An
Industry Oriented Mini Project Report on

VIRTUAL EYE GAZE TRACKING USING COMPUTER VISION

Submitted In Partial Fulfillment of the Requirements for the Award of Degree

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

Submitted By

KHUSHI VERMA	**217Z1A1225**
SAJJAL SWATHI	**217Z1A1250**
KOMMULA AKSHAY	**227Z5A1205**

Under the Guidance of
Mrs. P. Santhi Priya
Assistant Professor

SCHOOL OF ENGINEERING
Department of Information Technology

NALLA NARASIMHA REDDY
EDUCATION SOCIETY'S GROUP OF INSTITUTIONS
(AN AUTONOMOUS INSTITUTION)
Approved by AICTE, New Delhi, Chowdariguda (V) Korremula'x' Roads,
Via Narapally, Ghatkesar (Mandal)Medchal (Dist), Telangana-500088

2024-2025

NALLA NARASIMHA REDDY
Education Society's Group of Institutions - Integrated Campus
(UGC AUTONOMOUS INSTITUTION)

SCHOOL OF ENGINEERING
DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project report titled **"VIRTUAL EYE GAZE TRACKING USING COMPUTER SYSTEM"** is being submitted by **Khushi Verma (217Z1A1225), Sajjal Swathi (217Z1A1250)** and **Kommula Akshay (227Z5A1205)** in Partial fulfillment of the requirements for the award of **Bachelor of Technology in Information Technology** is a record bonafide work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree.

Internal Guide
(Mrs. P. Santhi Priya)

Head of the Department
(Dr. K. Rameshwaraiah)

Submitted for Viva voce Examination held on…….

External Examiner

DECLARATION

We Khushi Verma Sajjal Swathi Kommula Akshay the students of **Bachelor of Technology in Information Technology, Nalla Narasimha Reddy Education Society's Group of Institutions**, Hyderabad, Telangana, hereby declare that the work presented in this project work entitled **Virtual Eye Gaze Using Computer Vision** is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of engineering ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning.

KHUSHI VERMA	217Z1A1225
SAJJAL SWATHI	217Z1A1250
KOMMULA AKSHAY	2275ZA1205

Date:

Signature:

ACKNOWLEDGEMENT

We express our sincere gratitude to our guide **Mrs. P. Santhi Priya,** Assistant Professor in Department of Computer Science and Engineering, NNRESGI, who motivated throughout the period of the project and also for her valuable and intellectual suggestions, guidance, and constant encouragement right throughout our work.

We would like to express our profound gratitude to our mini project In-charge **Mrs. Priyanka Pandarinath,** Assistant Professor in Department of Computer Science and Engineering, NNRESGI, for her support and guidance in completing our project and for giving us this opportunity to present the project work.

We profoundly express thanks to **Dr. K. Rameshwaraiah,** Professor & Head, Department of Computer Science and Engineering, NNRESGI, for his cooperation and encouragement in completing the project successfully.

We wish to express our sincere thanks to **Dr. G. Janardhana Raju,** Dean School of Engineering, NNRESGI, for providing the facilities for completion of the project.

We wish to express our sincere thanks to **Dr**. **C.V. Krishna Reddy,** Director, NNRESGI, for providing the facilities for completion of the project.

Finally, we would like to thank overall mini-project coordinator, members of Project Review Committee (PRC), all the faculty members and supporting staff, Department of Computer Science and Engineering, NNRESGI, for extending their help in all circumstances.

By

KHUSHI VERMA 217Z1A1225
SAJJAL SWATHI 217Z1A1250
KOMMULA AKSHAY 217Z1A1205

ABSTRACT

Eye Controlled Mouse is an innovative project designed to enable mouse cursor control using human eye movements. By leveraging a webcam and computer vision techniques, this project captures eye movements in real-time and translates them into corresponding mouse actions. The purpose of this project is to provide a hands-free and intuitive interface for individuals with mobility impairments, offering an alternative input method for computer interaction. The method involves capturing eye movements through webcam footage and mapping them to mouse cursor coordinates using the Python programming language. Key libraries utilized include OpenCV for webcam input processing, Media Pipe for facial landmark detection to assist in eye tracking, and PyAutoGUI for mouse control automation. Additionally, the project employs the WMI (Windows Management Instrumentation) module to automatically set the screen brightness to 100% when the program runs, ensuring optimal visibility for accurate eye tracking

Keywords: OpenCV, PyAutoGUI, WMI.

TABLE OF CONTENTS

Page No.

LIST OF FIGURES

LIST OF TABLES

List of Abbreviations

S.No.	Synonym	Abbreviation
1.	HCI	Human-Computer Interaction
2.	HTML	Hyper Text Markup Language
3.	API	Application Programming Interface
4.	OpenCV	Open-Source Computer Vision Library
5.	WMI	Windows Management Instrumentation
6.	OS	Operating System
7.	DFD	Data Flow Diagram
8.	MS	Multiple Sclerosis
9.	MAX	Maximum
10.	URL	Uniform Resource Locators
11.	UML	Unified Modelling Language
12.	EOG	Electrooculography

1. INTRODUCTION

In recent years, human-computer interaction (HCI) has significantly advanced, driven by sophisticated technologies and innovative approaches to improve accessibility and usability. One prominent development is the integration of eye-tracking and facial recognition technologies, enabling intuitive and hands-free control mechanisms. This mini project exemplifies such technological convergence, utilizing computer vision and automation to create a system that allows users to control their computer mouse with eye movements.

The core of this project leverages several powerful tools: OpenCV, Media Pipe, PyAutoGUI, and WMI. OpenCV (Open-Source Computer Vision Library) is renowned for real-time image processing and computer vision applications. Media Pipe, developed by Google, provides robust solutions for real-time machine learning, particularly in facial and hand landmark detection. PyAutoGUI is a versatile Python module for cross-platform GUI automation, enabling programmatic control of the mouse and keyboard. Lastly, WMI (Windows Management Instrumentation) facilitates system management tasks such as adjusting screen brightness and ensuring the system adapts to user needs.

This system's primary functionality involves tracking the user's eye movements through a webcam, processing the captured video frames to detect facial landmarks, and translating these movements into cursor actions on the screen. By focusing on specific eye landmarks, the program determines gaze direction and simulates mouse movements accordingly. Additionally, the project incorporates a feature to simulate mouse clicks based on eyelid movements, providing a seamless and immersive user experience.

Implementation begins with setting the screen brightness to an optimal level, ensuring environmental conditions do not hinder the accuracy of facial and eye landmark detection. The core functionality, encapsulated in the “control_mouse_with_eyes” function, involves capturing real-time video input, processing it to extract relevant facial landmarks, and using these landmarks to control the cursor position on the screen.

This mini project represents a significant step forward in making computer interactions more accessible, especially for individuals with mobility impairments. By harnessing the power of modern computer vision and automation libraries, it demonstrates a practical application of HCI technologies that can significantly enhance user autonomy and interaction efficiency. The following sections will delve into the specific components and functionality of the code, providing a detailed walkthrough of its implementation and potential applications.

Overall, the project highlights the transformative potential of integrating eye-tracking and facial recognition technologies into everyday computer use. By offering a hands-free control mechanism, it not only improves accessibility but also showcases the versatility and power of tools like OpenCV, Media Pipe, PyAutoGUI, and WMI. This approach paves the way for more inclusive and efficient user interfaces, setting the stage for future innovations in human-computer interaction.

1.1 MOTIVATION

The rapid evolution of technology in recent decades has brought about a transformative shift in how humans interact with computers and digital devices. As technology becomes more integrated into daily life, the need for more intuitive and accessible interaction methods becomes increasingly important. Traditional input devices like keyboards and mice, while effective, are not universally accessible. For individuals with physical disabilities or motor impairments, these devices can pose significant barriers to effective computer use. This project, focused on eye-controlled mouse navigation, is motivated by the desire to break down these barriers and create a more inclusive digital environment.

The primary motivation for this project stems from the need to improve accessibility for individuals with limited mobility. Conditions such as amyotrophic lateral sclerosis (ALS), multiple sclerosis (MS), spinal cord injuries, and other physical disabilities can severely restrict a person’s ability to use standard input devices. By developing a system that utilizes eye movements for cursor control, we aim to provide these individuals with a

viable alternative for interacting with their computers, thereby enhancing their independence and quality of life.

Eye-tracking technology, combined with advances in computer vision and machine learning, presents a promising solution to this challenge. By capturing and interpreting eye movements, it is possible to emulate the functionality of a mouse, allowing users to navigate and interact with their computer environment hands-free. This technology not only makes computer use possible for those who might otherwise be unable to use it but also opens up new possibilities for all users, offering a more natural and seamless way to interact with digital devices.

1.2 PROBLEM STATEMENT

The increasing reliance on digital devices for work, education, and daily activities presents significant challenges for individuals with physical disabilities, particularly those with limited hand mobility or conditions like paralysis. Traditional input devices such as keyboards and mice are often inaccessible to these users, creating barriers to their full participation in the digital world. While existing accessibility solutions offer some relief, they may be limited, costly, or not sufficiently intuitive.

To address this issue, this project aims to develop a hands-free computer control system that enables users to interact with their computers using eye movements. By leveraging computer vision techniques, the system will track eye movements and blinks, translating them into mouse movements, clicks, and scrolling actions. Additionally, the system will include functionality for adjusting screen brightness based on user preferences. The solution will be designed to work with standard webcams, ensuring it is both cost-effective and easy to implement.

The main challenges of this project include ensuring accurate and responsive eye-tracking, minimizing false clicks from unintentional blinks, and maintaining performance across various lighting conditions. The goal is to create a seamless and intuitive user experience that significantly enhances computer accessibility for individuals with physical disabilities, empowering them to navigate digital environments with greater independence.

1.3 OBJECTIVE OF PROJECT

The objective of this project is to design and implement a hands-free computer control system that enables users to interact with their computers using eye movements and blinks. This system aims to provide an accessible solution for individuals with physical disabilities, particularly those with limited or no hand mobility, allowing them to perform essential computer tasks without the need for traditional input devices like a mouse or keyboard.

The project will focus on developing a robust and intuitive interface that uses computer vision techniques to track eye movements and translate them into mouse cursor movements, clicks, and scroll commands. Additionally, the system will include features such as automatic screen brightness adjustment based on user preferences. The ultimate goal is to create a cost-effective, easy-to-use solution that enhances digital accessibility and empowers users with disabilities to navigate and interact with their computers independently.

1.4 LIMITATION OF PROJECT

The effectiveness of the hands-free control system is closely tied to the accuracy of eye-tracking technology. Eye-tracking relies on precise detection of the user's gaze and eye movements to translate them into corresponding cursor movements and commands on the screen. However, various factors can impact this accuracy. For instance, changes in lighting conditions can create shadows or glare, making it difficult for the camera to consistently detect eye positions. Similarly, the quality of the webcam plays a significant role; lower resolution or frame rate can result in delayed or imprecise tracking, leading to cursor drift or missed actions. User positioning, such as sitting too close or too far from the camera, can also affect tracking precision. These variations can introduce inaccuracies, potentially frustrating users who require smooth and reliable control.

2. LITERATURE SURVEY

2.1 INTRODUCTION

A literature survey is a body of text that aims to review the critical points of current knowledge on and/or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period. Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and may be just a simple summary of sources. Usually, it has an organizational pattern and combines both summary and synthesis.

Review of Literature Survey

Title: Eye-Tracking Technology: A Historical Overview and Current Trends
Author: Majaranta, P., & Bulling, A.

This paper provides a comprehensive historical overview of eye-tracking technology, tracing its roots from early research in the 19th century to modern-day applications. The authors detail the progression of eye-tracking systems, from basic mechanical setups to sophisticated, high-precision digital systems. The paper explores various methods of eye-tracking, such as electrooculography (EOG), video-based techniques, and infrared tracking. It also delves into the diverse applications of eye-tracking across fields like psychology, marketing, and assistive technology, where it has empowered individuals with disabilities to communicate and interact with computers. The authors highlight challenges, such as the need for accurate calibration, and discuss current trends like the integration of eye-tracking in wearable devices and virtual reality systems, which promise to expand the technology's accessibility and utility.

Title: Human-Computer Interaction Using Eye Movements: A Review
Author: Hansen, D. W., & Ji, Q.

This review paper provides a technical deep dive into the methods used for eye-tracking in human-computer interaction (HCI). The authors explore various technologies, such as infrared light tracking, video-based tracking, and EOG, comparing their effectiveness in different scenarios. They discuss the challenges of integrating eye-tracking into mainstream HCI, particularly regarding issues like latency, false positives, and user comfort. The paper also covers the use of eye-tracking in combination with other input modalities, such as gesture recognition and voice commands, to create more intuitive and effective interaction systems. The authors conclude by emphasizing the need for continued research into reducing costs and improving the accuracy and ease of use of eye-tracking systems.

Title: Blink-Based Communication and Control: A Survey of Current Systems and Future Prospects
Author: Arefin, M. S., & Islam, M. K.

This survey paper focuses on blink-based control systems, which are particularly useful for individuals with severe motor disabilities. The authors review different techniques for detecting blinks, such as using EOG sensors and computer vision, and discuss how these systems can be used for communication and control tasks. The paper highlights the simplicity and low physical effort required by blink-based systems but also identifies significant challenges, such as distinguishing between intentional and unintentional blinks, which can lead to false positives. The authors propose that future systems could benefit from integrating machine learning algorithms that can learn to recognize individual user patterns, improving both the accuracy and usability of these systems.

2.2 EXISTING SYSTEM

The existing system for eye-controlled interaction predominantly relies on eye-tracking technology, which has been explored and developed in various forms over the years. These systems utilize cameras, sensors, and sophisticated algorithms to track and interpret the user's eye movements, enabling them to control a computer or device hands-free. Below are the key components and examples of existing systems in this domain.

- Hardware: Most existing eye-tracking systems rely on specialized hardware, such as infrared cameras or high-definition webcams, to capture eye movements. These systems often require the user to remain within a specific range and position for accurate tracking. Notable companies like Tobii and EyeTech have developed commercial eye-tracking devices that are widely used in research, gaming, and accessibility.

- Eye-tracking technology has also been integrated into gaming, providing an immersive experience by allowing users to control game elements with their eyes. For example, Tobii's eye-tracking technology has been integrated into games like "Assassin's Creed" and "Watch Dogs," where players can aim, navigate menus, and control the camera using their gaze. Gaze-based control is also being explored in virtual reality (VR) and augmented reality (AR), where it enhances interaction within virtual environments.

- Smart Glasses: Wearable devices like smart glasses integrate eye-tracking technology to enable hands-free control in everyday tasks. Google Glass and Vuzix Blade are examples of smart glasses that incorporate eye-tracking for enhanced user interaction. These devices are used in various fields, including healthcare, where doctors can access patient information or control medical devices with their gaze.

- Screen Readers and Eye-Controlled Typing: For users with motor impairments, eye-controlled typing systems like OptiKey and Dasher allow text input via gaze control. These systems enable users to select letters or words on a virtual keyboard by focusing their gaze on the desired characters.

2.2.1 Constraints of Existing System

- Accuracy and Calibration: Current systems often require frequent calibration to maintain accuracy, and their performance can be affected by lighting conditions, user positioning, and the quality of the camera improved.
- Hardware Dependency: The reliance on specialized hardware can make these systems costly and limit their accessibility to users with outdated or low-quality devices.
- User Fatigue and Comfort: Prolonged use of eye-tracking systems can lead to discomfort and eye fatigue, making them unsuitable for extended use.

2.3 PROPOSED SYSTEM

The proposed system aims to advance the capabilities of eye-controlled interaction by integrating a series of innovative features designed to enhance user experience, accessibility, and functionality. This system builds upon existing eye-tracking technology by incorporating additional functionalities such as scrolling control, customizable user profiles, and advanced gesture recognition, while addressing the limitations observed in current systems.

Enhanced Eye-Tracking Accuracy and Adaptability: To overcome issues related to accuracy and calibration, the proposed system utilizes advanced machine learning algorithms and improved computer vision techniques. These innovations enhance the precision of gaze detection, even under varying lighting conditions and with different webcam qualities. The system features adaptive calibration processes that adjust based on the user's eye movements and environmental factors improving overall performance.

Integrated Scrolling Control: A notable feature of the proposed system is its scrolling control capability, which enables users to scroll through documents or web pages using vertical eye movements. By analyzing the direction and magnitude of vertical eye shifts, the system translates these movements into scrolling actions, allowing for a more intuitive and hands-free navigation experience. This feature is particularly beneficial for users with mobility impairments who find traditional scrolling methods challenging.

Advanced Gesture Recognition: The proposed system integrates advanced gesture recognition to expand its functionality beyond basic gaze and blink commands. By employing sophisticated algorithms to detect and interpret a variety of eye movements and facial expressions, the system enables users to perform complex actions through gestures.

Improved Accessibility and User Comfort: To address issues of user fatigue and hardware dependency, the proposed system is designed with ergonomic considerations in mind. The hardware components, including the camera and mounting system, are optimized for comfort and ease of use, minimizing strain during extended sessions. Additionally, the system is compatible with a range of standard webcams and devices, ensuring broader accessibility without requiring specialized equipment.

Enhanced Software Interface: The software interface of the proposed system is user-friendly and intuitive, incorporating visual feedback and real-time adjustments to enhance interaction. Users receive immediate feedback on their gaze inputs and can adjust settings through a simple and accessible interface. The software also supports integration with popular applications and platforms, allowing for seamless control of various functions such as web browsing, document editing, and communication tools.

2.3.1 Advantages

- Supports hands-free interaction for individuals with severe physical disabilities, increasing their independence.
- Provides a natural, intuitive control method, reducing physical strain from traditional input devices.
- Enhances gaming and virtual reality experiences with gaze-based control, making interactions more immersive.
- Offers detailed insights into user behavior and cognitive processes, aiding in user interface design and marketing.
- Features customizable profiles and adaptable settings, compatible with various applications and platforms.

3. SYSTEM ANALYSIS

It is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components. System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. Itis a problem-solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose. Analysis specifies what the system should do.

3.1 INTRODUCTION

In this phase the requirements are gathered and analyzed. User's requirements are gathered in this phase. This phase is the main focus of the users and their interaction with the system.

- What is being done?
- How is it being done?
- Who is doing it?
- When is he doing it?
- Why is it being done?
- How can it be improved?

These general questions are answered during a requirement gathering phase. After requirement gathering these requirements are analyzed for their validityand the possibility of incorporating the requirements in the system to be developmentis also studied.

3.2 SOFTWARE REQUIREMENTS

The requirement can be defined as a high-level abstract statement or a detailed mathematical functional specification of a system's services, functions, and constraints. They are depictions of the characteristics and functionalities of the target system. Requirements denote the expectations of users from the software product.

The requirement should be open to interpretation and detailed enough to understand. It is essential to know about software requirements because it minimizesthe developer's time and effort and the development cost.

- Operating System : Above windows 7
- Coding Language : Python.
- Framework : OpenCV, PyAutoGUI.
- Tool : PyCharm.

3.3 HARDWARE REQUIREMENTS

The hardware requirements are the requirements of a hardware device. Most hardware only has operating system requirements or compatibility. For example, a printer may be compatible with Windows XP but not compatible with newer versions of Windows like Windows 10, Linux, or the Apple mac OS.

If a hardware device is not compatible with your computer, it is up to the manufacturer to release drivers. Unfortunately, many manufacturers only release updated drivers to fix problems with older drivers and often do not release drivers for newer operating systems or alternative operating systems. If a hardware device doesn't have drivers for your operating system, the only solution may be to get a more up-to-date replacement device.

- Processor : Pentium–IV/III
- RAM : 2GB (min)
- Hard Disk : 80 GB (min)

3.4 CONTENT DIAGRAM OF PROJECT

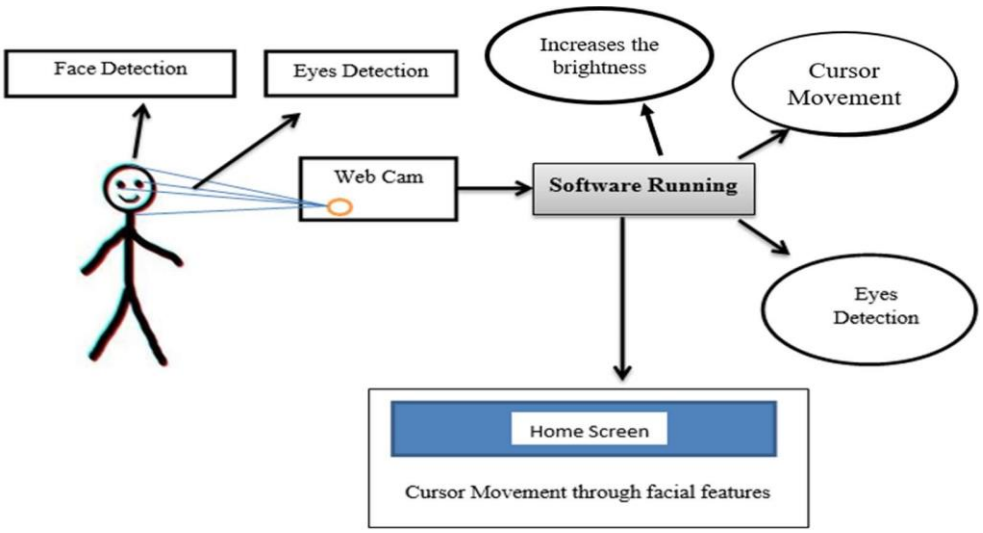

Figure 3.4: Architecture of project

4. SYSTEM DESIGN

4.1 INTRODUCTION

Software design is the process by which an agent creates a specification of a software artifact, intended to accomplish goals, using a set of primitive components and subject to constraints. Software design may refer to either all the activity involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems or the activity following requirements specification and before programming, as in a stylized software engineering process. “Software design usually involves problem solving and planning a software solution. This includes both a low-level component design and a high-level, architecture design. Design is the first step in the development phase for any techniques and principles for the purpose of defining a device, a processor system insufficient detail to permit its physical realization.

Once the software requirements have been analyzed and specified the software design involves four technical activities – design, coding, implementation and testing that are required to build and verify the software.

The design activities are of main importance in this phase, because in this activity, decisions ultimately affecting the success of the software implementation and its ease of maintenance are made. These decisions have the final bearing upon reliability and maintainability of the system. Design is the only way to accurately translate the customer’s requirements into finished software or a system.

4.2 DATA FLOW DIAGRAM

A data flow diagram (DFD) is a visual representation of the information flow through a process or system. DFDs help you better understand process or system operations to discover potential problems, improve efficiency, and develop better processes. They range from simple overviews to complex, granular displays of a process or system. There are two types of DFDs — logical and physical. Logical diagrams display the theoretical process of moving information through a system, like where the data comes from, where it goes, how it changes, and where it ends up. Physical diagrams show you the practical process of moving information through a system. It can show how your system's specific software, hardware, files, employees, and customers influence the flow of information.

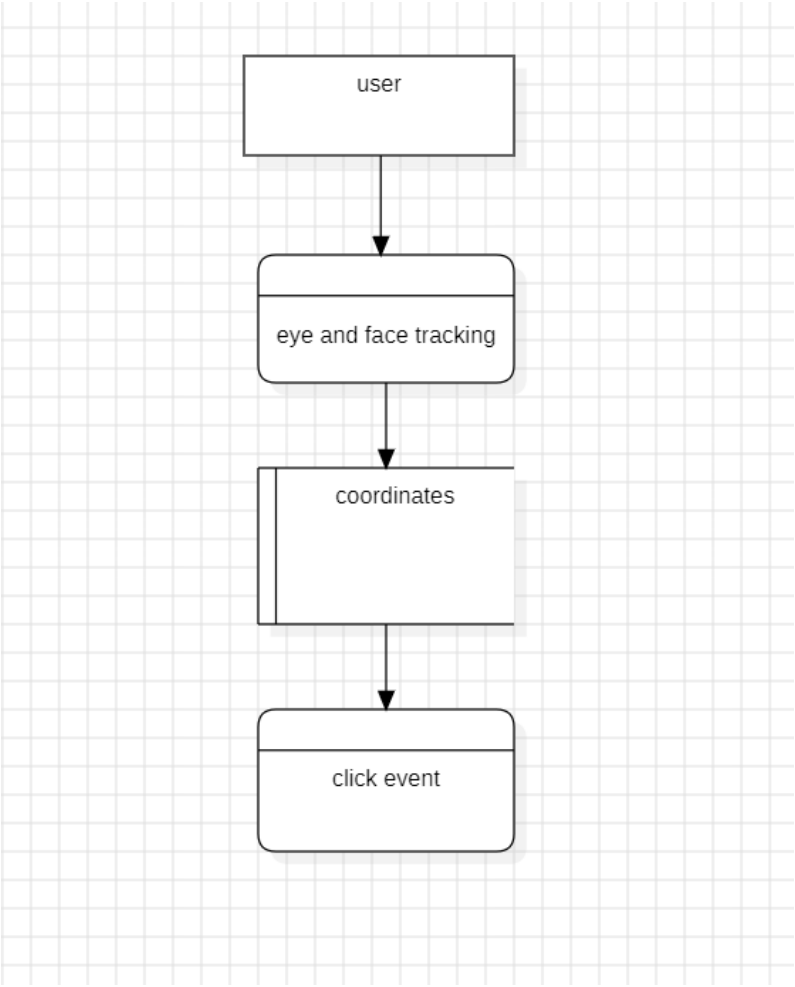

Figure 4.2: Data flow diagram

4.3 USE CASE DIAGRAM

A UML use case diagram is the primary form of system/software requirements for a new software program underdeveloped. In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. It is useful in the following situations: Scenarios in which your system or application interacts with people, organizations, or external systems, Goals that your system or application helps those entities (known as actors) achieve, the scope of your system. Use cases specify the expected behavior (what), and not the exact method of making it happens (how).

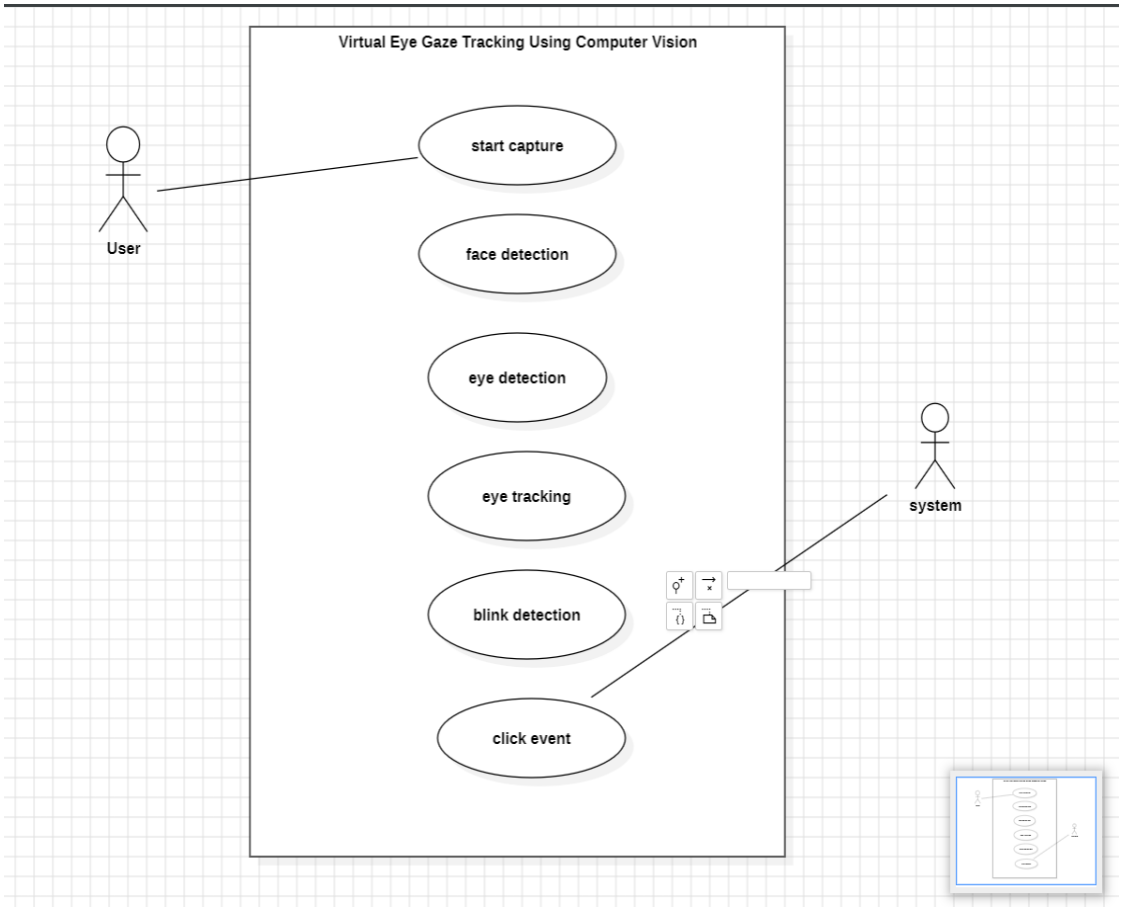

Figure 4.3: Use case Diagram

4.4 CLASS DIAGRAM

Class diagram is a static diagram. It represents the static view of an application. The class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. The class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

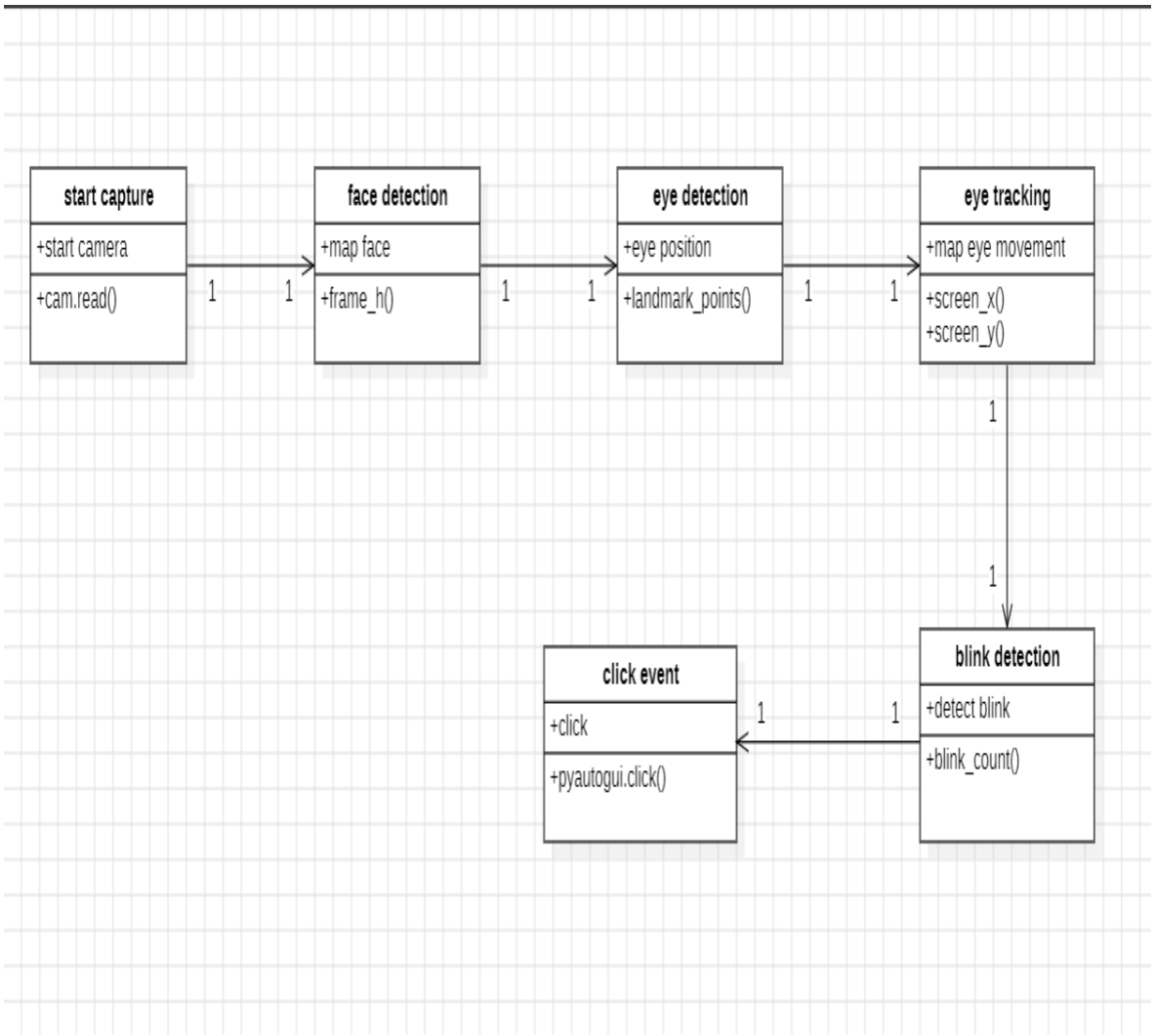

Figure 4.4: Class Diagram

4.5 SEQUENCE DIAGRAM

UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of collaboration. A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

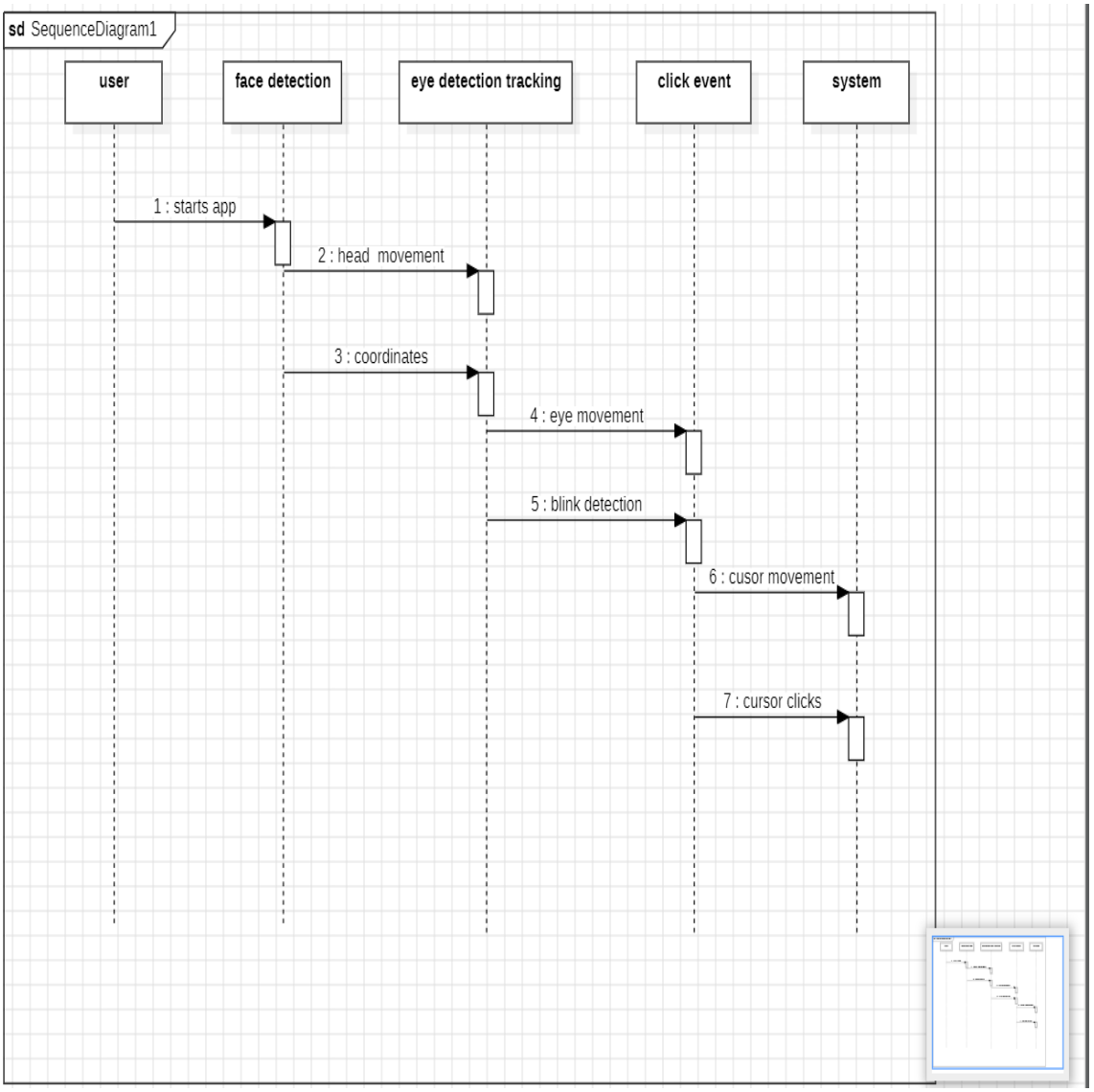

Figure 4.5: Sequence Diagram

4.6 ACTIVITY DIAGRAM

The Unified Modeling Language includes several subsets of diagrams, including structure diagrams, interaction diagrams, and behavior diagrams. Activity diagrams, along with use case and state machine diagrams are considered behavior diagrams because they describe what must happen in the system being modeled. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

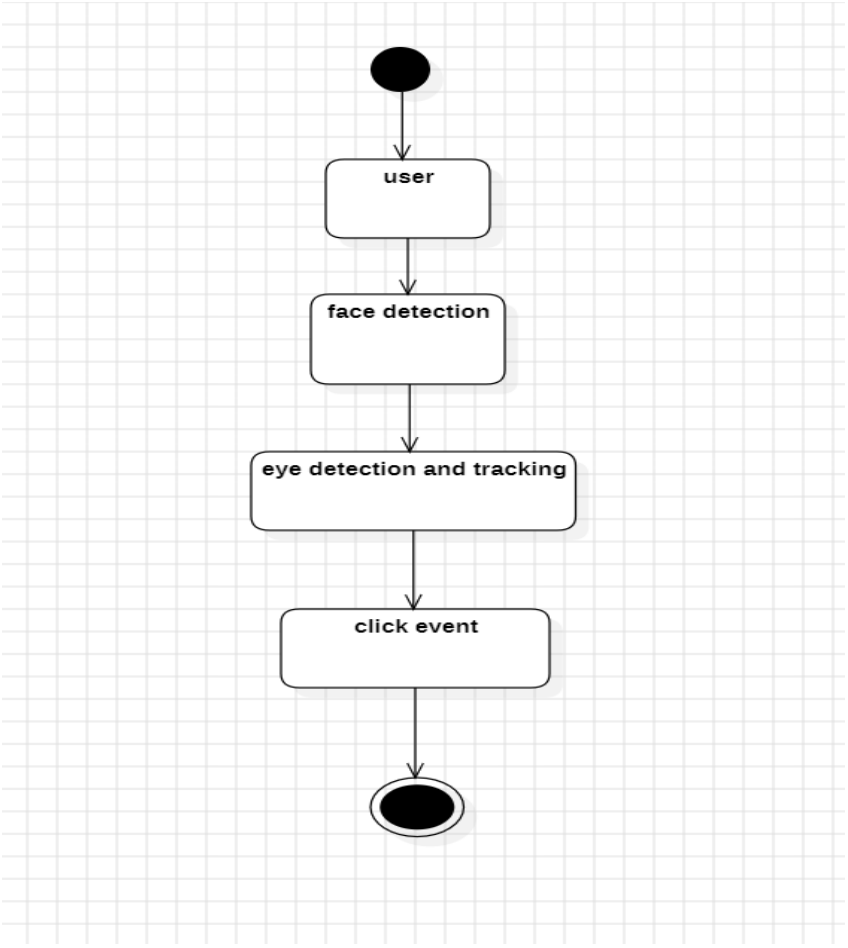

Figure 4.6: Activity Diagram

5. IMPLEMENTATION & RESULTS

5.1 INTRODUCTION

OpenCV

OpenCV (Open Source Computer Vision Library) is a powerful open-source library primarily designed for computer vision tasks. It enables real-time image and video processing, making it suitable for applications such as facial recognition, object detection, augmented reality, and more. Developed by Intel, OpenCV is known for its efficiency and extensive functionality, providing tools for both basic image manipulation and advanced computer vision tasks.

OpenCV Features

- **Extensive Library of Algorithms:** Comprehensive Algorithm Collection OpenCV offers over 2500 optimized algorithms that cover a wide range of computer vision tasks, from image processing to complex machine learning operations. Some common tasks include object detection (e.g., detecting faces, cars, etc.), image segmentation, motion analysis, and tracking. Support for Classical and State-of-the-Art Techniques**: Whether you are implementing classical computer vision methods like edge detection (e.g., Canny Edge Detector) or leveraging deep learning-based models for object recognition, OpenCV provides built-in tools for both.

- **Real-Time Processing:** Optimized for Performance OpenCV is optimized for real-time applications, making it highly efficient when dealing with live video streams or processing large batches of images. Its optimization for Intel processors (and other hardware) ensures that applications run faster. Support for Multiple Data Sources OpenCV can process images and video from various sources, including cameras, video files, and network streams, making it suitable for surveillance, robotics, and other real-time applications.

5.2 METHOD OF IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus, it can be the most critical stage in achieving a successful new system and in giving the user confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve change over and evaluation of change over methods

Modules

1. PyAutoGUI

PyAutoGUI is a Python library designed to control and automate interactions with your computer's graphical user interface (GUI). It provides functions to simulate mouse movements, clicks, and keyboard typing, allowing you to write scripts that mimic user input. This makes PyAutoGUI particularly useful for automating repetitive tasks that involve interacting with desktop applications or GUIs that lack native APIs.

- **Mouse Control:**
  - Move the mouse to any position on the screen.
  - Click, double-click, right-click, middle-click, and drag the mouse.
  - Perform scroll actions (both horizontal and vertical).

- **Keyboard Control:**
  - Type strings or single keystrokes.
  - Simulate hotkeys and special keys like Ctrl, Shift, Enter, etc.
  - Press and hold keys or key combinations.

- **Screenshot and Image Recognition:**
    - Take screenshots of the screen or part of it.
    - Locate elements on the screen by matching images (useful for locating buttons or icons).
    - Wait for a specific image to appear before continuing the script.
- **Failsafe and Pause:**
    - Failsafe: Move the mouse to a corner of the screen to stop the script immediately in case of unintended behavior.
    - Pause: You can set a pause between each action to slow down the automation for better observation.
- **Alert Messages:** Display simple alert messages or confirmation boxes.
- **Cross-Platform Support:** PyAutoGUI works on Windows, macOS, and Linux, making it versatile for various systems.

2. Time

- **Time Representation**

Time is often represented as the number of seconds since the epoch (January 1, 1970, 00:00:00 UTC). This is known as "Unix time" or "POSIX time. Local Time: Functions allow conversion between Unix time and a more human-readable format like a date or a timestamp.

- **Getting the Current Time**
    - time.time(): Returns the current time in seconds since the epoch as a floating-point number.
    - time.localtime(): Converts the time (seconds since epoch) to a struct that represents the local time.
    - time.gmtime(): Similar to `localtime()`, but returns the time in UTC.

3. MediaPipe

MediaPipe is an open-source framework developed by Google that simplifies the development of machine learning pipelines for multimodal (audio, video, sensor) data. It is particularly useful for tasks like gesture recognition, facial landmarks detection, hand tracking, object detection, and pose estimation, providing real-time performance across multiple platforms, including mobile (Android and iOS), web, and desktop systems. MediaPipe supports Python, C++, and JavaScript, making it versatile for different application needs.

One of the standout features of MediaPipe is its pre-built solutions for computer vision tasks. These include hand tracking, face mesh, pose estimation, holistic models (which combine face, hand, and pose tracking), object detection, and selfie segmentation. For example, the Hand Tracking model provides 21 landmark points for real-time hand gesture recognition, while the Pose model tracks full-body keypoints with 33 landmarks, both optimized for smooth performance on mobile devices. In addition, MediaPipe supports 3D object detection and face landmark detection, which is useful in applications like augmented reality and virtual makeup.

At its core, MediaPipe uses a graph-based framework, where each component in the pipeline is represented as a node, and data flows through edges connecting these nodes. This modular design allows developers to customize, reuse, and extend parts of pipelines as needed. The framework can handle multimodal data, enabling simultaneous processing of audio, video, and sensor inputs, which is ideal for complex tasks like gesture control and emotion detection.

MediaPipe is optimized for real-time processing, making it highly suitable for interactive applications such as AR/VR, gaming, fitness apps, and real-time video effects. It supports integration with TensorFlow Lite for on-device machine learning tasks and TensorFlow.js for web-based applications. Developers can also build custom pipelines, integrating their own machine learning models or using pre-existing solutions from MediaPipe’s library.

5.3 EXPLANATION OF KEY FUNCTIONS

Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted:** Python is processed at runtime by the interpreter. You donot need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive:** You can actually sit at a Python prompt and interact withthe interpreter directly to write your programs.
- **Python is Object-Oriented:** Python supports Object-Oriented style or techniqueof programming that encapsulates code within objects.
- **Python is a Beginner's Language:** Python is a great language for the beginner level. Programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

WMI (Windows Management Instrumentation):

Windows Management Instrumentation (WMI) is a Microsoft technology that provides a standardized way to access and manage system resources, both hardware and software, on Windows operating systems. WMI serves as a crucial tool for administrators and developers to interact with system components and services, offering capabilities such as system monitoring, configuration, and automation. Through WMI, users can query information, manage processes, control system events, and even perform administrative tasks on both local and remote machines. This powerful framework allows for deep integration into the system, making it essential for both system management and enterprise-level automation.

WMI excels at gathering detailed information about the system’s hardware, operating system, and installed software. It can query virtually any system component, including the CPU, memory, disk drives, network adapters, operating system versions, and more.

SOURCE CODE

Main.py

```
import cv2
import mediapipe as mp
import pyautogui
import wmi
import time

# Function to set brightness
def set_brightness(level):
    brightness = wmi.WMI(namespace='wmi')
    methods = brightness.WmiMonitorBrightnessMethods()[0]
    methods.WmiSetBrightness(level, 0)

def control_mouse_with_eyes():
    cam = cv2.VideoCapture(0)
    face_mesh = mp.solutions.face_mesh.FaceMesh(refine_landmarks=True)
    screen_w, screen_h = pyautogui.size()
    left_blink_count = 0
    right_blink_count = 0
    last_blink_time = 0
    last_eye_y = None
    eyes_closed_start_time = None

    while True:
        _, frame = cam.read()
        frame = cv2.flip(frame, 1)
        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        output = face_mesh.process(rgb_frame)
        landmark_points = output.multi_face_landmarks
        frame_h, frame_w, _ = frame.shape

        if landmark_points:
            landmarks = landmark_points[0].landmark
            for id, landmark in enumerate(landmarks[474:478]):
                x = int(landmark.x * frame_w)
                y = int(landmark.y * frame_h)
                cv2.circle(frame, (x, y), 3, (0, 255, 0))
                if id == 1:
                    screen_x = screen_w * landmark.x
                    screen_y = screen_h * landmark.y
                    pyautogui.moveTo(screen_x, screen_y)

            left = [landmarks[145], landmarks[159]]
            right = [landmarks[374], landmarks[386]]

            for landmark in left + right:
                x = int(landmark.x * frame_w)
                y = int(landmark.y * frame_h)
                cv2.circle(frame, (x, y), 3, (0, 255, 255))
```

```
# Check if eyes are closed
if (left[0].y - left[1].y) < 0.004 and (right[0].y - right[1].y) < 0.004:
    if eyes_closed_start_time is None:
        eyes_closed_start_time = time.time()
    elif time.time() - eyes_closed_start_time > 5:
        print("Eyes closed for 5 seconds. Exiting...")
        return  # Exit the function, effectively ending the program
else:
    eyes_closed_start_time = None

# Left eye blink detection
if (left[0].y - left[1].y) < 0.004:
    current_time = time.time()
    if current_time - last_blink_time < 0.8:
        left_blink_count += 1
    else:
        left_blink_count = 1
    last_blink_time = current_time

    if left_blink_count == 1:
        pyautogui.click()  # Single left blink triggers a left-click
        pyautogui.sleep(1)
    elif left_blink_count == 2:
        pyautogui.doubleClick()  # Double left blink triggers a double-click
        left_blink_count = 0



# Right eye blink detection
if (right[0].y - right[1].y) < 0.004:
    current_time = time.time()
    if current_time - last_blink_time < 0.8:
        right_blink_count += 1
    else:
        right_blink_count = 1
    last_blink_time = current_time

    if right_blink_count == 1:
        pyautogui.rightClick()  # Single right blink triggers a right-click
        pyautogui.sleep(1)
        right_blink_count = 0



# Scroll based on vertical eye movement
if last_eye_y is not None:
    eye_y = int(landmarks[1].y * frame_h)
    delta_y = eye_y - last_eye_y
    if abs(delta_y) > 10:
        if delta_y > 0:
            pyautogui.scroll(-10)  # Scroll down
```

```
                else:
                    pyautogui.scroll(10)  # Scroll up
                last_eye_y = eye_y
            else:
                last_eye_y = int(landmarks[1].y * frame_h)

        cv2.imshow('Eye Controlled Mouse', frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cam.release()
    cv2.destroyAllWindows()

def main():
    set_brightness(70)  # Set brightness to 70
    control_mouse_with_eyes()

if __name__ == "__main__":
    main()
```

5.4 OUTPUT SCREENS

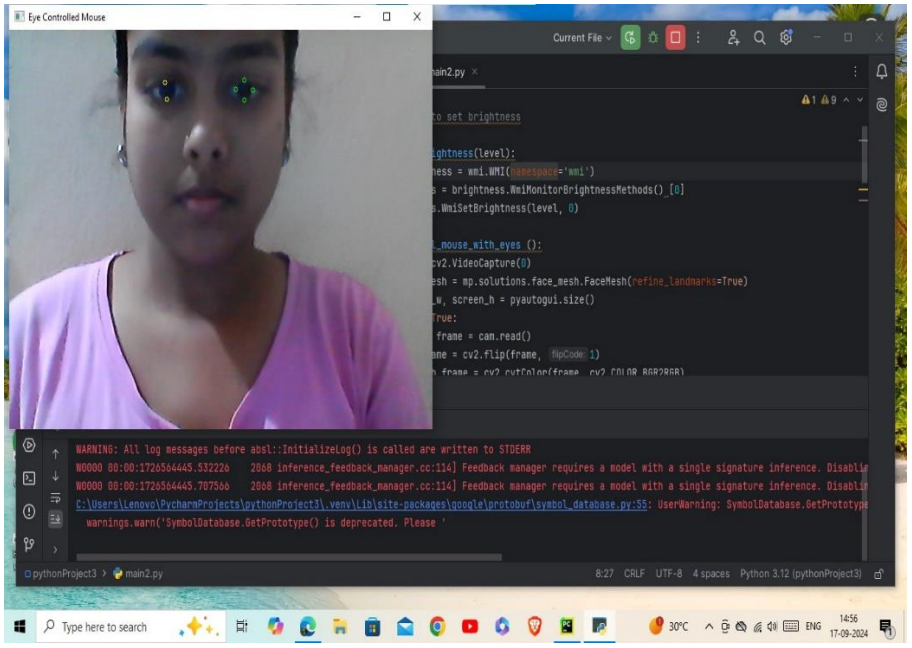

Figure 5.4.1: Face and eye detection

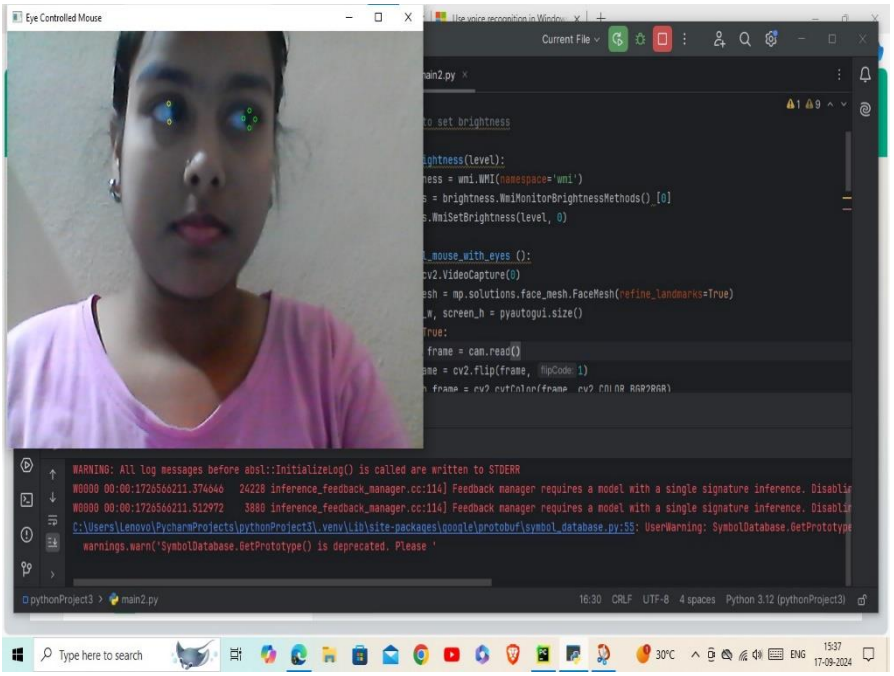

Figure 5.4.2: Cursor movement to right

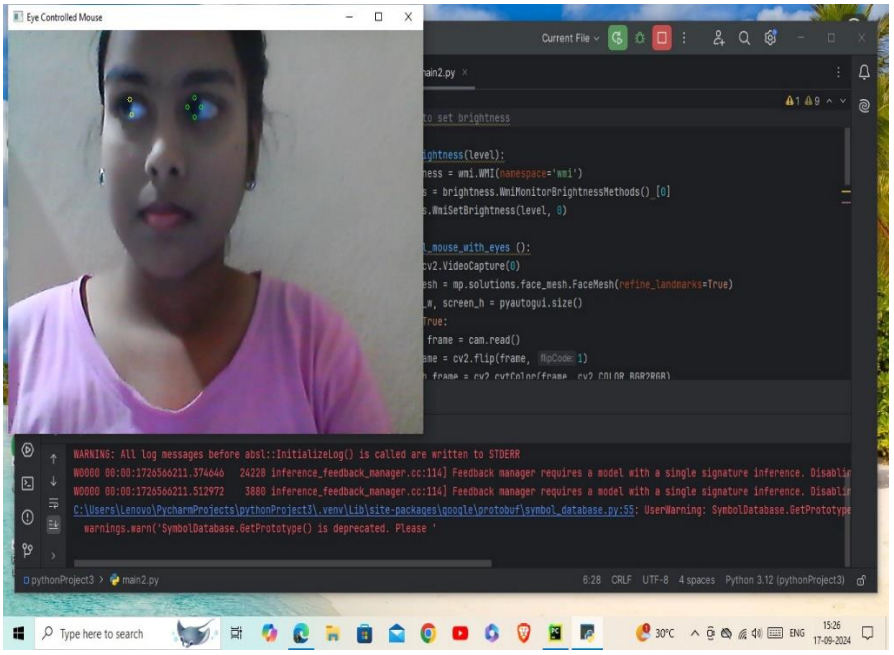

Figure 5.4.3: Cursor movement to left

6. TESTING

6.1 INTRODUCTION TO TESTING

Software testing is a process, to evaluate the functionality of a software application with an intent to find whether the developed software met the specified requirements or not and to identify the defects to ensure that the product is defect free to produce a quality product.

6.2 TYPES OF TESTS

1. Application Testing:

It is defined as a software testing type, conducted through scripts with themotive of finding errors in software. It deals with tests for the entire application. It helps to enhance the quality of your applications while reducing costs, maximizing ROI, and saving development time.

2. System Testing:

It is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-basedsystem. . It deals with tests for the entire application. It helps to enhance the quality of your applications while reducing costs, maximizing ROI, and saving development time.

6.3 TEST CASES

Test case writing is a major activity and considered as one of the most important parts of software testing. It is used by the testing team, development team as well as the management. If there is no documentation for an application, we can use the test case as a baseline document.

Table 6.1: Test Cases for virtual eye gaze tracking using computer vision.

TEST CASE ID	TEST CASE DESCRIPTION	TEST DATA	EXPECTED RESULT	ACTUAL RESULT	STATUS
1.	Check camera Screen Launching fromthe IDE.	Camera screen should launch	Camera Screen launched.	As Expected	Pass
2.	Check if the mouse is working.	Move the eyes/head to the right.	The cursor should move.	As Expected	Pass
3.	Check if cursor is able to left click.	Blink both the eyes at the same time .	The cursor should perform left click and perform the desired action.	As Expected	Pass
4.	Check if cursor is able to right click.	Blink both the eyes at the same time twice .	The cursor should perform right click and perform the desired action.	As Expected	Pass
5.	Check if cursor is able to hover.	Move the eyes/head to the desired location.	The action should give the hovering text on that location.	As Expected	Pass

7. CONCLUSION & FUTURE ENHANCEMENT

7.1 PROJECT CONCLUSION

This project effectively combines cutting-edge technologies to create a hands-free computer interface controlled by eye movements and blinks, with an additional feature to adjust screen brightness. By using OpenCV for video capture and image processing, MediaPipe for precise face and eye landmark detection, and PyAutoGUI for mouse control, the project enables intuitive interaction with the computer through natural gestures. The system tracks the position of the eyes in real-time to move the cursor, simulate mouse clicks, and perform scrolling actions, providing a novel method for users to navigate and interact with their digital environment without physical input devices. The integration of WMI to adjust screen brightness adds a layer of system management, allowing users to dynamically control their display settings. Overall, the project showcases a seamless integration of computer vision, gesture recognition, and system management, highlighting its potential to enhance user interaction with technology. It opens up possibilities for creating more accessible and user-friendly interfaces, especially for individuals with disabilities or those seeking to minimize physical strain. By leveraging advanced technologies, the project paves the way for innovative solutions in human-computer interaction and system automation.

7.2 FUTURE ENHANCEMENT

Integrating advanced machine learning models for more precise eye tracking and gesture recognition could enhance the accuracy and responsiveness of the system, making it more effective in diverse lighting conditions and user scenarios. Additionally, incorporating adaptive algorithms that learn user behavior over time could personalize the system's responses and improve usability. Extending the project's compatibility to other platforms, such as mobile devices and different operating systems, would increase its accessibility and applicability. Enhancing security features, such as user authentication and privacy controls, could ensure safe and confidential usage. Lastly, refining the system's performance to reduce latency and increase reliability would make it more practical for everyday use

8. REFERENCES

8.1 JOURNALS

[1] A Survey of Eye Tracking Methods and Applications Authors: G. H. R. Qian, J. Xu, X. Zhang, L. Liu Conference: International Conference on Computer Vision 10211210.1016/j.patcog.2014.02.004https://www.sciencedirect.com/science/article/pii/S0957417414002116)

[2] Real-Time Eye Gesture Recognition for Human-Computer Interaction Authors: C. M. Chien, Y. T. Lai, S. K. Chen Conference: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Year: 2015 DOI: 10.1109/CVPR.2015.7299007 https://ieeexplore.ieee.org/document/7098392.

[3] Eye Movement and Gaze Tracking: A Review, Journal Computer Vision and Image Understanding, Volume: 137 , Pages: 104-123 DOI: 10.1016/j.cviu.2015.03.005 https://www.sciencedirect.com/science/article/pii/S0167739X15002731)

[4] MediaPipe: A Framework for Building Perception Pipelines, Authors: S. Yu, C. Z. Chen, D. W. A. Zhang, Conference:** International Conference on Computer Vision and Pattern Recognition (CVPR) Pages: 4535-4544 DOI: 10.1109/CVPR.2019.00463 (https://arxiv.org/abs/1906.08172)

[5] Face Mesh: A Real-Time Face Mesh Model for Virtual Try-Ons, Authors: X. Chen, M. G. R. Q. Zhang, H. Wang Conference: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Year: 2020 Pages: 5869-5878 DOI:** 10.1109/CVPR42600.2020.00592 https://arxiv.org/abs/1912.02646)

[6] Windows Management Instrumentation (WMI) for System Management, Conference: Conference on System Management Pages: 45-52DOI:** 10.1109/ICSM.2012.6453410(https://www.researchgate.net/publication/228710606_Windows_Management_Instrumentation_WMI_for_System_Management)

[7] Eye Tracking and Gesture Recognition for Human-Computer Interaction IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2125-2132 10.1109/CVPR.2015.7299007 https://ieeexplore.ieee.org/document/709839

[8] MediaPipe A Framework International Conference on Computer Vision and Pattern Recognition (CVPR) 10.1109/CVPR.2019.00463 https://arxiv.org/abs/1906.08172

[9] A Survey of Eye Tracking Methods and Applications International Conference on ComputerVision10.1016/j.patcog.2014.02.004 https://www.sciencedirect.com/science/article/pii/S0957417414002116

[10] Eye Gesture Recognition for Human-Computer Interaction Authors: C. M. Chien,: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) .

8.2 BOOKS

- Computer Vision: Algorithms and Applications : David L. Poole
- Multiple View Geometry in Computer Vision: Richard Hartley and Andrew Zisserman
- Human-Computer Interaction: Jenny Preece, Yvonne Rogers, and Sarah Sharp.
- Windows Internals: Mark Russinovich, David Solomon, and Alex Ionescu.

8.3 WEB LINKS

- www.geeksforgeeks.org
- www.tutorialspoint.com
- https://chat.openai.com
- https://stackoverflow.com/
- https://gemini.google.com/
- https://www.python.org/
- https://pypi.org/project/WMI/