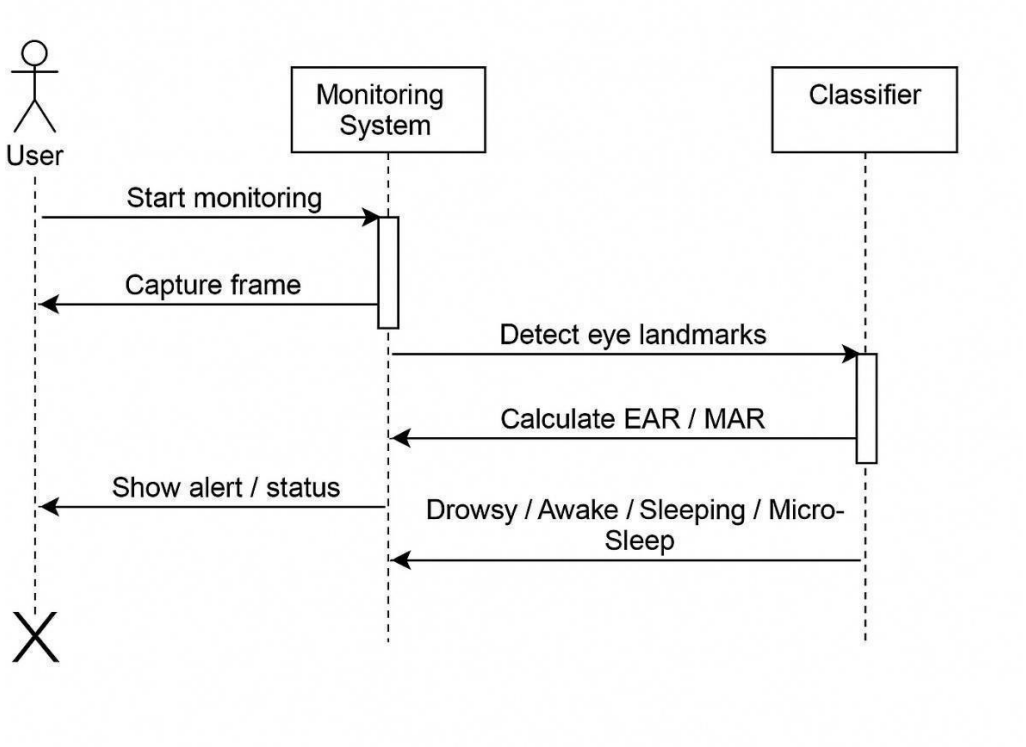


Figure 2.4: Sequence Diagram

The Activity Diagram and the Sequence Diagram of the drowsiness detection system describe the same "monitoring and classification" process from two different views. In the Activity Diagram, the focus is on the flow of actions, while in the Sequence Diagram the focus is on the interaction between the user and system objects.

Activity Diagram and Sequence Diagram

The mapping is as follows:

Table 2.2: Activity Diagram and Sequence Diagram

Activity Diagram (Activities)	Sequence Diagram (Messages / Interactions)
Start Monitoring	User → Monitoring System: *Start monitoring*
Capture Frame	Monitoring System → Camera/Module: *Capture frame*
Detect Face Landmarks	Monitoring System → Landmark Detector: *Detect eye landmarks*
Calculate EAR & MAR	Landmark Detector → Classifier: *Calculate EAR / MAR*
Classify State (Awake / Drowsy / etc.)	Classifier → Monitoring System: *Drowsy / Awake / Sleeping / Micro-Sleep*
Generate Alert & Update Log	Monitoring System → User: *Show alert / status*
Loop to Capture Next Frame / End Monitoring	Repeated message sequence until the user stops monitoring

2.4.6 Class Diagram

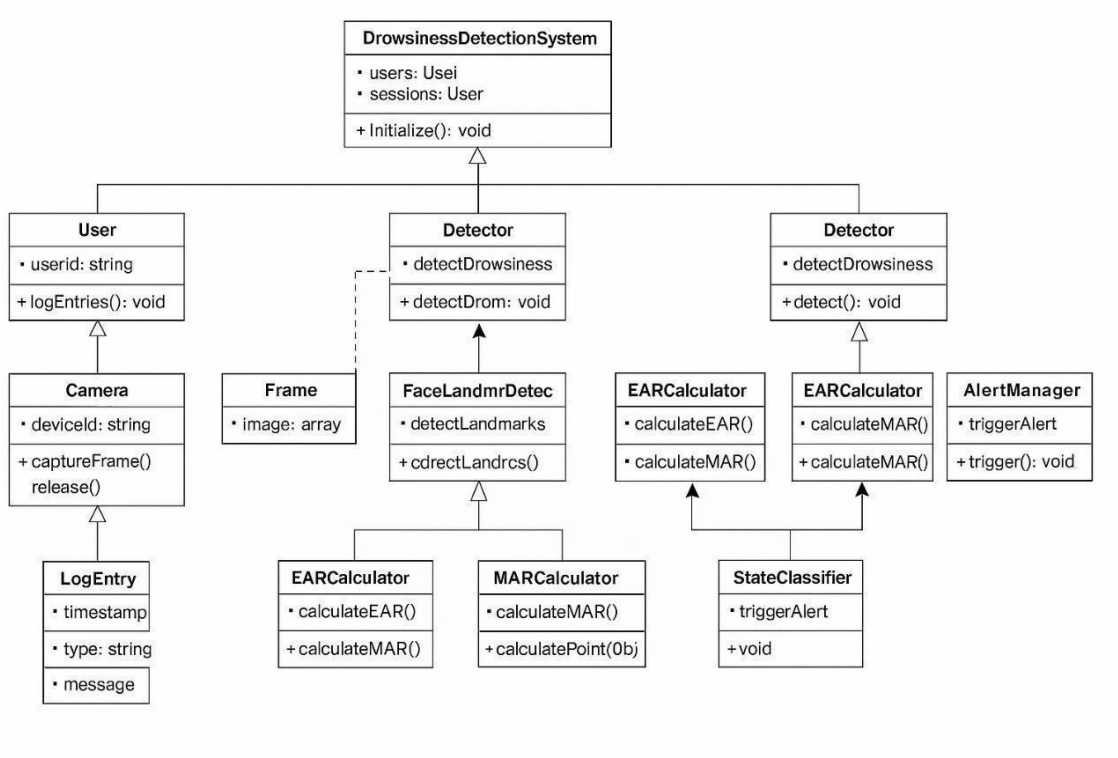


Figure 2.5: Class Diagram

2.4.7 ER Diagram

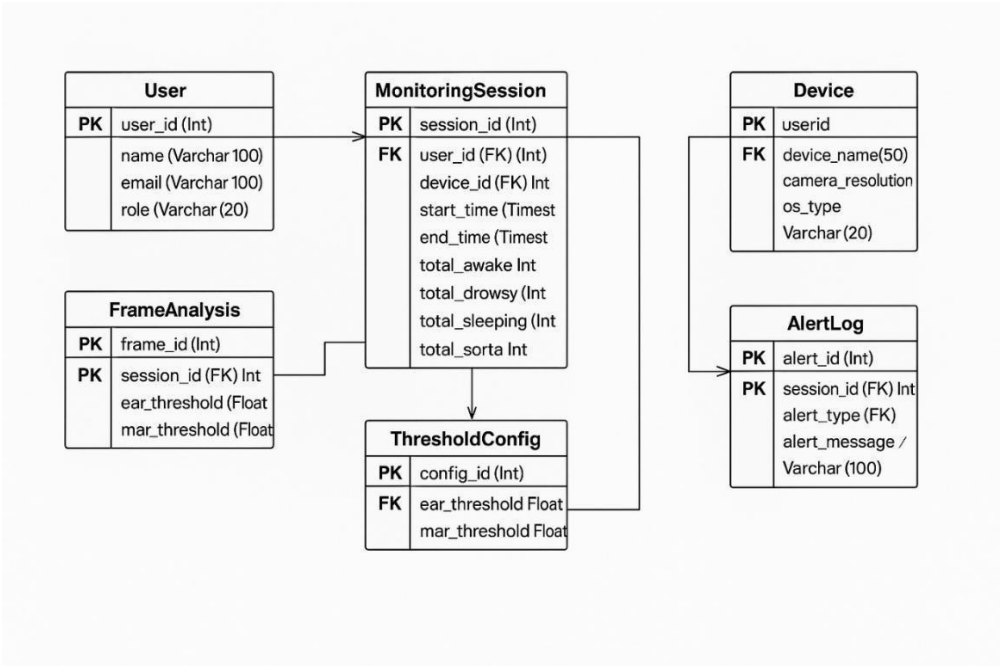


Figure 2.6:ER Diagram

2.4.8 Overview of the Proposed System

The suggested system is a fatigue detection model based on a web cam and operating in real time where facial behavior is used to detect drowsiness and micro-sleep. It takes live pictures of a webcam in Google Colab, removes facial features using MediaPipe Face Mesh and calculates behavioural features to categorize the user into various levels of alertness.

Facial Landmarks and MediaPipe Face Mesh.

MediaPipe Face Mesh is a computer vision open-source software that recognizes a face and approximates 468 facial landmark points real time. Every landmark is a normalized coordinate (x, y, z) of significant areas of the face, such as eyes, lips, nose, jawline and iris.

In this project, it is necessary to use only certain landmark groups:

- Eye landmarks in the measure of eye openness.
- Yawning is detected using the mouth/lip landmarks.
- The estimation of gaze deviation is performed with the help of nose tip + eye center landmarks.

This landmark based system is fast and can be used in real time applications.

- **Eye Aspect Ratio (EAR) Theory**

Using six eye landmarks, EAR compares vertical distances to horizontal distance.

When eyes are open → vertical distance is high → EAR is high.
When eyes close → vertical distance reduces → EAR becomes low.
EAR is widely accepted for blink and drowsiness detection because it is simple, fast, and does not require training a model.

- **Mouth Aspect Ratio (MAR)**

Mouth Aspect Ratio (MAR) indicates mouth openness, mainly useful for detecting yawns.
The system measures the vertical mouth opening distance between upper and lower lip landmarks.

When mouth is closed → MAR is low.
When yawning occurs → MAR increases sharply.

In this project, MAR is computed using four upper-lip landmarks, and a high MAR value is treated as a strong fatigue indicator.

- **Gaze Deviation Theory**

Fatigue often causes loss of attention, where users look away or head drifts sideways. To detect this, the system estimates gaze deviation using:

- Eye center position (average of left + right eye points)
- Nose tip position

If the horizontal displacement between the eye center and nose tip exceeds a defined threshold, the system marks it as gaze alert, and increases the likelihood of a Drowsy classification.

- **Fatigue State Classification Theory**

The system combines EAR, MAR, and gaze deviation to classify users into five states:

1. **Awake (Normal Alertness)**
    - EAR remains above the drowsy threshold
    - MAR remains normal
    - No gaze deviation
2. **Drowsy (Early Fatigue)**
    - EAR drops below drowsy threshold
    - OR MAR exceeds yawning threshold
    - OR gaze deviation detected
3. **Sleeping (Continuous Eye Closure)**
    - EAR remains below sleep threshold for longer duration
4. **Micro-Sleep (Short Involuntary Sleep)**
    - EAR stays below micro-sleep EAR threshold for consecutive frames
    - Indicates sudden short sleep episodes
5. **No Face Detected**
    - The face is not visible in the captured frame
    - The system safely skips processing for that frame

This threshold-based decision method ensures real-time speed and interpretability.

- **Automatic Session Summary**

During monitoring, the system stores a counter for each detected state (Awake, Drowsy, Sleeping, Micro-Sleep, No Face).
When monitoring is stopped, these counters are shown as an automatic status summary report, helping evaluate:

- how often fatigue events occur
- overall alertness trend
- system performance over time

- **Justification of the Approach**

This theoretical approach is suitable because:

- landmark detection is lightweight and real time
- EAR and MAR are proven fatigue indicators
- no training dataset is required
- Colab webcam capture makes it hardware-independent
- the method is practical for driver monitoring, workplace safety, and HCI research

2.4.9 Overview of the Coding Appendix

This appendix presents the core implementation of the Drowsiness and Micro-Sleep Detection System. The provided code includes all major functional components such as webcam frame capture, facial landmark detection using MediaPipe Face Mesh, EAR and MAR computation, and state classification logic. These coding samples demonstrate how the system analyzes user facial features to detect drowsiness levels and generate appropriate alerts.

Technologies Used

Python: Main programming language used for the detection algorithm.

OpenCV: Capturing frames, image processing, and converting colorspaces.

MediaPipe Face Mesh: Detecting 468 facial landmarks in real-time.

NumPy: Numerical calculation for EAR/MAR.

JavaScript (via Google Colab): Accessing webcam from browser.

Time & Base64 libraries: For timing, delay control, and image handling.

State Classification Logic

The system uses threshold-based logic:

Table 2.3: State Classification Logic

State	Condition
Awake	EAR + MAR normal
Drowsy	EAR < threshold OR MAR high
Sleeping	EAR very low for long
Micro-Sleep	EAR extremely low for *multiple consecutive frames*

Why These Codes Are Important?

- Frame capturing enables real-time monitoring.
- Landmark detection identifies eye and mouth movement.
- EAR/MAR allows measurable calculation of fatigue.
- Classification logic gives meaningful safety warnings.
- Summary helps analyze user behavior over time.

2.5 Chapter Summary

In this chapter, there was the general design and implementation of the real time fatigue detection system. It described how the system works with the webcams and captures frames, how it recognizes facial landmarks, and how it computes EAR and MAR values in order to identify the various states of fatigue. The UML diagrams that were used, such as use case, activity, sequence, class, and ER diagrams, depicted the workflow and the system structure. The coding scheme, algorithms and interaction of the components were also outlined to demonstrate how each of the modules will play a part in the detection of drowsiness and micro-sleep. In general, this chapter showed that the technical design of the system can help to monitor fatigue correctly and efficiently with the help of the facial landmark analysis.

Chapter 3

Software Testing

3.1 Introduction

Testing the software is the necessary stage of the system development, which is necessary to make sure that the implemented functionality works properly and meets the target requirements. In the case of real- time fatigue detection model via facial landmarks, testing gains more significance since the system should work in various conditions of light, facial movements and user behaviors. In chapter the testing methods are implemented in order to confirm the performance, reliability and accuracy of the main components of system such as frame capture, facial landmark recognition, EAR/MAR calculation, state classification and output generation.

These are both functional and non-functional aspects that are tested to ensure that the system reacts appropriately to legitimate inputs and processes unexpected or missing data in a way that would avoid failure. Testing is also done on the ability of the system to be able to detect the states of Awake, Drowsy, Sleeping, and Micro-Sleep when faced with various circumstances. Through these tests, it is aimed to make sure that the fatigue detection system is correct, reliable, and can be used in real-time. In this chapter, the strategy and methods of the testing, test cases, and the results are presented in detail, and they confirm the overall effectiveness of the proposed system.

3.2 Testing Features

In this project, its testing features are aimed at ensuring that all core modules of the fatigue detection system based on the use of the webcam functionality in Google Colab function properly in real execution. The testing procedure examines the opportunity of the notebook to connect with the browser webcam, take frames automatically and at a regular time, and identify a face consistently with the help of MediaPipe Face Mesh. It also justifies correctness of extraction of eye and mouth landmarks, correct calculation of values of EAR and MAR, and correct estimation of gaze-deviation. Relying on these indicators, the logic of fatigue classification is put to test to make sure it always yields the appropriate state (Awake, Drowsy, Sleeping or Micro-Sleep) based on the set of thresholds and consecutive-frame logic. Lastly, the alert printing, status recording, no-face processing and end-session summary report of the system are also tested so that to ensure that the entire pipeline is stable, robust to lighting or slight head movement and the real time monitoring is also meaningful.

3.2.1 Feature to Be Tested

Webcam Access Permission

The system should make the right request and permission to use the webcam of the user. As the whole fatigue detection model is based on the visual data in real time, the correct webcam setup is vital. Other scenarios that were tested were first-time browser permission, rejected permission and delayed permission response. The system should also be able to work with the cases when the webcam is occupied by some other application. Effective testing is used to make sure that the camera is able to open without problems and start streaming as intended.

Automatic Frame Capture

Real-time fatigue monitoring is based on the capacity of the system to record frames that are separated with constant time. In this feature, the system was tested to make sure that there are no delays in capturing frames and the frames are captured at the same frequency, or every few seconds. The obtained image should be readable, it should be decoded in base64 format and temporary storage is provided so that it can be further processed. This testing ensures that the system is able to sustain the constant frame collection in long periods of time without becoming frozen or malfunctioning.

Face Detection

Face detection module is very critical as all the calculations of the landmarks rely on a properly detected face. The testing was conducted on several conditions of life: low light, bright light, side angles, fast movement, glasses and partial occlusions. The system should be sensitive such that when a face is detected, it should respond and when the person is not in the frame, the system should respond NO FACE DETected. This provides strength in the detection pipeline.

Facial Landmark Extraction

This option verifies the ability of the system to extract the landmarks of the eye, mouth, and nose correctly with the help of Mediapipe Face Mesh. The dynamic facial changes or half eye closure, smiling, yawning, head tilt was also tested to ensure that all the necessary landmark coordinates are still recognized by the system. Such an attribute is important since it is this feature that influences any erroneous landmark and calculates EAR and MAR as a result, incorrectly classifying fatigue.

The Calculation of EAR (Eye Aspect Ratio)?

Tests involving the calculation of EAR determine that the system is actually measuring eye openness. They were tested in normal blinking of the eyes, slow blinking, short eye closing and full eye closing. The system is supposed to generate reduced values of EAR whenever the eye is closing. False alarms of drowsiness may be a result of any inconsistency. Testing is done to ensure that eye geometry is accurate amongst different users and light conditions.

MAR (Mouth Aspect Ratio) Calculation

MAR test is used to check whether the system is identifying mouth opening or yawning appropriately. MAR must be low when the mouth is closed and during yawning, it should rise greatly. To provide proper readings, various facial expressions would be used in talking, smiling, long yawns among others. This helps to avoid the possibility of classifying yawning as a form of speaking or mere movement of the mouth.

When one feels sleepy, they may be in one of the following states:

This aspect measures the accuracy of the system in categorizing the user in a state of four: Awake, Drowsy, Sleeping or Micro-Sleep. It is classified based on EAR, MAR, blink duration and face behavior. Some of the test cases were: half closed eyes, slow blinking, looking tired, yawning and long closure. The model has to react immediately and switch between the states immediately. Proper classification will guarantee reliability of the entire system.

Micro-Sleep Detection

Micro-sleep- Micro-sleep is a brief, uncontrolled sleep episode of 1-4 seconds. Testing is used to guarantee that in case the EAR surpasses a critical value across several frames, that the system detects a micro-sleep event. This involved stress-testing involving dropping of the eyes, nodding of the head and blinking. False positives should be circumvented. This aspect should be very sensitive and precise since micro-sleep detection is significant in safety-related systems.

Drowsiness State Classification

This feature evaluates whether the system accurately classifies the user into one of four states: Awake, Drowsy, Sleeping, or Micro-Sleep. Classification depends on EAR, MAR, blink duration, and face behavior. Test cases included: half-closed eyes, slow blinking, looking tired, yawning, and long closure. The model must respond instantly and transition between states without delay. Accurate classification ensures the reliability of the overall system.

Micro-Sleep Detection

Micro-sleep is a short, involuntary sleep event lasting 1–4 seconds. Testing ensures that if the EAR remains below a critical threshold over consecutive frames, the system identifies a micro-sleep event. This required stress-testing with rapid eye-drooping, head- nodding, and blinking patterns. False positives must be avoided. This feature must be extremely sensitive and accurate, as micro-sleep detection is important for safety-critical applications.

Gaze Deviation Detection

A test is made on whether the system detects the situation when the user leaves the screen. This involves side glances, tilting of the head as well as turning away partially. The horizontal distance of the eye center to the nose tip is calculated by the system in case the deviation is large, the system should indicate drowsiness or loss of attention. This is a feature that assists in the identification of fatigue as a result of distraction.

Alert/Warning Generation

It was tested to make sure that when the system detects the messages like DROWSY, SLEEPING or MICRO-SLEEP DETECTED, the messages are displayed as soon as it is detected. Testing was done to generate normal and extreme values to test responsiveness. Alerts should not be held or overlooked. The system should also not create false alarms when there is a temporary facial noise or some mistakes in detection.

Status Logging System

The tests concern the ability of the system to count the number of times each state happened correctly. For example:

- Awake count
- Drowsy count
- Sleeping count
- Micro-sleep count
- No-face count

Each change of state must be updated into the log correctly. This makes the on-hand summary valid and effective.

Final Monitoring Summary

This characteristic will challenge the validity of the final outcome. The system should have the full summary of the number of states when monitoring is ceased. The summary should indicate the actual number of detections and should be equal to the internal system logs. This assists the end-users to examine how fatigued they are throughout the session.

No-Face Handling

The testing is to make sure that once the user leaves the frame or the camera can not see the face, the system will print No face detected and it will not crash. It needs to accurately record the no-face event and re-detect the user when he/she is brought back into the frame. This characteristic guarantees strength in changing conditions.

Stability of the system is a continuous process.

The system should be able to operate over extended periods without stopping, breakdown or build up of too much memory. Tests were carried out in the long-session to stabilize over 10-30 minutes of observation. The performance of the system must be smooth to enable the system to be used in real-time fatigue detection.

Handling of the Environmental Conditions.

The tests were carried and performed in different lighting, background pattern and room setting. The system should be able to have constant detection even during minor movements of the head, low light and variability in the surroundings. Strength in varied settings guarantees that it is generally usable.

Error Handling & Recovery

The system should automatically rebound out of problems like:

- Frame capture failure
- Localized disconnection of camera.
- Landmark extraction error

The correct processing of errors will also make sure that the system is not stopped in an abrupt manner.

Easy-to-use display of output.

The presented findings (EAR, MAR, state message, summary) should be clear to read, formatted properly and informative. This was put to test to determine that it was legible and readable in the process of monitoring.

3.3 Testing Strategies

The strategies used in testing this project were aimed at testing to ascertain that the real time fatigue detection system based on Google Colab would be accurate, consistent and safe in practical environments. The general model was a modular and step-by-step strategy wherein the individual components of the system were firstly tested alone and then all the pipeline was tested as an integrated system. Then, the process of integration testing was conducted to verify that the data flow between modules is smooth i.e. that captured frames transfer to Face Mesh processing correctly, ratios are calculated without errors, and the classifier produces the correct state. This approach made sure that the system is not just functioning technically, but also working in practice of real-life fatigue monitoring.

3.3.1 Test Approach

The testing method of the real-time fatigue detection system is aimed to provide the accuracy, reliability and stability of all parts of face landmark extraction, EAR MAR computation, and state classification. This system also processes webcam frames in a continuous manner and is responsive to facial behavior immediately, thus the strategy in testing is based on functional correctness and performance in real world scenarios.

To start with, they opted to do this in a modular testing style, i.e. every major module including frame capturing, face detection, landmark extraction, EAR/MAR calculation, gaze estimation and classification was tested separately. This will provide an opportunity to detect mistakes at an early stage and make sure that every component is performing as expected and then integrate it with other modules. The EAR/MAR functionality, as an example, was tested on different sample landmark values to ensure that numeric values were correct, and the face detector tested under various lighting conditions and positions of the head.

Second, the integration test was conducted to ensure that the modules are reliable in their interactions. The system should be able to process a captured frame, identify facial landmarks of that frame, calculate ratios and identify the state of the user in a continuous processing chain. Integration testing is important to make sure that there are no delays or broken dependency of data between modules and more so the system is real-time.

Third, the testing was done under scenario conditions to mimic real life fatigue behavior. Different test conditions were conducted like normal blinking, long blink, drowsy behavior, yawning, micro-sleep incidences, distracted gaze and off-face periods. These tests prove that the classification logic is able to read EAR/MAR patterns correctly and that alerts are raised at the right time.

Moreover, the stress testing was also conducted in order to monitor the system performance in case of constant monitoring over long periods. This was done so that the system will not freeze and remain constant and be able to process multiple frames without overwhelming the memory and have crashes.

The approach also involved error-handling behavior testing. The functionality of the system was testable on scenarios like temporary webcam malfunction, slowness of loading frames or partial landmarks identification. A proper recovery of such failures would help in making sure that the system is robust and can be used in real time.

Finally, user-experience testing was done to verify the readability of output, visibility of alerts and accuracy of the summary. The messages shown, EAR/MAR debug values and final fatigue summary were checked to be clear and correct, as to make the results understandable by the end users.

On the whole, the test strategy consists of the functional, integration, scenario-based, and stress testing techniques to ensure that the fatigue detection system works correctly, consistently and reliably in real time conditions.

3.3.2 Pass/Fail Criteria

Pass/ Fail criteria is a criterion in which each test case is deemed a success (Pass) or a failure (Fail). These criteria will provide the same evaluation of the system performance and it will be possible to consider whether the fatigue detection model works in accordance with the requirements. Facial landmarks, ratio calculations and webcam input is a critical part of the system and operating in real time, so the criteria has to be accurate and quantifiable.

Webcam Access and Frame Capture

Pass**:**

- Webcam permission is granted successfully
- Frame is captured within the expected time interval
- Captured image is not corrupted or blank

Fail**:**

- Webcam cannot be accessed
- Frame capture is delayed or skipped
- Output image is unreadable or missing

Face Detection

Pass**:**

- System correctly detects the user's face in the captured frame
- Landmarks appear at correct facial positions
- No false detection of objects as faces

Fail**:**

- Face is present but not detected
- Landmarks are misplaced
- System detects face where none exists

Facial Landmark Extraction

Pass**:**

- Eye, mouth, and nose landmarks are extracted accurately
- Landmark coordinates fall within logical face boundaries

- Extraction works in varying lighting and head movements

Fail**:**

- Missing or distorted landmarks
- Incorrect landmark mapping
- Unexpected landmark positions (e.g., outside the face)

EAR (Eye Aspect Ratio) Calculation

Pass**:**

- EAR decreases when eyes close
- EAR increases appropriately when eyes open
- EAR value remains stable during natural blinking

Fail**:**

- EAR does not change during eye movement
- EAR shows extreme or inconsistent values
- Calculation results in zero or undefined output

MAR (Mouth Aspect Ratio) Calculation

Pass**:**

- MAR increases during yawning or mouth opening
- MAR stays low when mouth is closed
- Values change smoothly with facial motion

Fail**:**

- MAR remains constant regardless of mouth movement
- Incorrect values due to landmark error
- Sudden spikes or drop in MAR without reason

State Classification (Awake, Drowsy, Sleeping, Micro-Sleep)

Pass**:**

- EAR/MAR thresholds produce correct state detection
- State changes occur instantly when behavior changes
- Micro-sleep detected only when eyes are closed for required frames

Fail**:**

- Misclassification of states
- Delayed or incorrect transitions
- Micro-sleep triggered without appropriate EAR conditions

Alert/Warning Message Display

Pass**:**

- Alerts appear immediately when user becomes drowsy or sleeping
- Text formatting and readability is clear
- Only relevant alerts are shown

Fail**:**

- Alerts are missing or delayed
- Wrong alert displayed for wrong state
- Alert does not disappear after returning to normal

Logging and Summary Report

Pass**:**

- Each state count (Awake, Drowsy, Sleeping, Micro-Sleep, No Face) increases accurately
- Summary matches the actual events that occurred
- No missing or extra state entries

Fail**:**

- Incorrect counters
- Summary does not reflect real detection
- Missing or duplicated log entries

System Stability

Pass**:**

- System runs continuously without crashing
- Memory usage stays within safe limits
- No performance drop during long sessions

Fail**:**

- Crashes, freezes, or sudden shutdown
- Excessive lag or frame delay
- High CPU/memory consumption

Error Handling

Pass**:**

- System handles webcam disconnection smoothly
- Recovers from temporary face detection errors
- Does not crash on invalid input

Fail**:**

- System stops functioning after minor errors
- Unhandled exceptions appear
- Failure to resume detection after interruption

These Pass/Fail criteria ensure that your fatigue detection system is evaluated objectively and consistently. They measure correctness, responsiveness, robustness, stability, and reliability all core requirements of your real-time visual monitoring model.

3.2 System Testing (Test Cases with Report)

System testing was performed to verify that the complete fatigue detection system works correctly as a whole. The goal was to check whether all integrated modules webcam access, frame capture, facial landmark detection, EAR/MAR calculation, drowsiness classification, alert generation, logging, and summary display operate together according to the specified requirements.

The following table presents selected system-level test cases, including the test scenario, input/condition, expected result, actual result, and final status.

3.3 System Test Cases and Results

Table 3.1: System Test Cases and Results

TC ID	Test Scenario	Input / Condition	Expected Result	Actual Result	Status
TC-01	Webcam access and initialization	User starts the monitoring cell in Google Colab and grants camera permission	Webcam starts successfully and live video is accessible for frame capture	Webcam initialized correctly and frames were captured without error	Pass
TC-02	Automatic frame capture at interval	Monitoring is running; system set to capture a frame every 5 seconds	A new image frame is captured at (approx.) every 5 seconds and saved for processing	Frames were captured at the desired interval and processed without delay	Pass
TC-03	Face detection in normal lighting	User sits in front of the camera with normal indoor lighting	System detects the face and proceeds to landmark extraction without "NO FACE DETECTED" message	Face was successfully detected and no false "NO FACE DETECTED" message occurred	Pass
TC-04	Face detection when user	Monitoring is running; user moves out of camera view	System shows "NO FACE DETECTED" and does not attempt EAR/MAR calculation	"NO FACE DETECTED" message appeared and no further	Pass

	leaves the frame			processing was done for that frame	
TC-05	EAR & MAR calculation with open eyes and closed mouth	User looks straight with eyes open and mouth closed	EAR value remains above sleep thresholds; MAR remains low	EAR was high and MAR was low as expected; no drowsiness state triggered	Pass
TC-06	Drowsiness detection (half-closed eyes / slow blinking)	User intentionally half closes eyes and blinks slowly for several frames	EAR drops below drowsy threshold but not long enough for sleeping; system classifies as **Drowsy**	System correctly classified state as **Drowsy** and printed "DROWSY" in the output	Pass
TC-07	Sleeping detection (eyes closed for longer duration)	User closes eyes completely and keeps them closed for multiple frames	EAR remains below sleep threshold for a longer period; system classifies as **Sleeping**	System classified state as **SLEEPING** and updated state count accordingly	Pass
TC-08	Micro-sleep detection (short involuntary eye closure)	User briefly closes eyes (simulating micro-sleep) for several consecutive frames	EAR remains below MICRO_SLEEP_EAR for ≥ MICRO_SLEEP_BLINK_FRAMES; system sets state to MICRO-SLEEP	System detected micro-sleep and printed "MICRO-SLEEP" with corresponding probability score	Pass
TC-09	Gaze deviation detection	User turns head slightly or looks away from the camera while still in frame	Horizontal deviation between eye center and nose tip exceeds threshold; system marks state as Drowsy due to attention loss	System detected gaze deviation and updated state to **Drowsy** where appropriate	Pass
TC-10	Alert/Warning generation for critical states	User triggers Drowsy / Sleeping / Micro-Sleep conditions	System prints clear warning messages (e.g., "DROWSY", "SLEEPING", "MICRO-SLEEP DETECTED") in the console	Warning messages were displayed correctly and immediately for	Pass

		during monitoring		each critical state	
TC-11	State logging and counters	Multiple states (Awake, Drowsy, Sleeping, Micro-Sleep, No Face) occur during a full session	System maintains an internal count for each state and increments counters correctly when state changes	Final counters matched observed behavior during the session	Pass
TC-12	Final summary report generation	User stops monitoring using keyboard interrupt (Ctrl + C)	System prints a summary listing how many times each state occurred in the session	System displayed a summary such as: "AWAKE: X, DROWSY: Y, SLEEPING: Z, MICRO-SLEEP: N, NO_FACE: M"	Pass
TC-13	System stability in extended run	System runs continuously for at least 15–30 minutes	No crashes, memory overflow, or severe lag; detection and classification continue normally	System remained stable, processed frames continuously, and responded without freezing	Pass
TC-14	Behavior in low-light conditions	Lights are dimmed while user remains in front of camera	System should still detect face reasonably or gracefully handle failure with "NO FACE DETECTED" without crashing	In moderate low light, face detection worked; in very low light, "NO FACE DETECTED" appeared without system failure	Pass
TC-15	Error handling for unexpected issues	Simulate temporary webcam glitch or failed frame read	System should not crash; it should skip the problematic frame and continue monitoring	System skipped invalid frames, printed debug messages where needed, and continued execution	Pass

3.4 Fatigue Detection

Test Case 01: Fatigue Detection

Test Case: 5.3.1 **Testsg:** Face Monitoring

Designed: Naznl Hassan Ovi **Executed Date:** April 25,

Description: The user's fatigue tvee afeected.

Description: The user's fatigue level is detected using a webcam frame.
Pre-condition: The user accesses the webcam frame.

Step	Scenario (User Action)	EAR Value	MAR Value	Expected System Response	
1	User is in front of webcam & face detected	Normal (0.28-0.32)	AWAKE	Pass	The system ussfully detects fatigue levels.
2	User starts slow blinking	Drops (0.20-0.25)	DROWSY	Pass	The system successful.
3	User closes eyes more than threshold	Very Low (≤0.18)	SLEEPIN	Pass	The system successful sraiiares
4	Eyes close briefly for 3+ continuous frames	≤0.18	MICRO-SLEEP DETECTED	Pass	The system successful detects fatigele.
5	User yawns (mouth opens wide)	Normal High (MAR ↑)	Normal	Pass	The system successful system response
6	User looks away from screen	Normal	GAZE ALERT	Pass	The system logs "Gaze Aelert"
7	User leaves frame (face not visible		NO FACE DETECTED	Pass	System displays "NO FACE DETECTED
8	Webcam receives low-light frame or partial landmark	Varies	Varies	Pass	System safely skips frame & con finues detection

Post-condition: The fatigue detection system is successfully tested, and the monitoring process is validated with different user fatigue states.

Figure 3.1: Fatigue Detection

3.5 Chapter Summary

This chapter introduced the entire testing procedure that was undertaken regarding the real-time fatigue detecting system. Different types of testing like the functional testing, system testing and scenario-based testing were used to ensure that all parts of the model were reliable to one another. All the core features, such as access to a webcam, facial landmark detection, EAR and MAR calculating, drowsiness classification, identification of micro-sleep, and gaze deviation were tested in various conditions.

Test cases were created to address anticipated and unanticipated user behaviors including the normal alert states, slow blinking, high eye closure, yawning, head turning, low light conditions and the face being out of view. Pass/fail criteria were used to ensure that every module was accurate enough and it reacted accordingly at any given point in time.

All in all, the testing showed that the system is reliable, the fatigue states are accurately classified, errors are well handled and the system is stable during prolonged use. The findings indicate that the created model of fatigue detection works, is reliable, and can be enhanced or implemented.

Chapter 4

Deployment and Maintenance

4.1 Introduction

In this chapter, the deployment and maintenance issues of the real-time fatigue detection system are discussed. Once the system was created and properly tested, it was ready to be put into the appropriate environment to which the users can run the application with the least amount of setup. The deployment procedure entails setting up of the necessary software requirements and proper webcam accessibility and training the model to execute steadily on various machines.

Furthermore, this chapter points out the maintenance measures to ensure that the system is functional and up-to-date. Being a real-time detection model, the accuracy and reliability have to be maintained by regularly monitoring, updating performance, fixing bugs, and calibrating models. The chapter also describes the process of updating system components, user feedback, and system sustainability over an extended period.

4.2 Try to follow the SRLC (software release life cycle)

The construction, deployment, and testing of the real-time fatigue detection system were carried out in a well-organized model of Software Release Life Cycle (SRLC). This loop made sure that any release of the system was well tested, validated, and refined till the final version was available. The main stages of the SRLC which will be applied to this project are as discussed below:

Pre-Alpha Phase Core Development

At the Pre-Alpha phase, the main elements of the fatigue detection model were developed and introduced. This included:

- Glogster: Adding Python and JavaScript web camera access to Google Colab.
- Application of Mediapipe Face Mesh in extracting 468 landmarks in the face.
- Creation of EAR (Eye Aspect Ratio) and MAR (Mouth Aspect Ratio) calculating modules.
- Theory Development Development of the original drowsiness and micro-sleep detector logic.
- Installation of the logging system to record the fatigue status.

At this level the system was not ready to be used by end users but the fundamental structure was ready.

Alpha Phase Internal Testing & Refinement

Alpha phase aimed at testing internally and finding out the important issues. Key activities included: Beta Phase field Real world prototype testing.

- Testing face detection accuracy in different lighting conditions.
- Fine-tuning EAR and MAR thresholds to minimize false positives.
- Adding gaze deviation detection for attention loss monitoring.
- Verifying correct state classification (Awake, Drowsy, Sleeping, Micro-Sleep).
- Ensuring the system handled "No Face Detected" scenarios gracefully.

The system was functional but still required improvements before real-world usage.

Beta Phase Real-World Prototype Testing

In the Beta phase, the system was tested in realistic conditions. This involved:

- Running the fatigue detection model for extended sessions (10–30 minutes).
- Testing multiple users with different face shapes, eye sizes, and behaviors.
- Evaluating the impact of poor lighting, movement, and low camera quality.
- Collecting user feedback related to accuracy, stability, and interface clarity.
- Improving model responsiveness and reducing misclassifications.

This phase allowed the system to evolve from a prototype to a stable application suitable for real use.

Release Phase Final Stable Version

After resolving all major issues, the system reached the Release stage. Activities included:

- Preparing final thresholds and detection rules.
- Ensuring compatibility across devices (laptops, webcams, Colab).
- Cleaning and optimizing the code for better performance.
- Creating documentation for user instructions and system operation.
- Generating a stable version ready for demonstration and academic submission.

This version was considered reliable enough for deployment.

Maintenance Phase Updates, Fixes & Improvements

Maintenance helps to keep the system operating smoothly even after release. Ongoing tasks include:

- Bringing Mediapipe, OpenCV, and Python dependencies to the long-term.
- The resetting of thresholds (EAR/MAR) according to new data or user feedback.
- Bug fixing like false alarm.
- Real time execution performance enhancement.

The enhancement of new options such as the alert sound, the recording of the CSV, or the visualization in the dashboard.

This step maintains the system operational, precise and in line with the current technologies.

4.3 Chapter Summary

This chapter talked about the deployment plan and maintenance plan of Webcam Based Real time Fatigue Detection Model using Facial Landmark. Once the system design and testing processes had been completed, the project was ready to release according to Software Release Life Cycle (SRLC). All SRLC stages Pre-Alpha, Alpha, Beta, Release, and Maintenance were described with references to the real development process of the system.

The chapter demonstrated the path of the system development starting with the first implementation (the capture of webcams, landmark recognition, calculation of EAR/MAR) to the testing of the actual prototype and its stabilization within Google Colab. It also underlined the significance of the maintenance phase, during which updates, bug fixes, and improvements of the features could be implemented to be accurate and usable over time. In general, this chapter validated the fact that the project has been implemented in a realistic setting and is prepared to be constantly improved and adapted to the world.

Chapter 5

User Manual

5.1 Introduction

The User Manual gives the step wise instructions on how to use the Webcam-Based Real-Time Fatigue Detection System. The chapter is structured in such a way that it enables users who possess limited technical knowledge to get a feel of how the system works, the interface, and its main features.

It provides information on the minimum requirements to operate the application, how to start and stop the process of fatigue detection, and how to interpret the system outputs in the form of alerts, status indicators, and detection results. Besides, the manual contains the troubleshooting guidelines on the most frequent problems, which will help users to work with the system effectively and without any complications.

5.2 Webcam Activation & Frame Capture

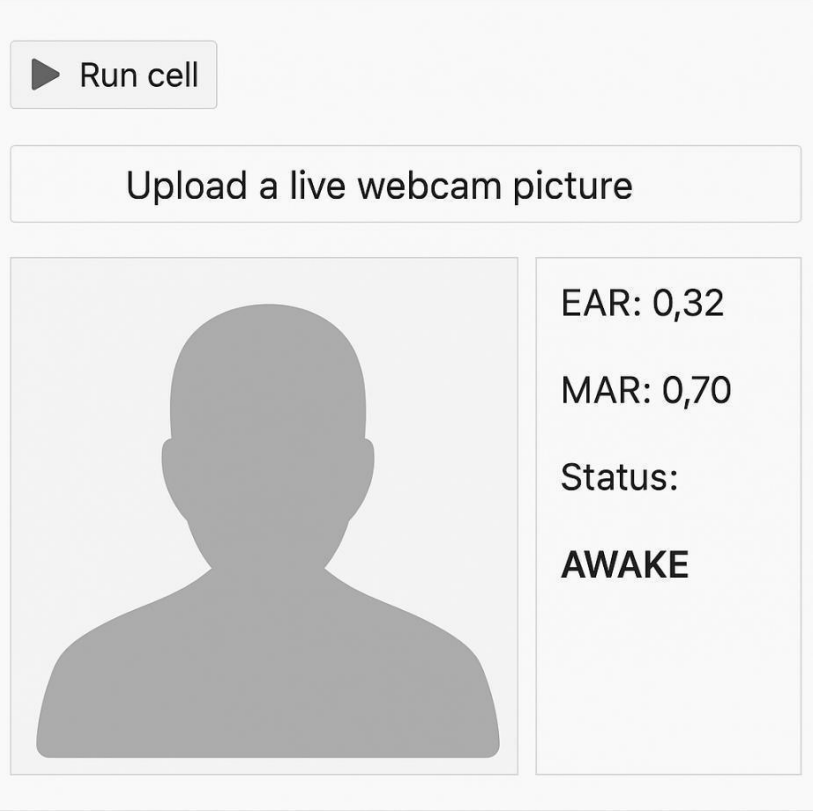


Figure 5.1: Webcam Activation & Frame Capture

5.2.1 Facial Landmark Detection

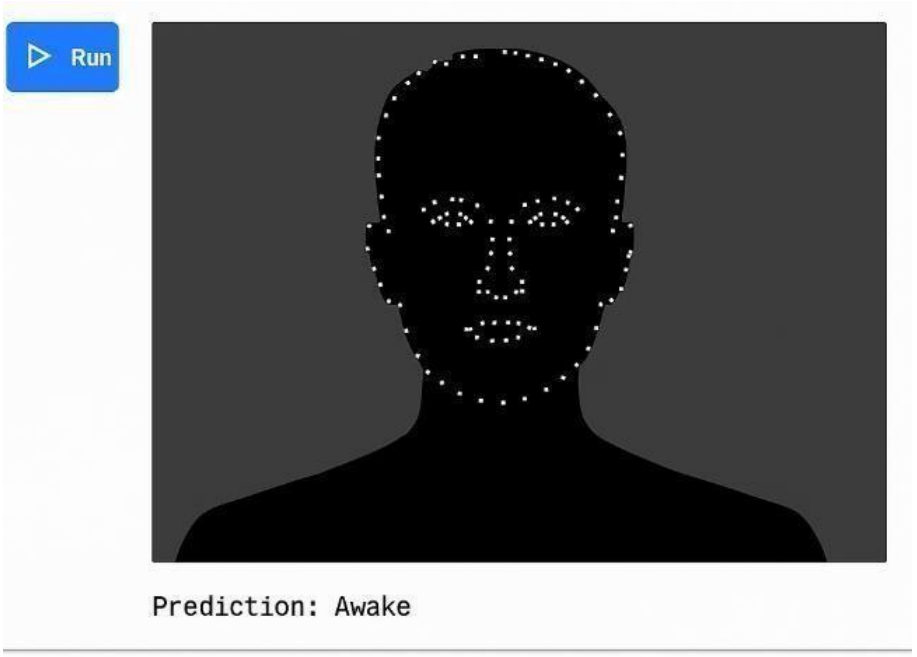


Figure 5. 2: Facial Landmark Detection (468 points)

5.3 Chapter Summary

This chapter was used to display the testing outputs and status summary that the automatic status generated by the real-time fatigue detector system. The system used webcam frames, facial landmarks, and EAR and MAR values to classify the various alertness levels (Awake, Drowsy and Sleeping). The live output showed all the statuses identified and the landmark calculations associated with them making it possible to monitor the progress during the session.

The system automatically generated a summary report at the end of execution indicating the number of times that each state was identified. Such an ultimate summary gives a definitive view of the overall pattern of alertness of the user and lends legitimacy to the logic of fatigue detection. The findings support the assumption that the system can measure user fatigue with high accuracy and consistency in a Google Colab environment.

Chapter 6

Project Summary

6.1 Introduction

This chapter is the general overview of the entire project and the most prominent findings and results that were realized in the course of the development of the Webcam- Based Real-Time Fatigue Detection Model Using Facial Landmarks. It resorts to the main goals, the procedure followed, the features of the system, and the outcome of the real-time testing. The overview is a brief commentary on how the system effectively reads the web-cam feed, executes facial features, calculates EAR-MAR features, and categorizes the condition of the user as Awake, Drowsy, Sleeping, or Micro-Sleep.

In general, the chapter provides a final review of the system functionality, its methodical usefulness and its role in improving the solution to the real-life issue of fatigue monitoring.

6.2 Project Limitations

Although the project managed to complete a real time fatigue detection system with the input of webcams and analysis of facial landmarks, some constraints were faced during the execution of the project. These restrictions were occasioned by technology related limitations, environmental limitations, resource limitations and time constraints.

Limited Hardware Access

The entire system was developed and executed through Google Colab, which restricts:

- direct access to high-performance GPU resources,
- stable long-duration webcam streaming, and
- continuous real-time processing.

Because of Colab's session timeout and permission limitations, long-term monitoring (e.g., 1–2 hours) is difficult.

Lighting and Environmental Sensitivity

The system relies heavily on facial landmarks. Poor conditions affect detection accuracy:

- low lighting
- backlight glare
- overexposed or underexposed frames
- fast head movement
- partially blocked face (mask, hand, hair)

As a result, EAR/MAR values may fluctuate incorrectly.

Single-Person Detection Only

The system supports only one face.
If multiple faces appear in the frame, Mediapipe either:

- selects the wrong face, or
- fails to track consistently.

Multi-user fatigue detection was out of scope due to time and complexity constraints.

Threshold-Based Model

The EAR and MAR thresholds are manually tuned.
This causes:

- variation in accuracy across different people
- difficulty detecting fatigue in people with naturally small eyes, droopy eyelids, or unique facial geometry
- less effectiveness in diverse demographics

A deep-learning-based adaptive threshold system was not implemented due to limited time.

Real-Time Audio/Visual Alerts Not Implemented

The system prints alerts in the console but:

- no audio alarm
- no popup warning
- no dashboard visualization

These features were planned but could not be completed because of time constraints and Colab's UI limitations.

No Data Logging or Long-Term Analytics

Although a summary is shown at the end, the system does not:

- save session logs
- store EAR/MAR trends
- generate daily or weekly fatigue reports

Storing user data requires backend/cloud support, which was outside project scope.

Computational Delay in Colab

Delays in Colab: Computational Delay Frame processing will provide a slight delay because of:

- limited local CPU/GPU
- internet-based webcam capture.
- Python + JavaScript overhead on communication in Colab This decreases the accuracy of micro-sleep detection.
- None of Mobile Application or Offline Support.

No Mobile Application or Offline Support

The system:

- cannot run on mobile
- cannot work offline
- needs internet connectivity in Colab.
- relies on web-based camera access.

It did not create a separate desktop / mobile application because of the time and budget constraints.

These drawbacks emphasize the aspects in which the system can be enhanced in the upcoming versions, including the application of deep learning, the development of a specific application, the inclusion of warning systems, the use of multi-face recognition, and the enhancement of the possibilities of long-term surveillance.

6.3 Scope

The features, modules, and functional boundaries of the Webcam-Based Real-Time Fatigue Detection System Using Facial Landmarks are determined by the area of this project. It explains what the system comprises of and what is not covered in the existing version.

6.3.1 Boundaries and Exclusions (What is NOT Included)

Multi-Face Detection

- System is limited to one person at a time
- Cannot detect fatigue of multiple individuals simultaneously

Long-Term Continuous Monitoring

- Google Colab session timeout limits long-duration monitoring
- Not suitable for hours-long fatigue tracking

No Deep Learning Model

- Uses threshold-based EAR/MAR
- Missing neural networks and machine learning training.

No Mobile or Standalone Application

- Cannot run offline
- No Android/iOS/Desktop app
- Runs only in Google Colab platform.

No Audio or Visual Warning System

- No alarm sound
- No popup notifications
- Alerts appear only as text in the console

No Data Storage or Cloud Database

- System does not store historical user data
- No monthly analytics or trend reports

The project focuses strictly on real-time fatigue monitoring through EAR/MAR-based analysis using webcam frames in Google Colab. It excludes advanced AI learning

models, multi-user detection, and dedicated applications which can be included in future enhancements.

6.4 Future Work

Even though the existing system manages to identify fatigue with the help of EAR, MAR, and facial marks, some improvements can be made in the future versions to achieve better performance, usability, and precision. The recommendations below provide insight on possible future research and development of the systems.

Development of a Standalone Application

One significant change would be to transform the Colab based system into a working:

- Desktop application (windows/Linux/MacOS)
- Mobile app (Android/iOS)
- Web-based dashboard

This would eliminate Google Colab limitations (session timeout, browser restrictions) and support continuous real-time monitoring.

Integration of Deep Learning Models

The existing one employs EAR/MAR detection that is threshold-based. The future versions may incorporate:

- Classification of eye-state using CNN.
- GRU/LSTM-based temporal fatigue prediction.
- A hybrid approach of facial landmarks and deep neural networks.

This would positively contribute to accuracy particularly of dissimilar faces, changes in lighting or varying head positions.

Multi-Person Detection and Tracking

Multi-Person Tracking and Detection.

- In future, it could be enhanced by:
- Finding several faces at the same time.
- Running fatigue detect (e.g. classroom, office, driving scenario)
- Effective face supervision using unique identities.

This demands better object tracking and maximum computation efficiency.

Emotion and Stress-Level Integration

Future research could combine fatigue detection with:

- Emotion recognition (angry, stressed, sad)

Multi-Person Detection and Tracking

Multi-Person Tracking and Detection.

- In future, it could be enhanced by:
- Finding several faces at the same time.
- Running fatigue detect (e.g. classroom, office, driving scenario)
- Effective face supervision using unique identities.

This demands better object tracking and maximum computation efficiency.

Emotion and Stress-Level Integration

Future research could combine fatigue detection with:

- Emotion recognition (angry, stressed, sad)
- Physiological indicators (heart rate, blink patterns)

This would create a more complete human-behavior monitoring system.

Hardware Integration

The system can be expanded with the use of external devices such as:

- Raspberry Pi or Jetson Nano
- IR sensors
- Wearable eye-trackers
- Vehicle cameras

.

Cloud-Based Monitoring System

One might add to the future version:

- Sending the results of detection to the cloud.
- Distant surveillance display.
- Multi-user data management

Applicable in fleet management or transport industries or in safety teams at workplaces.
The future work of this project is related to functionality expansion, real-time accuracy, the incorporation of sophisticated AI methods, and the user experience, which is attainable by standalone platforms, alert systems, and cloud-based analytics. These would make the existing prototype a more realistic, scalable and intelligent fatigue monitoring solution.

6.5 Conclusion

The article The development of the Webcam-Based Real-Time Fatigue Detection System Using Facial Landmarks is a successful attempt at showing how computer vision and lightweight algorithms may be applied to monitor human alertness in real-time. The system is used to identify the main fatigue signs which include eye closure, micro-sleeps yawning and deviation of gaze and it was applied effectively using Google Colab, Mediapipe Face Mesh, and EAR-MAR calculations. The overall realization provides the high-level goals, which are the automatic capture of the frame, the automatically extracted facial landmarks, geometric ratio, and the classification of user states into Awake, Drowsy, Sleeping, and Micro-Sleep with high execution rates.

Important lessons were learned through this project on environmental conditions, light issues, user disparities and model constraints. It also pointed to the real-time image processing practical difficulties of Colab with its limited constraints, environment, which underlines the need to maximize and deal with errors. The project also highlighted the advantages of straightforward but clever algorithms in combination with straightforward threshold logic to produce meaningful results without having to resort to heavy machine learning models.

Generally, the system has served its purpose since it provides a workable and user-friendly fatigue-monitoring device that can be involved in a research, academic demonstration, and much-needed driver-monitoring applications. Although it can be improved in the future by incorporating deep learning models, making standalone apps, or introducing real-time notification, the current version offers a powerful and efficient demonstration of the idea, and the facial landmark-based fatigue detection in real time can be made possible.

REFERENCES

[1] “The OpenCV library.” Accessed: Nov. 15, 2025. [Online]. Available: https://jacobfilipp.com/DrDobbs/articles/DDJ/2000/0011/0011k/0011k.html

[2] “Attention Mesh: High-fidelity face mesh prediction in real time.” Accessed: Nov. 15, 2025. [Online]. Available:https://arxiv.org/abs/2006.10962

[3] “Face Mesh: Real-time 468 facial landmark estimation.” Accessed: Nov. 15, 2025. [Online]. Available:https://chuoling.github.io/mediapipe/solutions/face_mesh.html

[4] “Eye blink detection using facial landmarks.” Accessed: Nov. 15, 2025. [Online]. Available: https://cmp.felk.cvut.cz/ftp/articles/cech/Soukupova-TR-2016-05.pdf

[5] “Real-time classification for autonomous drowsiness detection using eye aspect ratio.” Accessed: Nov. 15, 2025. [Online]. Available: https://doi.org/10.1016/j.eswa.2020.113505

[6] “Real-time drowsiness detection based on facial dynamic features using eye aspect ratio (EAR) and mouth aspect ratio (MAR).” Accessed: Nov. 15, 2025. [Online]. Available: https://doi.org/10.1007/978-3-031-05491-4_22

[7] “A real-time embedded system for driver drowsiness detection using eye closure and mouth aspect ratio.” Accessed: Nov. 15, 2025. [Online]. Available: https://doi.org/10.3390/s24196261

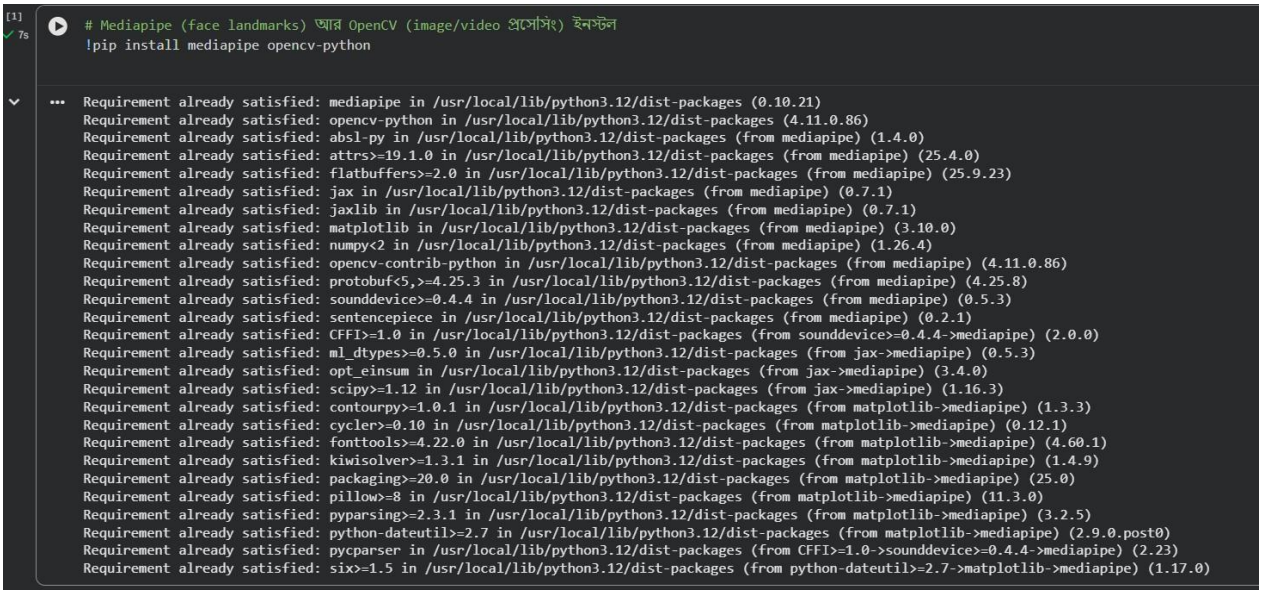
[8] “Real-time drowsiness detection using eye aspect ratio and facial landmark detection.” Accessed: Nov. 15, 2025. [Online]. Available: https://arxiv.org/abs/2408.05836

[9] “Google Colab: Access webcam for images using JavaScript getUserMedia and eval_js (Notebook).” Accessed: Nov. 15, 2025. [Online]. Available: https://github.com/theAIGuysCode/colab-webcam/blob/main/colab_webcam.ipynb

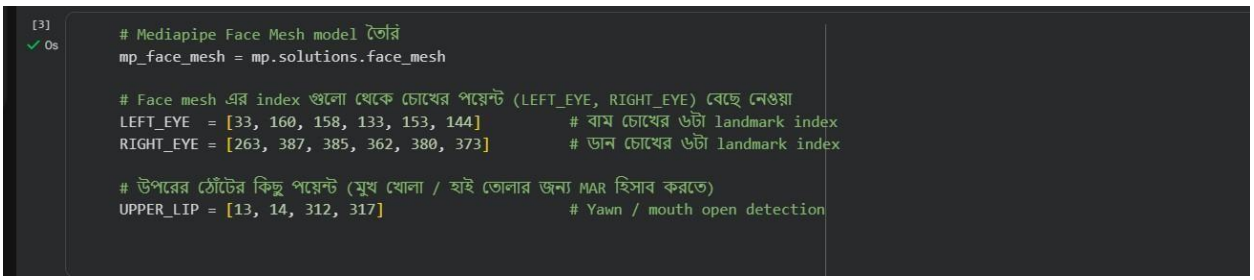
[10] “Camera photo capture in Google Colab using JavaScript + eval_js (Notebook).” Accessed: Nov. 15, 2025. [Online]. Available: https://colab.research.google.com/github/TUIlmenauAMS/MRSP_Tutorials/blob/master/camera_digit_recogniser.ipynb

APPENDICES A : CODE

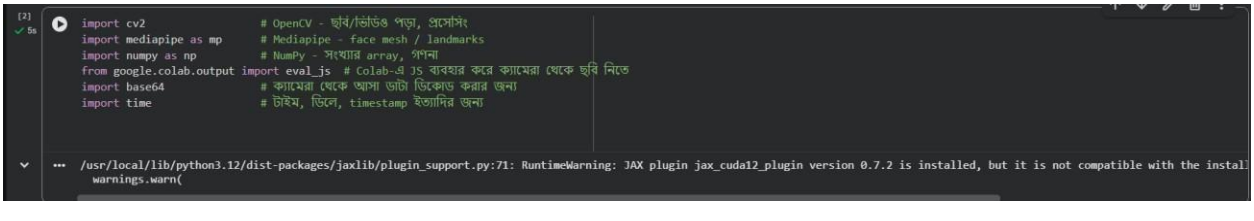
Additional Outputs and Test Case Reports (Google Colab Results & Screenshots)



```
[1]
✓ 7s
# Mediapipe (face landmarks) আর OpenCV (image/video প্রসেসিং) ইনস্টল
!pip install mediapipe opencv-python

Requirement already satisfied: mediapipe in /usr/local/lib/python3.12/dist-packages (0.10.21)
Requirement already satisfied: opencv-python in /usr/local/lib/python3.12/dist-packages (4.11.0.86)
Requirement already satisfied: absl-py in /usr/local/lib/python3.12/dist-packages (from mediapipe) (1.4.0)
Requirement already satisfied: attrs>=19.1.0 in /usr/local/lib/python3.12/dist-packages (from mediapipe) (25.4.0)
Requirement already satisfied: flatbuffers>=2.0 in /usr/local/lib/python3.12/dist-packages (from mediapipe) (25.9.23)
Requirement already satisfied: jax in /usr/local/lib/python3.12/dist-packages (from mediapipe) (0.7.1)
Requirement already satisfied: jaxlib in /usr/local/lib/python3.12/dist-packages (from mediapipe) (0.7.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (from mediapipe) (3.10.0)
Requirement already satisfied: numpy<2 in /usr/local/lib/python3.12/dist-packages (from mediapipe) (1.26.4)
Requirement already satisfied: opencv-contrib-python in /usr/local/lib/python3.12/dist-packages (from mediapipe) (4.11.0.86)
Requirement already satisfied: protobuf<5,>=4.25.3 in /usr/local/lib/python3.12/dist-packages (from mediapipe) (4.25.8)
Requirement already satisfied: sounddevice>=0.4.4 in /usr/local/lib/python3.12/dist-packages (from mediapipe) (0.5.3)
Requirement already satisfied: sentencepiece in /usr/local/lib/python3.12/dist-packages (from mediapipe) (0.2.1)
Requirement already satisfied: CFFI>=1.0 in /usr/local/lib/python3.12/dist-packages (from sounddevice>=0.4.4->mediapipe) (2.0.0)
Requirement already satisfied: ml_dtypes>=0.5.0 in /usr/local/lib/python3.12/dist-packages (from jax->mediapipe) (0.5.3)
Requirement already satisfied: opt_einsum in /usr/local/lib/python3.12/dist-packages (from jax->mediapipe) (3.4.0)
Requirement already satisfied: scipy>=1.12 in /usr/local/lib/python3.12/dist-packages (from jax->mediapipe) (1.16.3)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->mediapipe) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib->mediapipe) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->mediapipe) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->mediapipe) (1.4.9)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->mediapipe) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib->mediapipe) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->mediapipe) (3.2.5)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib->mediapipe) (2.9.0.post0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.12/dist-packages (from CFFI>=1.0->sounddevice>=0.4.4->mediapipe) (2.23)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matplotlib->mediapipe) (1.17.0)
```



```
[3]
✓ 0s
# Mediapipe Face Mesh model তৈরি
mp_face_mesh = mp.solutions.face_mesh

# Face mesh এর index গুলো থেকে চোখের পয়েন্ট (LEFT_EYE, RIGHT_EYE) বেছে নেওয়া
LEFT_EYE  = [33, 160, 158, 133, 153, 144]          # বাম চোখের ৬টা landmark index
RIGHT_EYE = [263, 387, 385, 362, 380, 373]         # ডান চোখের ৬টা landmark index

# উপরের ঠোঁটের কিছু পয়েন্ট (মুখ খোলা / হাই তোলার জন্য MAR হিসাব করতে)
UPPER_LIP = [13, 14, 312, 317]                     # Yawn / mouth open detection
```



```
[2]
✓ 5s
import cv2                 # OpenCV - ছবি/ভিডিও পড়া, প্রসেসিং
import mediapipe as mp     # Mediapipe - face mesh / landmarks
import numpy as np         # NumPy - সংখ্যার array, গণনা
from google.colab.output import eval_js  # Colab-এ JS ব্যবহার করে ক্যামেরা থেকে ছবি নিতে
import base64              # ক্যামেরা থেকে আসা ডাটা ডিকোড করার জন্য
import time                # টাইম, ডিলে, timestamp ইত্যাদির জন্য

/usr/local/lib/python3.12/dist-packages/jaxlib/plugin_support.py:71: RuntimeWarning: JAX plugin jax_cuda12_plugin version 0.7.2 is installed, but it is not compatible with the instal
  warnings.warn(
```

[4]
✓ 0s

```python
# ========================================
# lm_to_xy()
# Mediapipe landmark (x,y normalized 0-1) → ইমেজের পিক্সেল coordinate-এ রূপান্তর
# lm : landmark list
# w, h : ইমেজের width ও height
# ========================================
def lm_to_xy(lm, w, h):
    return np.array([(int(p.x * w), int(p.y * h)) for p in lm])
    # p.x*w → normalized থেকে pixel
    # p.y*h → normalized থেকে pixel


# ========================================
#  eye_EAR()
# EAR = Eye Aspect Ratio
# চোখ কতটা খোলা/বন্ধ তা মাপার একটি জনপ্রিয় ফর্মুলা
#
# EAR কমে গেলে = চোখ বন্ধের দিকে
# EAR বাড়লে = চোখ খোলা
#
# pts = চোখের ৬টি landmark পয়েন্ট
# A = vertical distance (p2 - p6)
# B = vertical distance (p3 - p5)
# C = horizontal distance (p1 - p4)
#
# EAR = (A + B) / (2 × C)
# ========================================
def eye_EAR(pts):
    A = np.linalg.norm(pts[1] - pts[5])  # উপর-নিচ দূরত্ব (p2-p6)
    B = np.linalg.norm(pts[2] - pts[4])  # উপর-নিচ দূরত্ব (p3-p5)
    C = np.linalg.norm(pts[0] - pts[3])  # ডানে-বামে দূরত্ব (p1-p4)

    # C = 0 হলে division error হবে, তাই চেক
    return (A + B) / (2.0 * C) if C != 0 else 0.0


# ========================================
# 🔹 mouth_MAR()
# MAR = Mouth Aspect Ratio
# MAR ↑ হলে মুখ খোলা (yawn) বেশি
# MAR ↓ হলে মুখ বন্ধ
#
# pts = ঠোঁটের landmark (4টি পয়েন্ট)
# top = উপর দিকের গড় পয়েন্ট
# bottom = নিচের গড় পয়েন্ট
#
# MAR = bottom - top (vertical distance)
# ========================================
def mouth_MAR(pts):
    top = np.mean([pts[0][1], pts[1][1]])      # উপরের দুই পয়েন্টের গড় Y
    bottom = np.mean([pts[2][1], pts[3][1]])   # নিচের দুই পয়েন্টের গড় Y
    return bottom - top
```

```
[5]
✓ 0s
from google.colab.output import eval_js
import base64

# ============================================
#  capture_frame()
# → Colab-এর ব্রাউজার ক্যামেরা থেকে ১টা ছবি নেওয়া
# → সেই ছবিটাকে base64 থেকে decode করে JPG ফাইল হিসেবে সেভ করা
# → আর তার filename return করা
# ============================================

def capture_frame():
    js = """
    async function captureFrame() {
        // ভিডিও element তৈরি
        const video = document.createElement('video');
        const stream = await navigator.mediaDevices.getUserMedia({video: true});
        video.srcObject = stream;
        document.body.appendChild(video);
        await video.play();

        //  ক্যামেরা স্টার্ট হওয়ার জন্য 2.5 sec অপেক্ষা
        await new Promise(resolve => setTimeout(resolve, 2500));

        // ক্যানভাস তৈরি (ভিডিও ফ্রেম কপি করার জন্য)
        const canvas = document.createElement('canvas');
        canvas.width = video.videoWidth;
        canvas.height = video.videoHeight;

        // ভিডিও → ক্যানভাসে আঁকা
        canvas.getContext('2d').drawImage(video, 0, 0);

        // ক্যামেরা বন্ধ করে দেওয়া
        stream.getTracks().forEach(track => track.stop());
        video.remove();

        // ক্যানভাস → Base64 encoded image output
        return canvas.toDataURL('image/jpeg', 0.8);
    }
    captureFrame()
    """

    # JS রান করে base64 image নেওয়া
    data = eval_js(js)
    binary = data.split(',')[1]

    filename = 'auto_frame.jpg'

    # base64 → jpg ফাইল সেভ
    with open(filename, "wb") as f:
        f.write(base64.b64decode(binary))

    return filename   # ফাইলের নাম রিটার্ন
```

```
[6]
✓ 0s
# ============================
#  EAR / MAR thresholds
# এগুলো তোমার EAR/MAR লাইভ ভ্যালু দেখে পরিবর্তন করতে হতে পারে
# ============================
# Threshold Values (MUST ADD)
MICRO_SLEEP_EAR = 0.18
MICRO_SLEEP_BLINK_FRAMES = 3

EAR_THRESHOLD_SLEEP = 0.20
EAR_THRESHOLD_DROWSY = 0.25
MAR_YAWN = 0.60
```

```
[ ]
Start coding or generate with AI.
```

```
[7]
✓ 0s
# ============================================
#  Mediapipe FaceMesh ইনিশিয়ালাইজ
# → মুখ detect করা
# → 468 landmark collect করা
# ============================================

face_mesh = mp_face_mesh.FaceMesh(
    max_num_faces = 1,          # শুধু ১টা মুখ নেবে
    refine_landmarks = True,    # iris landmarks সহ refined mesh
    min_detection_confidence = 0.5,
    min_tracking_confidence = 0.5
)
```

```
[8]
✓ 0s
# ============================================
#  Blink Counter + Micro-sleep Flag
# → blink_count : কতবার পরপর EAR কম (চোখ বন্ধ) পাওয়া গেছে
# → micro_sleep_flag : Micro-Sleep active কিনা
# ============================================

blink_count = 0         # পরপর কয়বার চোখ বন্ধ অবস্থায় পাওয়া যাচ্ছে (Ear < threshold)
micro_sleep_flag = False # Micro-Sleep শুরু হলে True হবে
```

```
[10]
✓ 58s
import time
from collections import defaultdict  # ← যোগ করা হয়েছে কাউন্টার এর জন্য

blink_count = 0
micro_sleep_flag = False

# মোট result রাখার জন্য
status_counts = defaultdict(int)

print("\033[96m Colab Unique Sleep Detection Started \033[0m")
print("Automatic frame capture every 5 sec")
print("Ctrl+C to stop")

while True:
    try:
        # Automatic frame capture
        filename = capture_frame()  # use automatic capture_frame()
        frame = cv2.imread(filename)
        h,w = frame.shape[:2]
        rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        res = face_mesh.process(rgb)

        status = 'NO_FACE'
        ear_val = 0.0
        mar_val = 0.0
        prob_score = 0
        gaze_alert = False

        if res.multi_face_landmarks:
            lm = res.multi_face_landmarks[0].landmark
            left = lm_to_xy([lm[i] for i in LEFT_EYE], w,h)
            right = lm_to_xy([lm[i] for i in RIGHT_EYE], w,h)
            ear_val = (eye_EAR(left)+eye_EAR(right))/2.0

            mar_val = mouth_MAR(lm_to_xy([lm[i] for i in UPPER_LIP], w,h))

            # Gaze detection
            eye_center = (left.mean(axis=0)+right.mean(axis=0))/2
            nose_tip = np.array([int(lm[1].x*w), int(lm[1].y*h)])
            if abs(eye_center[0]-nose_tip[0])/w > 0.08:
                gaze_alert = True
```

```
[10]
✓ 58s
            # Blink / micro-sleep
            if ear_val < MICRO_SLEEP_EAR:
                blink_count += 1
            else:
                blink_count = 0
                micro_sleep_flag = False

            if blink_count >= MICRO_SLEEP_BLINK_FRAMES:
                status = 'MICRO-SLEEP'
                micro_sleep_flag = True

            if not micro_sleep_flag:
                if ear_val < EAR_THRESHOLD_SLEEP:
                    status = 'SLEEPING'
                elif ear_val < EAR_THRESHOLD_DROWSY or mar_val > MAR_YAWN:
                    status = 'DROWSY'
                else:
                    status = 'AWAKE'

            if gaze_alert and status=='AWAKE':
                status = 'DROWSY'
                prob_score = 40

            #  DEBUG: EAR / MAR / STATUS দেখার জন্য
            print(f"DEBUG EAR: {ear_val:.3f}, MAR: {mar_val:.3f}, STATUS: {status}")

            # Probability score
            prob_score = {'AWAKE':0,'DROWSY':50,'MICRO-SLEEP':75,'SLEEPING':100}.get(status,0)

            # মোট কতবার কোন স্ট্যাটাস হয়েছে, তা count করা
            status_counts[status] += 1

            #  marks + color coding
            color_dict = {'AWAKE':'\033[92m','DROWSY':'\033[93m','MICRO-SLEEP':'\033[91m','SLEEPING':'\033[91m','NO_FACE':'\033[0m'}
            mark_dict = {'AWAKE':'✅','DROWSY':'⚠️','MICRO-SLEEP':'⚠️','SLEEPING':'❌','NO_FACE':'❌'}
            color = color_dict.get(status,'\033[0m')
            mark = mark_dict.get(status,'')
            print(f"{color}[{time.strftime('%H:%M:%S')}] Status: {status} {mark} | EAR: {ear_val:.2f} | MAR: {mar_val:.2f} | Prob: {prob_score}%\033[0m")

        else:
            status_counts["NO_FACE"] += 1   # face না থাকলেও count করব
            print(f"[{time.strftime('%H:%M:%S')}] No face detected ")
```

```
        else:
            status_counts["NO_FACE"] += 1   # face না থাকলেও count করব
            print(f"[{time.strftime('%H:%M:%S')}] No face detected ")

        time.sleep(5)  # automatic 5 sec interval

    except KeyboardInterrupt:
        print("Stopped by user")

        # 🔚 শেষের TOTAL RESULT
        print("\n========== TOTAL RESULT ==========")
        for s, c in status_counts.items():
            print(f"{s}: {c} times")
        print("==================================")

        break
```

```
...  Colab Unique Sleep Detection Started
Automatic frame capture every 5 sec
Ctrl+C to stop
DEBUG EAR: 0.257, MAR: -0.500, STATUS: AWAKE
[19:42:48] Status: AWAKE ✅ | EAR: 0.26 | MAR: -0.50 | Prob: 0%
DEBUG EAR: 0.169, MAR: 0.000, STATUS: SLEEPING
[19:42:56] Status: SLEEPING ❌ | EAR: 0.17 | MAR: 0.00 | Prob: 100%
DEBUG EAR: 0.201, MAR: 0.000, STATUS: DROWSY
[19:43:04] Status: DROWSY ⚠️ | EAR: 0.20 | MAR: 0.00 | Prob: 50%
DEBUG EAR: 0.183, MAR: 0.000, STATUS: SLEEPING
[19:43:13] Status: SLEEPING ❌ | EAR: 0.18 | MAR: 0.00 | Prob: 100%
DEBUG EAR: 0.195, MAR: 0.000, STATUS: SLEEPING
[19:43:21] Status: SLEEPING ❌ | EAR: 0.20 | MAR: 0.00 | Prob: 100%
DEBUG EAR: 0.235, MAR: 0.000, STATUS: DROWSY
[19:43:30] Status: DROWSY ⚠️ | EAR: 0.24 | MAR: 0.00 | Prob: 50%
DEBUG EAR: 0.271, MAR: 0.000, STATUS: AWAKE
[19:43:38] Status: AWAKE ✅ | EAR: 0.27 | MAR: 0.00 | Prob: 0%
Stopped by user

========== TOTAL RESULT ==========
AWAKE: 2 times
SLEEPING: 3 times
DROWSY: 2 times
==================================
```

221-35-903

ORIGINALITY REPORT

10%	8%	2%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

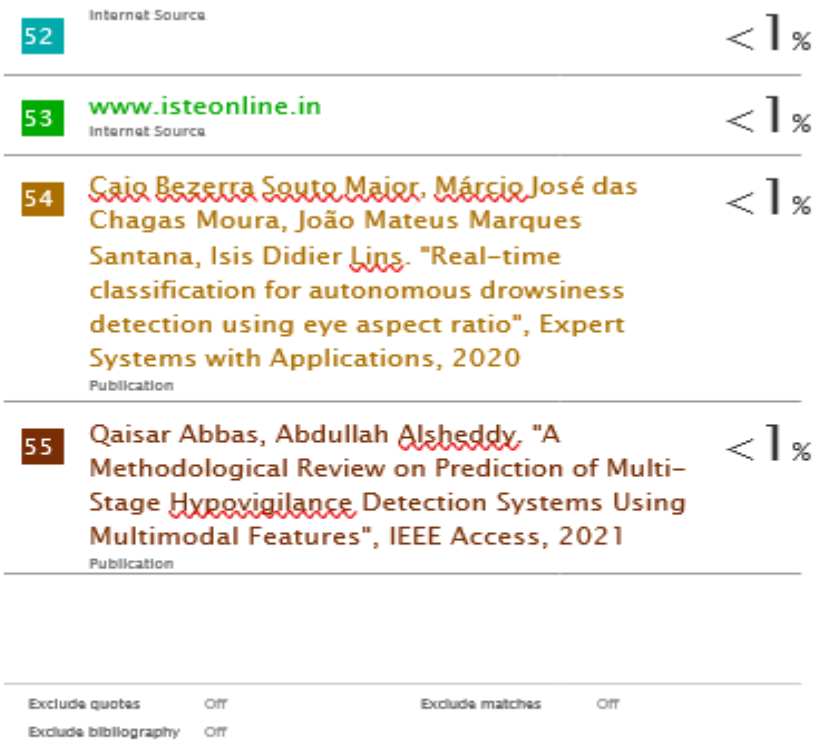
PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	3%
2	Submitted to Midlands State University Student Paper	2%
3	dspace.daffodilvarsity.edu.bd:8080 Internet Source	1%
4	Submitted to NCC Education Student Paper	1%
5	umpir.ump.edu.my Internet Source	<1%
6	publikasi.dinus.ac.id Internet Source	<1%
7	curin.chitkara.edu.in Internet Source	<1%
8	Submitted to KDU College Sdn Bhd Student Paper	<1%
9	Submitted to Mangalam College Of Engineering Student Paper	<1%
10	www.filibeto.org Internet Source	<1%
11	www.researchsquare.com Internet Source	<1%
12	www.mdpi.com Internet Source	<1%
13	S.P. Jani, M. Adam Khan. "Applications of AI in Smart Technologies and Manufacturing", CRC Press, 2025	<1%

	Publication	
14	iarjset.com Internet Source	< 1 %
15	www.coursehero.com Internet Source	< 1 %
16	Submitted to Fakultas Teknik Student Paper	< 1 %
17	Submitted to Middlesex University Student Paper	< 1 %
18	lirias.kuleuven.be Internet Source	< 1 %
19	Submitted to Computer Science Engineering Student Paper	< 1 %
20	Submitted to Dublin Business School Student Paper	< 1 %
21	Submitted to NIS Almaty-Medeu Student Paper	< 1 %
22	Submitted to Universidad Carlos III de Madrid - EUR Student Paper	< 1 %
23	Submitted to Universiti Malaysia Pahang Student Paper	< 1 %
24	Submitted to University of Bolton Student Paper	< 1 %
25	Li, Baopu, Can Yang, Qi Zhang, and Guoqing Xu. "Condensation-based multi-person detection and tracking with HOG and LBP", 2014 IEEE International Conference on Information and Automation (ICIA), 2014. Publication	< 1 %
26	Submitted to University of Westminster Student Paper	< 1 %
27	Van der Schyff, Marco. "Bandwidth Efficient Virtual Classroom", University of	< 1 %

	Johannesburg (South Africa), 2021 Publication	
28	arxiv.org Internet Source	< 1 %
29	ouci.dntb.gov.ua Internet Source	< 1 %
30	pdfsecret.com Internet Source	< 1 %
31	Hoda El Boussaki, Rachid Latif, Amine Saddik. "Drowsiness detection using Dlib: an overview", 2023 7th IEEE Congress on Information Science and Technology (CiSt), 2023 Publication	< 1 %
32	Pushpa Choudhary, Sambit Satpathy, Arvind Dagur, Dhirendra Kumar Shukla. "Recent Trends in Intelligent Computing and Communication", CRC Press, 2025 Publication	< 1 %
33	access.atis.org Internet Source	< 1 %
34	Submitted to Laureate Education Inc. Student Paper	< 1 %
35	Moe, Kyaw Hlwan. "Designing a Usable Mobile Application for Field Data Collection", University of the Witwatersrand, Johannesburg (South Africa), 2025 Publication	< 1 %
36	Submitted to Universiti Pertahanan Nasional Malaysia Student Paper	< 1 %
37	core.ac.uk Internet Source	< 1 %
38	www.springerprofessional.de Internet Source	< 1 %

39	Lecture Notes in Computer Science, 2015. Publication	< 1 %
40	jozilla.net Internet Source	< 1 %
41	link.springer.com Internet Source	< 1 %
42	www.ijcrt.org Internet Source	< 1 %
43	"Universal Threats in Expert Applications and Solutions", Springer Science and Business Media LLC, 2026 Publication	< 1 %
44	Submitted to Monash University Student Paper	< 1 %
45	Nawawi, Abdul Hadi. "Knowledge Elicitation Methodology from Multiple Experts for Rating Valuation by the Comparison Method for Commercial and Industrial Properties in Malaysia: Volumes 1 and 2", University of South Wales (United Kingdom), 2020 Publication	< 1 %
46	docsbay.net Internet Source	< 1 %
47	journals.scholarpublishing.org Internet Source	< 1 %
48	learn.scu.edu.au Internet Source	< 1 %
49	library.binus.ac.id Internet Source	< 1 %
50	ndl.ethernet.edu.et Internet Source	< 1 %
51	norma.ncirl.ie Internet Source	< 1 %
	support.openecu.com	


52
Internet Source
< 1 %
53
www.isteonline.in
Internet Source
< 1 %
54
Caio Bezerra Souto Maior, Márcio José das Chagas Moura, João Mateus Marques Santana, Isis Didier Lins. "Real-time classification for autonomous drowsiness detection using eye aspect ratio", Expert Systems with Applications, 2020
Publication
< 1 %
55
Qaisar Abbas, Abdullah Alsheddy. "A Methodological Review on Prediction of Multi-Stage Hypovigilance Detection Systems Using Multimodal Features", IEEE Access, 2021
Publication
< 1 %
Exclude quotes Off
Exclude matches Off
Exclude bibliography Off