

AI VIRTUAL MOUSE

A Capstone Project report submitted
in partial fulfilment of requirement for the award of degree

BACHELOR OF TECHNOLOGY

in

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE

by

SWARAJ KAMMAGANI

2203A51793

PAVAN PUTTA

2203A51685

MOHAMMAD SHAHEDPASHA

2203A51494

Under the guidance of

Mr. Y. NAGENDAR

Assistant Professor, School of CS&AI.



SR University, Ananthasagar, Warangal, Telangana-506371

SR UNIVERSITY

Ananthasagar, Warangal.



CERTIFICATE

This is to certify that this project entitled “AI VIRTUAL MOUSE” is the bonafide work carried out by **SWARAJ KAMMAGANI, PAVAN PUTTA, MOHAMMAD SHAHEDPASHA** as a Major Project for the partial fulfilment to award the degree **BACHELOR OF TECHNOLOGY** in **School of Computer Science and Artificial Intelligence** during the academic year 2025-2026 under our guidance and Supervision.

Mr. Y. NAGENDAR

Asst Professor,

SR University

Ananthasagar, Warangal

Dr. M. SHESHIKALA

Professor & Head,

School of CS&AI,

SR University

Ananthasagar, Warangal.

Reviewer-1

Name:

Designation:

Signature:

Reviewer-2

Name:

Designation:

Signature:

ACKNOWLEDGEMENT

We owe an enormous debt of gratitude to our Major Project guide **Mr. Y. Nagendar, Asst.Prof.** as well as Head of the School of CS&AI, **Dr. M. Sheshikala, Professor** for guiding us from the beginning through the end of the Capstone Project with their intellectual advices and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us to the right direction.

We express our thanks to project co-ordinators **Mr. Sallauddin Md, Asst. Prof., and Dr. D. Ramesh Asst.Prof.** for their encouragement and support.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

CONTENTS

S.No.	Content	Page Number
1.	Abstract	1
2.	Introduction	2
2.1	Introduction	2
2.2	Purpose	3
3.	Problem Statement	5
4.	System Analysis	6
4.1	Introduction	6
4.2	Analysis Model	6
4.3	Existing System	10
4.4	Proposed System	11
5.	Feasibility Study	13
5.1	Technical Feasibility	13
5.2	Economic Feasibility	14
5.3	Operational Feasibility	15
6.	Software Requirement Specification	16
6.1	Functional Requirements	16
6.2	Non-Functional Requirements	19
6.3	Hardware Requirements	21
6.4	Software Requirements	22
7.	System Design	23
7.1	Introduction	23
7.2	UML Diagrams	23
8.	Methodology	29
8.1	Algorithms and Tools Used	29
8.2	Capturing video frames using camera	31
8.3	Performance analysis and measures	38
9.	Testing	41
10.	Result	44
11.	Future Enhancements	49
12.	Conclusion	50
13.	Bibliography	51

1. ABSTRACT

The fast pace of Human-Computer Interaction (HCI) has also increased the demand for intuitive, contactless, and highly adaptable input methods. Although computer mouse interactions are typically very reliable, these interactions are based on physical touch, which can be a limitation from hygiene and accessibility perspectives (especially for hospitals, labs, and shared workspaces). In an effort to address these challenges, this project presents an AI-based Virtual Mouse System that allows the user to interact with their computer via a standard webcam using natural hand movements. The system utilizes MediaPipe hand-tracking, OpenCV image-processing, and gesture-mapping algorithms to monitor real-time hand movement and trigger mouse actions such as cursor control, left and right clicks, drag-and-drop, scrolling, and gesture-based interactions.

What is unique about the system is the integration of a graphical user interface that has an authentication procedure for gait recognition. This is used to ensure particular user sessions, manage profiles, and secure access to the application. The application maintains detailed statistics on user gestures, including click counts, scrolling, and user patterns, so that performance and user analysis can be provided based on specific users. The modularity of the architecture supports reliable gesture detection, smooth cursor movement, and low latency based on inferred actions through hand and finger gestures, while providing a user engagement experience that brings aspects of traditional input devices to the user experience.

The intended system has the benefits of being lightweight, inexpensive, and low-barrier to use, as it does not require any hardware other than a webcam. Furthermore, it presents a real-life application that supports touchless interaction in environments where hygiene, accessibility, or hands-free operation is desirable. Ultimately, the AI Virtual Mouse provides a concrete example of how computer vision and gesture-based interaction can enhance human-computer interaction and provide a compact and convenient solution for modern computing needs.

2. INTRODUCTION

2.1 Introduction

The Human-computer interaction (HCI) has also greatly impacted on the interaction by its users with digital systems especially where accuracy, speed, and convenience are expected in the application. A long tradition of primary tools of interaction with computers is the traditional input devices like the mouse or the keyboard. Nonetheless, these gadgets need to be constantly handled, which may restrict access and hygiene in those conditions, in which cleanliness and contact-free interactivity are critical. Hospitals, laboratories, and other public facilities, as well as high-use workstations, are encompassed by settings that are increasingly demanding concerning more modern interfaces which will limit physical contact whilst remaining efficient and user-friendly.

These issues are solved using the AI Virtual Mouse system which allows controlling the work of a computer using hand movements measured with a camera. The system is tracking hand movements in real time with the assistance of computer vision and machine learning algorithms instead of based on input devices involving hardware application. The system finds the position of the fingers, tracks the movements, and relates them to mouse behaviors with the help of OpenCV to process video frames and the help of MediaPipe to locate the hands and identify their landmarks properly. These are the movement of the cursor, left and right clicks, drags, and other crucial interactions that make the navigation easy.

The AI Virtual Mouse can be used more cleanly, conveniently, and easily because it does not require physical contact. It has a user-friendly relationship feature with the people who might have poor or strict mobility with a physical mouse because of a disability, injury or other disability. Touchless interface can also prove helpful in the workplace where sterility and contact with surfaces are of utmost importance. Furthermore, the system embodies the increased movement of intelligent, contactless technologies to support the modern computing requirements.

In general, AI Virtual Mouse is a new step in the development of digital communication and it shows that computer vision and gesture recognition can make the interaction between users truer and more natural. With contactless technology being highly applicable, such a system leads to the evolution of adaptive, flexible, and convenient ways of interaction, which meet the need of the current and future technologies.

2.2 Purpose

The Virtual Mouse, an AI-based system we are proposing, seeks to address the shortcomings of traditional input devices with a straightforward, contactless, and intuitive user experience on computers. By implementing real-time hand-tracking and gesture-recognition methods, the system allows a user to conduct essential mouse features using hand gestures, and all through a standard webcam. This means eliminating the need entirely for a physical device, improving hygiene, and creating a flexible user experience appropriate for the modern digital context.

Using computer vision and machine learning methods, the system accurately interprets finger locations and gestures, allowing users to move around the interface, conduct clicking actions, and execute commands in a timely manner. Overall, our main goal here is to provide a user-friendly, easy-to-use interaction interface that improves comfort, reduces physical constraints, and allows for hands-free use.

This gesture-based controlled mechanism intends to improve productivity, creates an area for innovation in Human-Computer Interaction, and provides an accessible solution for work areas in which it is not possible to use common input devices, as is common in areas such as hospitals, laboratories, public kiosks, and shared work spaces with shared-technology. In short, the Virtual Mouse strives to construct a new, adjustable, and exploratory user experience that is designed specifically for emerging contactless technology.

Input Modalities

The Virtual Mouse system uses primarily hand movement that is captured live, via video captured by a webcam, as its main input modality. Hand movement is processed using computer vision to handle the movement of the cursor and mouse clicking actions, allowing for a seamless and contactless interaction with a computer device without any other commonplace input device.

Data Acquisition

The Virtual Mouse System utilizes a webcam to acquire real-time gesture data from the user's hand. The webcam continually captures video of the user's hand and the frames are processed to enhance the hand landmarks, as well as recognizably figure out fingertip information for gesture triggered control actions.

Data Processing

The Virtual Mouse system uses computer vision and machine learning to process video frames from the webcam. Hand landmarks are identified, read, and translated into cursor movement and mouse click interactions, enabling the system to respond and accurately detect and identify user gestures in near real-time.

User Intent Recognition

The Virtual Mouse system discerns the user's intended computer action by focusing on the subject, hand gesture-based movements, into action. Specific fingers can be raised and move into regions, as essentially mapped to mouse actions including cursor movements, clicking and dragging, which allows the system to identify user commands in near real-time.

Cursor Control

The Virtual Mouse system handles the recognition of hand gesture-based movement to on-screen cursor movements. The system provides consistent but precise control by continuously monitoring hand position and simultaneously determines the on-screen cursor movement, and updates the on-screen representation.

3. PROBLEM STATEMENT

Traditional computer mouse interfaces give a hard time to individuals with motor disabilities, or people having repetitive strain injuries, or people working in environments where making physical contact with an input device is impractical or unsanitary. Traditional mouse rely on exact or precision-handed movements and physical contact, which presents hinderances to users with limited dexterity, tremors, or other mobility impairments. Moreover, within modern collaborative workspaces or shared computing environments, there is a rising need for touchless modes of interaction to ensure safe computing when dealing with germ potential, while not sacrificing productivity. These concerns demonstrate that alternative interactive paradigms need to be developed, which promote intuitive and hands-free modes of control.

In addition to accessibility concerns, existing mouse control systems lack user-specific security measures and adaptation capabilities. Within multi-user environments, there is no way to know, or substantiate, that only authorized users can interact with the computer through gesture control interface interactions, which leads to security concerns and privacy issues. Furthermore, the user has difficulty customizing their interaction experience, because traditional mouse provide limited adjustability for users with different hand sizes, movement styles, or other preferences. The lack of intelligent gesture recognition systems, which can recognize multiple target users in the same space, decreases the usability and security of computing solutions developed with gesture-controlled modes of interaction.

4. SYSTEM ANALYSIS

4.1 Introduction

The AI Virtual Mouse is a new system that changes the interaction between the human and the computer in the sense that it enables individuals to have control of a computer by being able to make simple movements of their hands as opposed to a physical mouse. After integrating the use of artificial intelligence and the use of computer vision techniques, the system handles live hand movements when a webcam is used and transforms implicit actions as seen by the camera into cursor movement, clicking, drag-and-drop, and scrolling.

The analysis of the system in detail will assist in analyzing the most valuable aspects of gesture detection accuracy, real-time performance, hardware requirements, system reliability, and the overall usability of the system. The analysis of the virtual mouse will help that the mouse works efficiently and offers a smooth, intuitive, and accessible interaction experience through the virtual mouse to all users.

4.2 Analysis Model

The AI virtual Mouse machine converts the movements of the hands at the webcam in real-time to the mouse movements. It relies upon OpenCV and MediaPipe to cross the hand landmarks and process photographs to extract the skill of contactless information dealings with computer systems with ease. The equipment should be human friendly, responsive and flexible as well as the test will comprise every practical, architectural and performance details:

❖ Functional Requirements Analysis

Defines the core actions needed for user interaction, using computer vision to detect gestures and translate them into cursor control:

Cursor Movement: Tracks the tip of the index finger to control the on-screen cursor position with smooth motion.

Left Click: Pinch gesture (thumb + index finger) performs a left click.

Right Click: Recognizes a two-finger gesture for right-click operations.

Double Click: Rapid pinch gesture triggers a double-click.

Drag and Drop: Pinch-and-move gesture allows dragging objects.

Scrolling: Index + middle finger vertical or horizontal swipe performs scrolling.

Zoom In/Out : Finger spreading or pinching gesture for zooming.

Custom Gestures: Allows user-defined gestures for specialized commands.

Click Counters: Tracks the number of left clicks, right clicks, double clicks, and drag-drop actions.

Visual Feedback: Displays detected gestures or landmarks on the screen to guide the user.

Settings Adjustments: Cursor speed, gesture sensitivity, and visual feedback can be customized.

❖ **System Architecture Using Computer Vision**

The architecture involves several modules that work together to capture, process, and interpret gestures as cursor actions:

Input Hardware:

A webcam captures continuous video frames of hand movements.

Software Modules:

Image Preprocessing Module: Preprocesses video frames through color filtering, background subtraction, and filtering noise to isolate the hand.

Hand Detection and Tracking: Locates and tracks the hand through computer vision functions, including contour detection, skin segmentation, and/or deep-learning detection.

Gesture Recognition Module Classifies gestures into observable actions by studying the hand region through basic computer vision methods or a trained CNN or machine learning model.

Cursor Control Module Responds to defined gestures as controlled cursor movements, clicks, drag-drop, scroll, and zoom, as well as user-customized gestures to act on commands on the user's screen.

Feedback Module: Provides visual feedback to the user to display the status of gesture recognition to build user confidence and accuracy when interacting with gesture input.

❖ **Data Flow and Processing Model**

Input Stage

Webcam is used to take live video of the user's hand and environment.

Preprocessing

Each frame is filtered and processes to identity the hand.

Gesture Detection

Each processed frame is evaluated using a trained model to identify the gesture.

Action Mapping

The recognized gesture is mapped to an operating system action (movement, or click, drag-drop, scroll, or zoom).

Output

The command is executed to the operating system as standard mouse actions.

❖ **Gesture Recognition Using Computer Vision Techniques**

Computer vision in the AI virtual mouse relies on specific algorithms and models to detect and classify gestures accurately:

Hand Segmentation

Methods such as color-based segmentation and background subtraction help separate the hand from the surrounding background.

Feature Extraction

Key features like fingertips or palm position are extracted to determine gestures and track hand movement.

Convolutional Neural Networks (CNN)

Used to classify hand shapes and movements in real-time, CNNs are trained on labeled datasets of hand gestures, making it possible to detect common gestures accurately.

Object Detection Models

Advanced models like YOLO or MobileNet can be used for robust hand detection, allowing the system to identify hand location and shape even under varying lighting or backgrounds.

❖ Performance and Usability Metrics

Evaluates the system's effectiveness, with a focus on real-time performance and ease of use:

Latency

Ensures the virtual mouse responds instantly to hand movements, creating a seamless user experience.

Accuracy

High recognition accuracy minimizes incorrect cursor movements or clicks.

Resource Efficiency

Optimizes CPU and memory usage for smooth operation, even on standard hardware.

User Adaptability

Allows customization of gestures, sensitivity, and interface preferences, adapting to individual user needs.

❖ Security and Privacy Analysis

Since the system captures live video, security is critical:

Data Protection

Ensures that video data is processed in real-time without being stored or transmitted, protecting user privacy.

Permissions and Access Control

Limits access to the camera, safeguarding the video feed from unauthorized use.

4.3 Existing System

Various systems that have been developed have investigated other forms of human-computer interaction, mostly centered around the technology of gesture recognition, touchless, and eye-tracking. The main traditional systems that are being commonly utilized are the use of the standard mouse and keyboard, which are very reliable but has shortcomings in the mobility, hygiene, and accessibility when used in such sensitive areas as a hospital, laboratory, and public workstations.

Gesture-Based Systems:

The prior research and application have implemented these webcams or depth sensors to read the hand gestures and control the movement of the cursors. Most systems are based on a basic finger movement or template gesture recognition in order to execute functionality like cursor control, clicks and scrolling. Although they work well when used in a controlled setting, these systems are usually challenged by issues such as:

- Limited gesture vocabulary
- Poor oral re-created abilities in diverse lighting circumstances.
- Late Latency or Jitter Cursor Movement.
- Absence of real-time natural communication.

Eye-Tracking Systems:

Alternative input modality has also been provided through use of eye-tracking whereby users can control cursors using their eyes. Despite the non-contact nature of interaction, these systems are costly, hardware-based and usually experiences fatigue when used over a long period of time.

Brain- Computer Interface (BCI) Systems:

Neural control is offered by BCI-based methods of input through hands-free control. They are largely experimental systems, demand specialized equipment and are not currently feasible in the routine computing workload because of the complexity of the systems and the effort of setting them up.

Limitations Of Existing System

- The majority of them need specialized or costly hardware.
- Gesture recognition is both inaccurate and restricted.
- It is not always possible to provide real-time responsiveness and easy control of the cursor.
- Feedback to user interface is not always provided so gesture recognition can hardly be verified.

4.4 Proposed System

The system presented here demonstrates a way to utilize a gesture controlled virtual mouse to interact with a computer by merely placing your hand near a web camera. A computer vision based camera recognizes, and tracks, the user's hand, identifying spatially relevant positions of the fingers, and interpreting gestures in real time utilizing the recorded finger positions. Recognized gestures will be translated into mouse-enabled actions such as scrolling, dragging objects on a screen, and moving a cursor pointer.

The system looks to facilitate more natural touch-less interaction methods, and more accessible user input. The virtual mouse design is based on continuously tracking the user's hand, and then analyzing postural finger movements to translate into smooth and accurate interaction methods any physical mouse would provide. This system has virtually little to no hardware dependencies, and provides many benefits to user convenience and hygiene in touchable user input methods within shared or sensitive environments. Since the virtual mouse only relies on simple to perform gestures and simple to use interface, this can be friendly people of all ages and require little or training to interact.

The proposed system therefore focuses on real-time processing, accurate gesture recognition and detection, and the ability to provide the comfort for user interaction. Finally, it is modular in design so the scope of the system could be extended to include additional gestures, or voice/AI features.

Advantages

- Delivers a fully touch-free interaction with your computer.
- Provides a natural and intuitive way to control the computer with only simple hand gestures.
- Uses only a low-cost webcam, making the system very affordable.
- Increases accessibility for users who are unable to use traditional mouse devices.

- No hardware dependencies and no motion-related issues or wear and tear.
- Cursor movement is smooth, responsive, and in real time.
- Can be used on any computer, so users will find it very portable.
- Improves hygiene in shared or sensitive environments.
- Easy to add new gestures and features as functionality expands.

5. FEASIBILITY STUDY

A feasibility study evaluates whether a proposed system can be successfully developed and deployed given the required time, resources and technology. In the case of the AI Virtual Mouse, it will assess the feasibility of developing a gesture-based interaction system that is accurate, responsive, economical, and user-friendly.

The feasibility is assessed in three key areas:

1. Technical Feasibility
2. Economic Feasibility
3. Operational Feasibility

5.1 Technical Feasibility

Technical feasibility evaluates if the technological tools that exist can achieve the function of the proposed system.

Hardware Requirements

The system relies on basic hardware – a normal webcam and personal computer. There are no additional sensors or equipment necessarily.

Software Requirements

The implementation relies on Python, using libraries such as OpenCV to complete computer vision operations, and MediaPipe to identify hand-landmarks. These tools are open-source, well-supported, and run on the most common operating systems.

System Performance

The developed algorithms allow the system to function in real time with minimal latency for smooth pointer movement and accurate gesture based functionality including left click, right click, scrolling, dragging, and general point functionality.

Modularity

The project is built using a modular design. Each system organization (gesture detection, pointer control, UI,) is modular in structure, reducing risk and making upgrading with new gestures or new functionality manageable to integrate.

Compatibility:

The system is compatible with the leading operating systems of Windows, Linux, and macOS providing greater accessibility.

Conclusion:

The project is technically feasible because of the availability and access to required tools, minimal hardware resources, and the ability to perform gesture recognition in real-time.

5.2 Economic Feasibility

The economic feasibility of developing and operating the system involves understanding the various costs for development and evaluating the expected benefits.

Hardware Cost:

All necessary hardware costs are minimal, since a standard web camera is used from an existing computer.

Software Cost:

There was no monetary cost to the software stack as all the libraries used were open source, thus there were no costs associated with licensing the software.

Development Cost:

The project development costs remained very low because the various tools and libraries were free of charge, and because no new hardware or costly equipment was used.

Maintenance Cost:

The maintenance of the application consisted of periodic software updates or refinements. The costs associated with maintenance of the application were minimal, particularly in comparison to hardware-based systems.

Cost-Benefit Ratio

Compared with commercial gesture-based or touchless input devices, the AI Virtual Mouse has provided at least the same benefits as the commercially available systems that were reviewed at a fraction of the price.

Conclusion:

This project is economically feasible, providing a low-cost alternative to previously reviewed systems of gesture-interaction.

5.3 Operational Feasibility

Operational feasibility assesses if the system is capable of effectively operating for its intended users, in a realistic setting.

User Friendly

The interface is designed to be intuitive and includes on screen visuals and indicators to walk users through the different gestures.

Accessibility

The hands-free feature of the control system serves as an advantage when touch options are either impractical (circumstances when the user cannot touch a screen) or unhygienic (for example hospitals, laboratories, or in environments focused on accessibility).

Customizability

The user has the ability to adjust parameters, such as gesture sensitivity, mouse cursor response, and visual feedback on display to user's preferences.

Reliability

The system continuously and accurately translates gestures into mouse commands, allowing a seamless transition into standard mouse operations.

Integration

It can be integrated into users existing computer system, without changing normal flow of operation and does not require additional software.

Conclusion:

The AI Virtual Mouse is operationally feasible, which means it provides an effective, reliable, accessible, and user-friendly method for manipulating a computer using hand gestures.

6. SOFTWARE REQUIREMENT SPECIFICATION

The Software Requirements Specification (SRS) gives a clear and detailed description of the AI-Based Virtual Mouse System. This chapter aims to provide a thorough explanation of all system requirements, including both functional and non-functional aspects. This document helps all stakeholders—developers, evaluators, faculty, and end users—understand what the system is meant to do, how it works, and the limitations it has.

The AI Virtual Mouse allows users to control a computer without a physical mouse. It uses a webcam to capture real-time hand movements and interprets these gestures through computer vision and machine learning techniques. By identifying hand landmarks and corresponding gestures, the system converts finger movements into mouse actions like cursor movement, clicking, scrolling, and dragging. The project incorporates several modules, including gesture recognition, user interface, coordinate mapping, and an optional authentication module.

This chapter outlines the expected behavior of the system and ensures consistency between the design and the actual code.

6.1 Functional Requirements

Functional requirements detail what the system must do. They relate directly to the actions coded using OpenCV, MediaPipe, PyAutoGUI, your UI, and authentication.

The AI Virtual Mouse system carries out several key functions:

1. Real-Time Image and Video Capture

❖ Webcam Initialization

The system must automatically find and activate the default webcam connected to the computer. If the camera is not available, the system must show an appropriate error message.

❖ Continuous Frame Acquisition

The system must continuously capture video frames as long as the application is running. The captured frames must be sent immediately to the gesture-processing module.

❖ **Frame Preprocessing**

The system must resize, flip, or change color spaces (e.g., BGR to RGB) to fit the MediaPipe detection model. This preprocessing must ensure uniformity for gesture recognition.

2. Hand and Finger Landmark Detection

❖ **Hand Detection**

The system must detect whether a user's hand is present or absent within the camera's frame. If a hand exits the frame, the cursor must either stay still or keep its last known state.

❖ **Extraction of 21 Hand Landmarks**

MediaPipe must identify 21 landmark points, including fingertips, joints, and palm locations. These coordinates will be used for gesture recognition and cursor calculations.

❖ **Single-Finger Mode for Cursor Control**

The index fingertip position (landmark 8) must serve as the primary cursor pointer.

3. Gesture Classification and Interpretation

❖ **Cursor Movement Gesture**

When the user holds their index finger straight, the system must recognize this as a cursor-movement gesture. The cursor must follow the fingertip's position in real time.

❖ **Left-Click Gesture**

A left-click action must occur when the distance between the index finger and thumb is below a specific threshold (pinch gesture). The system must prevent repeated clicks by implementing gesture debouncing.

❖ **Right-Click Gesture**

A right-click must happen when a specific hand position is detected (e.g., a middle finger bending pattern).

❖ **Scroll Gesture**

Upward or downward finger movements must trigger scroll actions. The system must differentiate between scroll directions based on vertical movement.

❖ **Drag-and-Drop Gesture**

A continuous pinch gesture must activate dragging mode. Letting go of the pinch gesture must drop the selected item.

4. Cursor Mapping and Motion Control

❖ **Coordinate Normalization**

Raw camera coordinates must be converted into normalized screen coordinates. This ensures consistent cursor movement across different screen resolutions.

❖ **Smoothing and Stabilization**

The system must use filters (e.g., weighted averaging) to minimize jitter. Cursor movement must remain smooth, even with detection noise.

5. User Interface and Settings Control

❖ **Real-Time Feedback**

The UI must show a live video preview with visual markers for detected landmarks. Detected gestures must be displayed on the screen as feedback.

❖ **Sensitivity Control**

The system must let users adjust:

- cursor speed
- gesture detection sensitivity
- scroll intensity

❖ **Start/Stop Tracking**

Users must be able to pause gesture detection and resume without restarting the application.

6. Authentication Module

❖ User Login

The system must require users to log in before accessing restricted features.

❖ Password Security

Passwords must be hashed using a secure algorithm. The system must prevent duplicate usernames.

❖ Session Handling

The system must recognize the current logged-in user and limit unauthorized actions.

6.2 Non-Functional Requirements

Non-functional requirements specify system quality, performance limits, usability standards, and operational guidelines.

1. Performance Requirements

❖ FPS Requirement

The system must maintain at least 15 frames per second for gesture tracking, with potential to exceed 25-30 frames under optimal conditions for smoother performance.

❖ Low-Latency Detection

The response time from hand movement to cursor action must stay under 100-150 milliseconds.

❖ Resource Efficiency

The system must process video frames using minimal CPU resources. Memory usage must not exceed typical application standards.

2. Usability Requirements

❖ Clear User Interface

All controls (start, stop, sensitivity) must be easy to find and use. Users must receive clear visual feedback for every detected gesture.

❖ Accessibility

The system must support users with mobility challenges. The touchless design reduces effort and physical strain.

3. Security Requirements

❖ Authentication Security

Passwords must never be stored in plain text. Only hashed passwords must be saved.

❖ Data Privacy

User preferences and settings stored locally must be encrypted or secured through appropriate file permissions.

❖ Unauthorized Access Prevention

The system must block user entry after multiple failed login attempts.

4. Reliability Requirements

❖ System Stability

The system must run continuously without crashing for long durations.

❖ Error Handling

The system must be able to:

- detect camera failures
- manage frame read errors
- keep running without sudden termination

❖ **Gesture Accuracy Dependability**

The system must maintain 95% tracking accuracy under proper lighting conditions.

5. Maintainability Requirements

❖ **Modular Architecture**

Each system component (UI, gesture logic, authentication) must be in separate modules.

❖ **Code Documentation**

Every major function must have comments explaining its operation.

❖ **Logging Support**

The system must log errors, warnings, and performance data for developers to review.

6. Compatibility Requirements

❖ **Hardware Compatibility**

The system must support:

- standard webcams
- USB webcams
- built-in laptop cameras

❖ **OS Compatibility**

Must work on: - Windows

- Linux distributions

- macOS

❖ **Resolution Adaptability**

Cursor mapping must automatically adjust to different screen resolutions.

6.3 Hardware Requirements

The following hardware components are required to run the system effectively:

1. Webcam

A working webcam is crucial for capturing hand gestures. Higher clarity improves detection accuracy. Real-time streaming is needed to process gestures without delay.

2. Computing Device

A laptop or desktop with the following minimum specs:

- 4GB RAM
- Dual-core processor
- At least 1GHz clock speed
- Compatible GPU (optional for faster processing)

6.4 Software Requirements

The system relies on several Python libraries and frameworks:

❖ Python Environment

A Python 3.x environment must be installed.

❖ Libraries Used

- **OpenCV:** For image capture, preprocessing, and handling video streams.
- **NumPy:** For mathematical operations on arrays and coordinates.
- **MediaPipe:** For hand landmark detection and tracking.
- **PyAutoGUI / Pynput:** For controlling mouse movement and clicks.
- **Autopy:** An alternative for mouse control.
- **Tkinter / Custom UI modules:** For creating interactive interfaces.

❖ Operating System Support

- Windows 10/11

7. SYSTEM DESIGN

7.1 Introduction

Systems design is the second major step in the software development life cycle (SDLC) and commences with transitioning from conceptual requirements (mentioned earlier) into an actual plan- a detailed development plan. This step transforms a logical descriptor of user needs into a working technical plan for development. Development begins when a Software Requirements Specification (SRS) is been created. The SRS indicates what a system must do when it is requested, and the design phase is how those requirements will be fulfilled.

System design acts as a bridge between both problem and solution domain. It translates both functional and non-functional requirements into architectural components, data structures, layouts of the interface, logic of processing, and workflows of operation. Rationale is applied for both high-level and low-level design decisions, ensuring the resulting system will be organized, efficient, maintainable and will meet to the user expectations. Ultimately a complete design specification is produced at the end of this stage, as it acts as the basis for coding, testing, and deployment.

7.2 UML diagrams

Unified Modeling Language (UML) diagrams provide visual representations that address various views of a software system. They contain structural views - for example, class or component diagrams - which show how the parts of a system are organized and connected and behavioral views - for example, use-case or sequence diagrams - which show how a system acts to user actions or even from internal events. By defining symbols and a complete layout, UML diagrams allow us to discussion complex systems using a visual approach regardless of programming language.

Specifically, these diagrams represent a communication bridge between stakeholders, architects, developers, and testers. They help clarify the requirements of the system by catching hidden interactions, allow inconsistencies in the design to be discover in an early phases of development, and provide a representation that allows implementation. Having a balanced set of structural and behavioural UML diagrams allows a system's architecture to be known, as well as known of the system at runtime. Having reliable and maintainable software is possible through communication using UML diagrams.

Use Case Diagram

Use Case Diagrams provide a simplified view of how people or external systems interact with software. Instead of describing internal logic, they showcase what the user expects the system to do. Each use case represents one meaningful task the system performs, while actors represent the individuals or components that initiate those tasks. The purpose of this diagram is to highlight the major services of the system and clarify the responsibilities of both the user and the software. By presenting these interactions visually, the diagram helps confirm that the system's features align with real user needs before development begins.

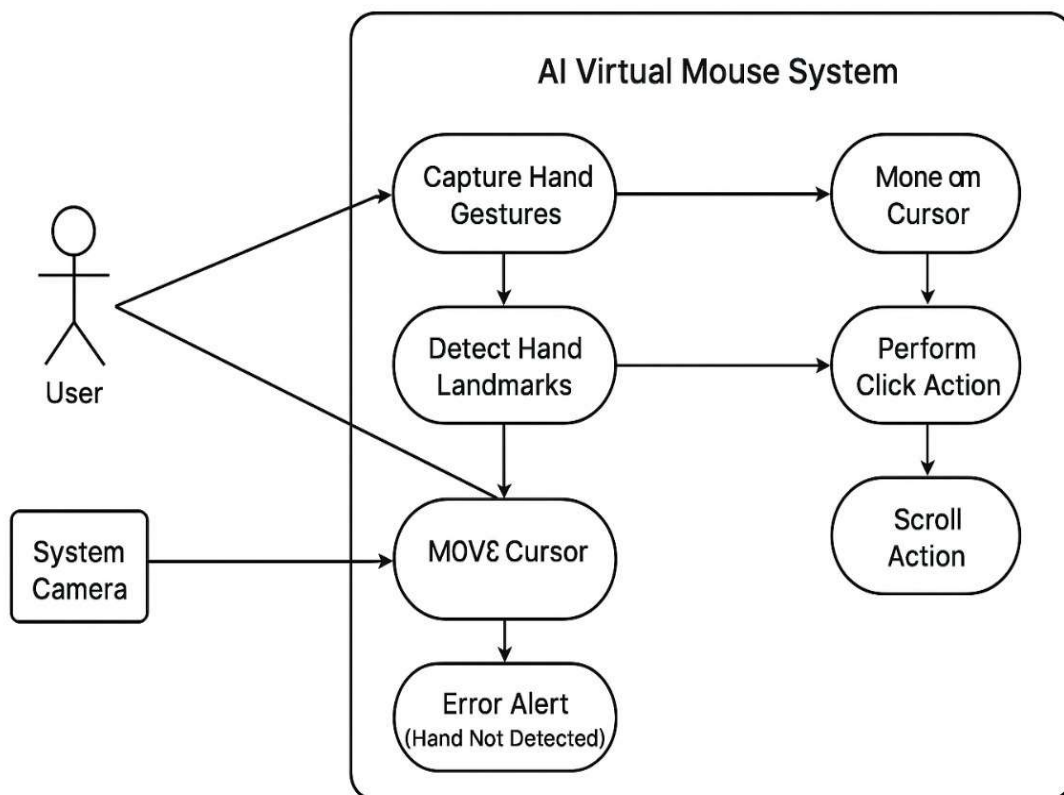


Fig: 7.2.1 Use Case Diagram

Class Diagram

Class diagrams are utilized to show how the system is structured internally. They outline the main classes the software will use, the information each class will store, and the operations possible on each class. Additionally, the diagram shows how these classes relate to one another in terms of dependence on each other, inheritance, or sharing of information. This visual structure acts as a blueprint for developers, who rely on it to guide how the system's data and functionality should be arranged during implementation. Since it captures the core building blocks of the system, the Class Diagram plays a major role in shaping a clear and maintainable design.

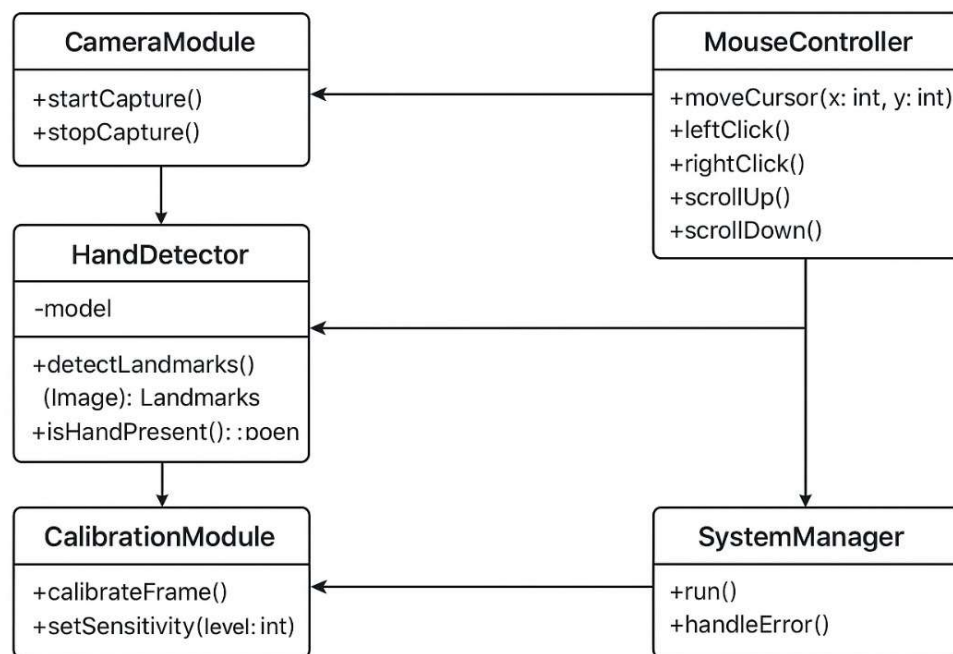


Fig:7.2.2 Class diagram

Activity Diagram

The Activity Diagram for the AI Virtual Mouse outlines the sequence of steps followed during a typical session. The flow begins with user login, after which the virtual mouse module is activated. The system then starts the webcam, captures frames, and processes them to detect hand landmarks using MediaPipe. Recognized gestures—such as cursor movement, clicking, dragging, or scrolling—are converted into mouse actions. Throughout the session, the system updates user statistics and continues processing gestures until the user stops the application or logs out. This diagram provides a simplified view of the real-time interaction cycle within the system.

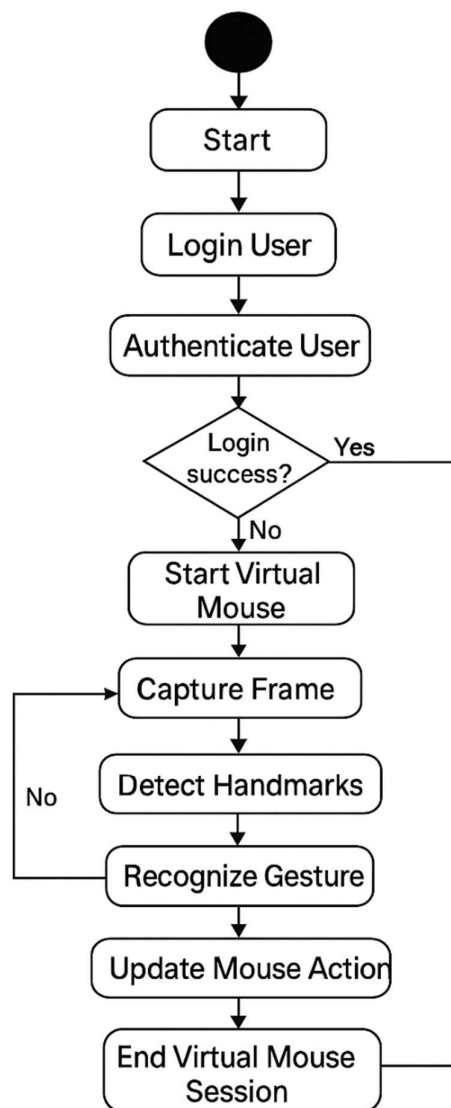


Fig: 7.2.3 Activity Diagram

Block Diagram

The Block Diagram of the AI Virtual Mouse presents the main modules involved in processing gestures. It starts with the user interface and authentication, followed by the camera input that captures hand movements. These frames move to the hand-tracking and gesture-recognition blocks, where gestures are identified. The recognized gesture is then passed to the cursor-control block to perform actions like movement or clicking. A statistics block records all user interactions during the session.

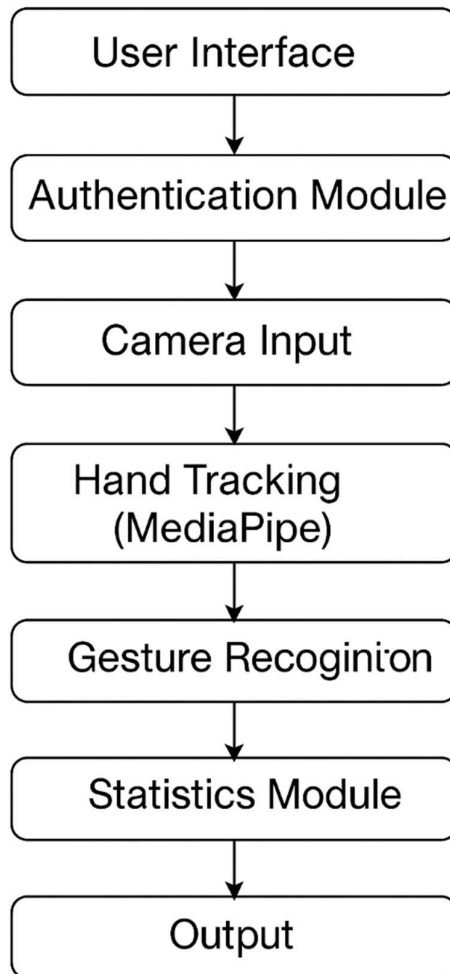


Fig:7.2.4 Block Diagram

Component Diagram

The Component Diagram of the AI Virtual Mouse highlights the major software modules and how they interact. It includes the user interface for login and navigation, the authentication module for validating users, and the camera component that captures live hand movements. The hand-tracking and gesture-recognition components process these frames to identify user gestures, while the cursor-control component converts them into mouse actions. A separate statistics component records clicks, scrolls, and other interactions. Together, these components show how the system is organized and how data flows between its functional parts.

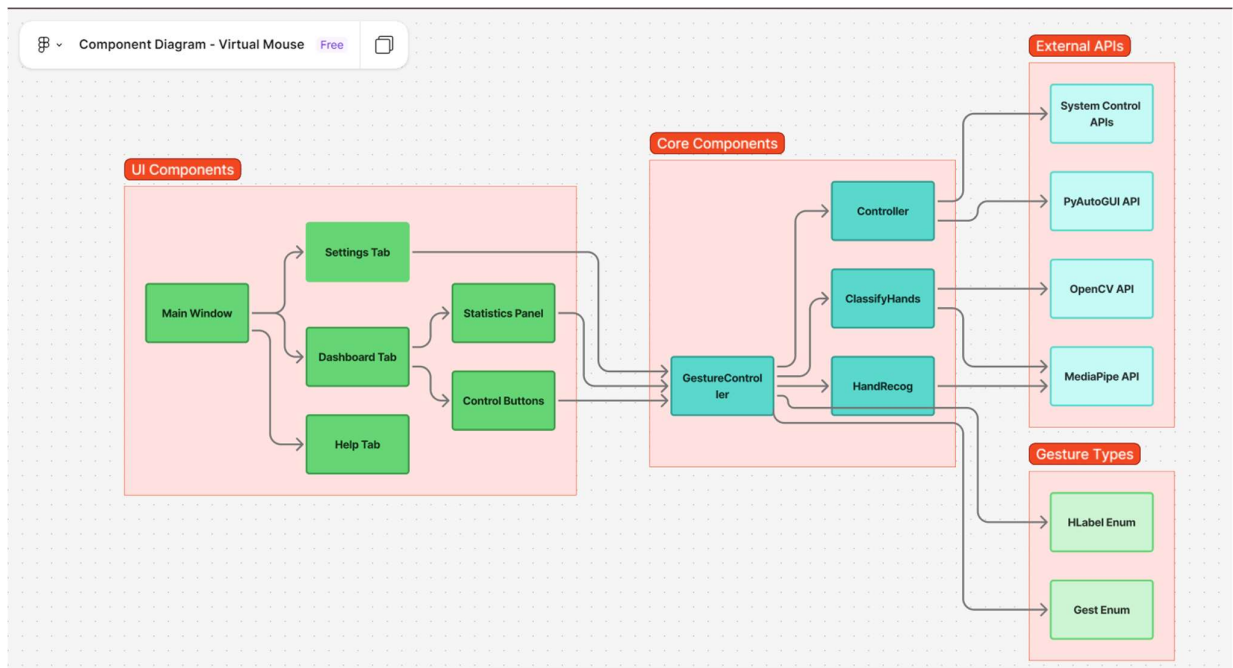


Fig:7.2.5 Component Diagram

8. Methodology

8.1 Algorithms and Tools Used

The AI Virtual Mouse is created based on both the sophisticated computer vision technology and machine-learning-driven devices that could interpret the hand movements in real-time. The system combines the use of MediaPipe in the detection of hand landmarks and OpenCV in processing images, camera operations, and frame manipulation. Collectively, these technologies offer a secure channel of transforming physical hand actions into digital mouse actions.

Mediapipe

In this project, MediaPipe is the ultimate driving force of gesture recognition. It provides a set of trained models and configurable parts that can be used to accomplish visual perception tasks. Specifically, the MediaPipe Hands solution is applied to detect and track the vital landmarks of the human hand. This model carries out two key tasks: a palm detector detects the presence and approximate position of the palm; a specific landmark model calculates the precise locations of 21 points, such as finger tips, joints, as well as the wrist.

The framework works like a directed graph whereby information flows via interlinked nodes characterized as calculators. Every calculator performs a particular task, e.g., region detection, landmark prediction, or coordinate transformation. Such a modular design enables MediaPipe to handle continuous streams of video frames with very little delay making it particularly suitable to real-time interaction systems. Also, the model is optimized to run on CPUs, so there is no need to run high-performance GPUs or specialized hardware.

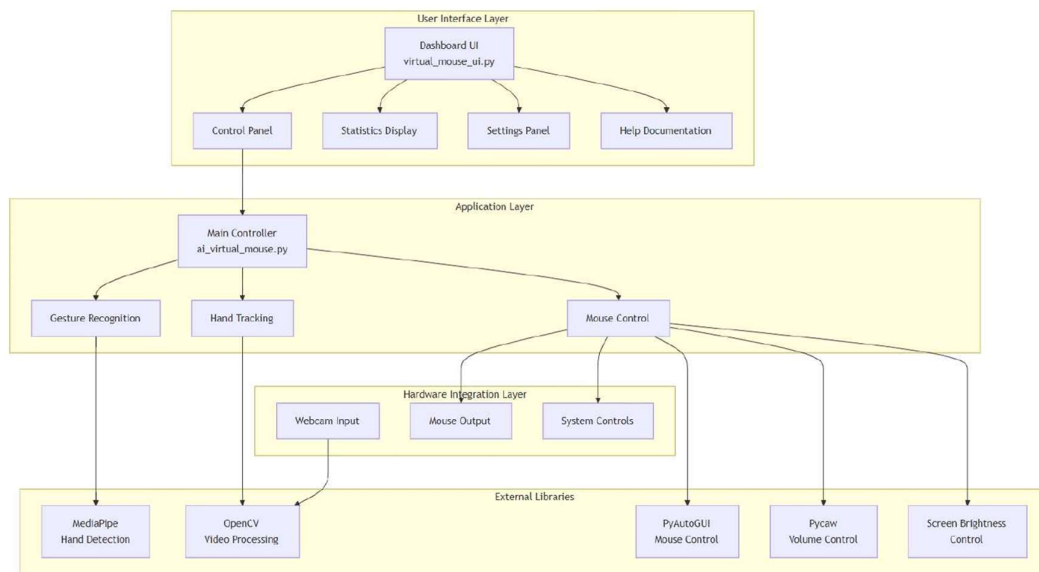


Fig:8.1.1 System Architecture Diagram

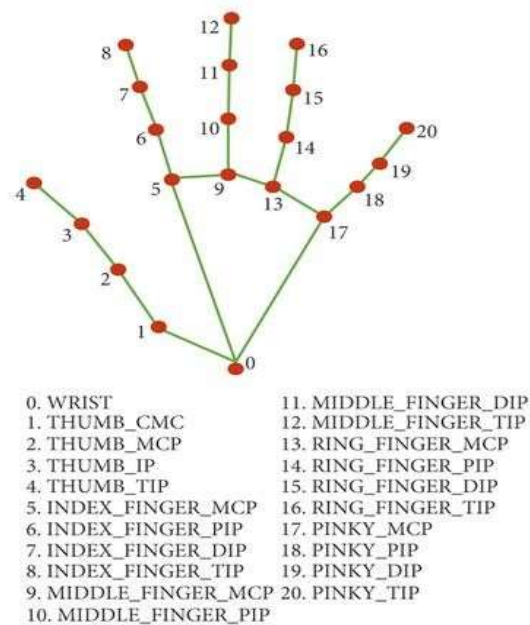


Fig: 8.1.2 Co-ordinates or land marks in the hand

Open-CV

OpenCV also augments the MediaPipe because it takes care of all the camera-related functions as well as the preprocessing functions. It records real-time shots on the webcam, manipulates image formats and other tools to use pixel data. Color conversion, blurring, scaling, and contour drawing are only some of the functions used to prepare the input frames to facilitate proper analysis. The

hand landmarks can also be visualized with the help of OpenCV to enable the user to view gesture feedback on the screen.

Outside the simplest image processing, OpenCV aids in the motion smoothing, noise elimination, and enhancing the stability of the cursor by filtering the variations in landmark positions. This makes sure that the small trembles of the hands or some unintentional movements would not influence the functioning of the system.

Gesture Mapping Algorithm

The system also uses a custom gesture-mapping algorithm, in addition to MediaPipe and OpenCV, to read in the extracted landmark coordinates. It is a gesture recognition algorithm that measures the position of fingers and inter-landmark distances and relative hand position to distinguish between gestures like moving a cursor, clicking, scrolling or dragging. It is an algorithm which makes the difference between one gesture and another, based on threshold values, angle measurements, and the geometrical relations.

Integration of tools

MediaPipe in combination with OpenCV and the gesture-mapping logic creates an experience of a virtual mouse, which is responsive and smooth. MediaPipe is used to give the structural hand data, OpenCV is used to do operations at the image level and the custom algorithm is used to process the gestures to actionable commands. The AI Virtual Mouse is based on this integrated workflow, which enables the virtual mouse to substitute traditional input devices with a touchless and easy-to-use one

8.2 Capturing video frames using camera

The overview of the hand gesture recognition the hand is detected using the background subtraction method and the result of this is transformed to a binary image. The fingers and palm are segmented to facilitate the finger recognition. The fingers are detected and recognized.

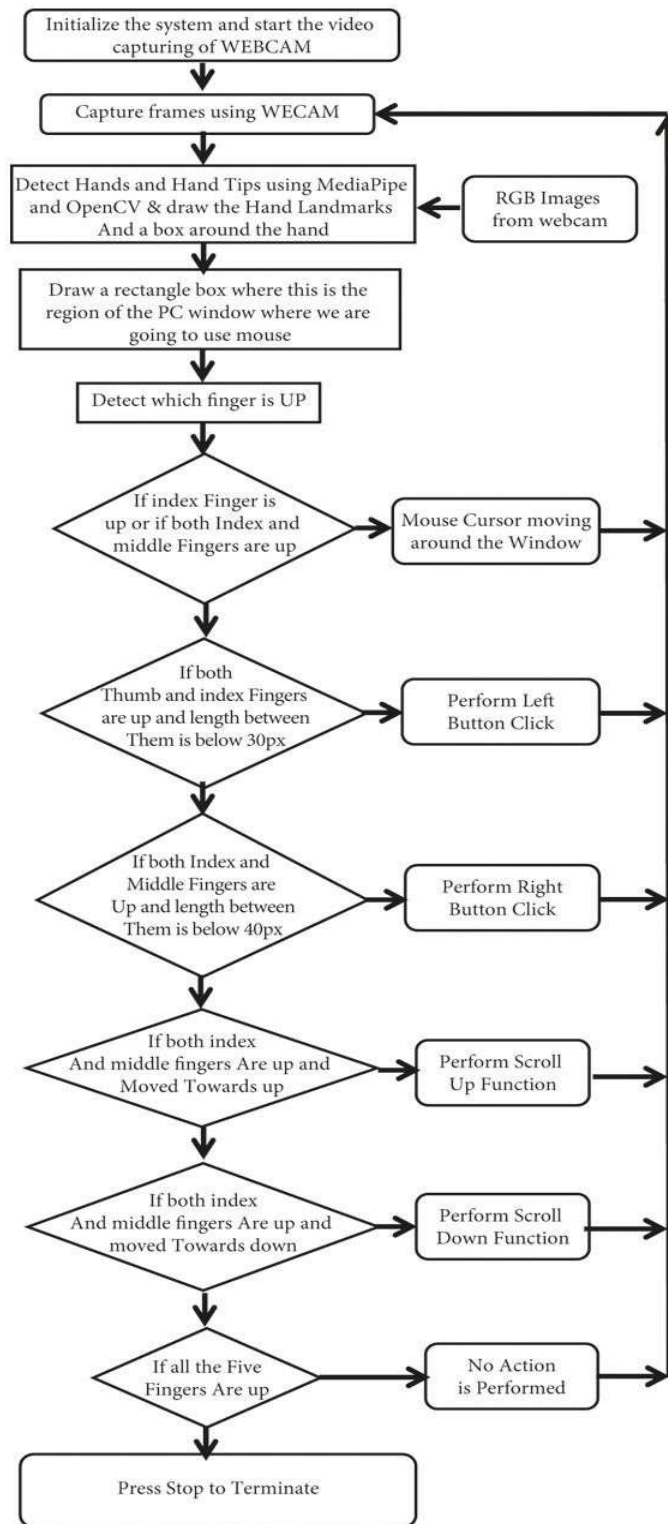


Fig:8.2.1 Flowchart of the real-time AI virtual mouse system.

Camera Input and Frame Acquisition

The AI Virtual Mouse works by taking constant video frames on the inbuilt web camera on a laptop or desktop computer. With OpenCV, a video-capture object is created, which opens the camera and broadcasts the frames into the system during the session. Those frames constitute the major input of the gesture-recognition pipeline, which allows understanding the hand movements in real-time.

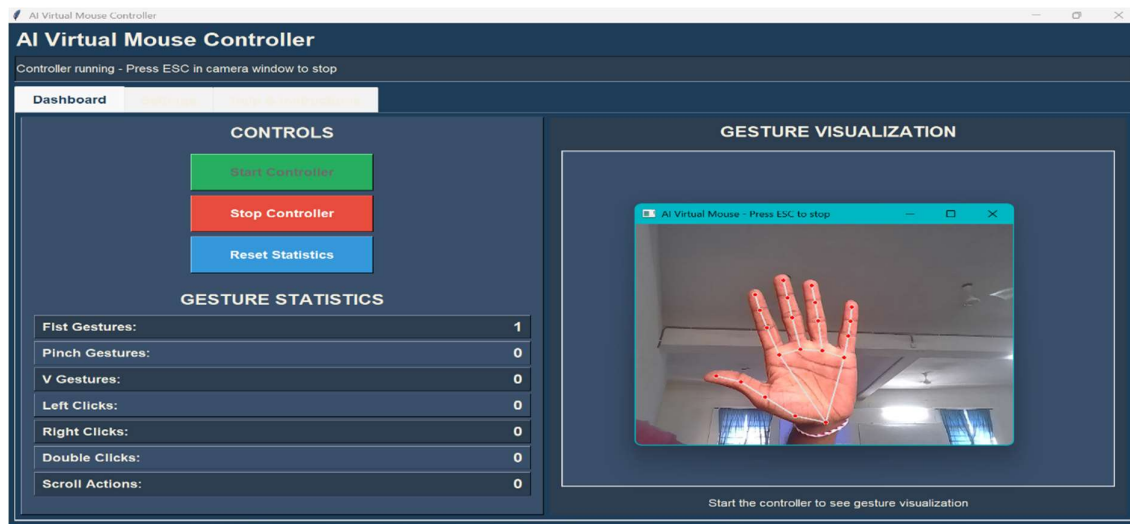


Fig: 8.2.2 Capturing video using the webcam (computer vision).

Frame Conversion and Pre-processing

Processing of each captured frame is done and the process of gesture analysis commences. Given that MediaPipe takes images in RGB format, the system becomes RGB-conversion of the frames of the default BGR-representation of OpenCV. This conversion enables MediaPipe to recognise the hand landmarks of a frame by frame, which assures accurate tracking in movement.

Mapping Webcam Co-ordinates to Screen Co-ordinates.

To interpret the movements of the hands to create valuable actions in the cursor, the system establishes an interaction threshold in the camera stream. This border is modeled as an area of a rectangle that confines the movement area in the webcam. The fingertip coordinates reached within this zone are scaled and adjusted to fit the entire display resolution enabling the cursor to move across the entire screen smoothly

Gesture Interpretation and Finger State Detection.

The system then identifies the raised fingers in each frame by analyzing the locations of the landmark points obtained by MediaPipe. The tip coordinates are calculated in order to find out patterns of various gestures. Depending on the identified finger formation, the system causes the relevant action of the mouse.

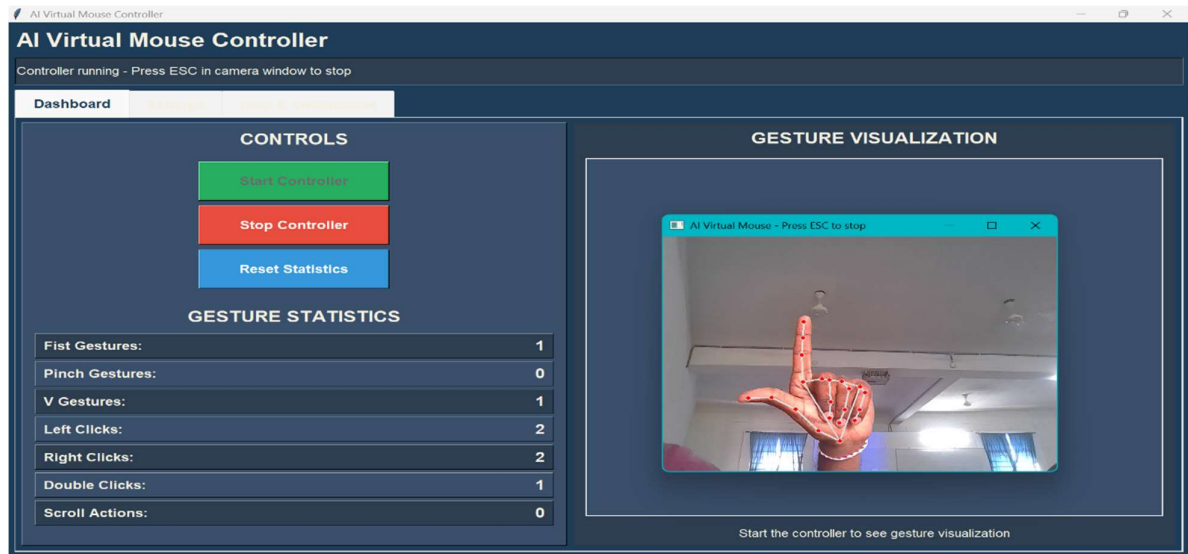


Fig: 8.2.3 Detection of which finger is up.

Gesture-Based Mouse Controls are included in the list of computer technologies as well.

Cursor Movement

The index finger is raised, and only the thumb finger is raised or when the index finger and the middle finger are raised at the same time, the cursor is controlled. PyAutoGUI maps the movement of the raised finger directly to the cursor motion enabling the user to move about the screen.

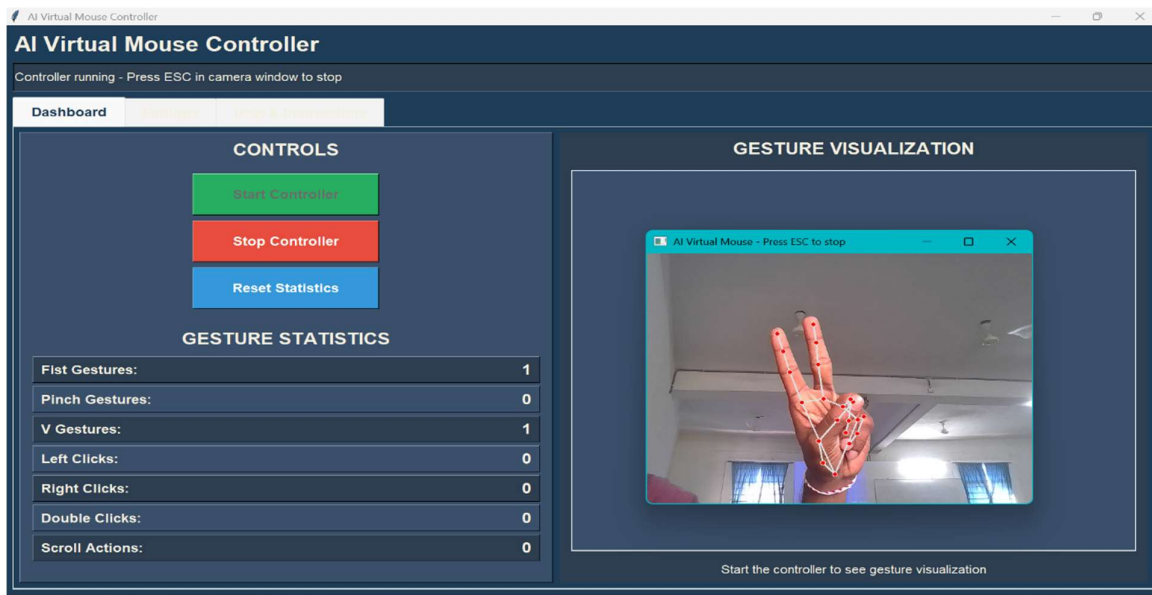


Fig:8.2.4 Mouse cursor moving around the computer window.

Left Click

An action of left clicking is performed when the index finger and thumb bring each other close to each other. The system calculates the distance between the two fingertips, and when it is lower than a certain set code, a left click command is sent.

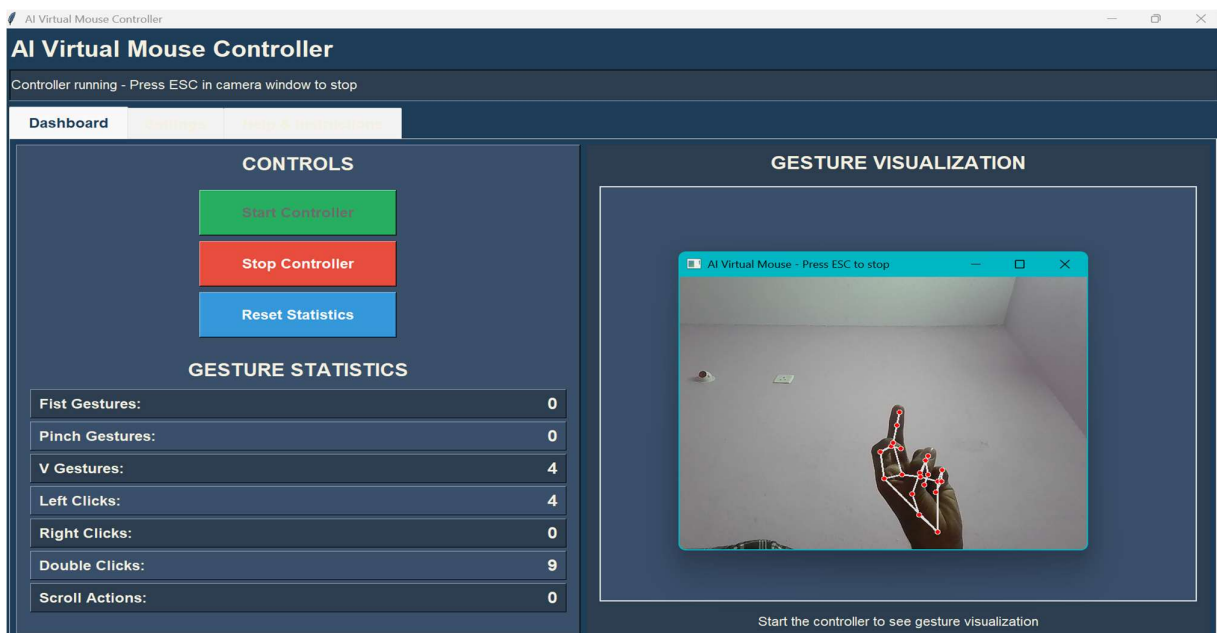


Fig:8.2.5 Gesture for the computer to perform left button click.

Right Click

A right-click gesture is identified upon a congruence of the index and middle fingers in a particular distance. After this condition is identified, the system does a right-click operation.

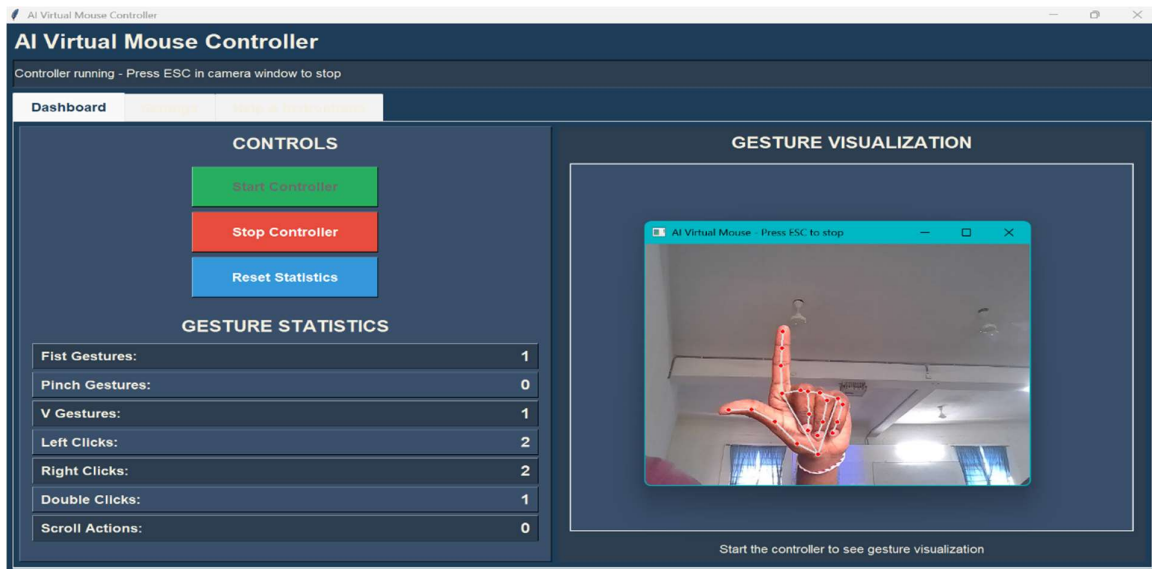


Fig:8.2.6 Gesture for the computer to perform right button click.

Scroll up

One way in which the scrolling up is initiated is by keeping the index and the middle fingers separated and moving up the screen. The direction in which the scrolling is performed depends on the comparative movement of the two fingers.

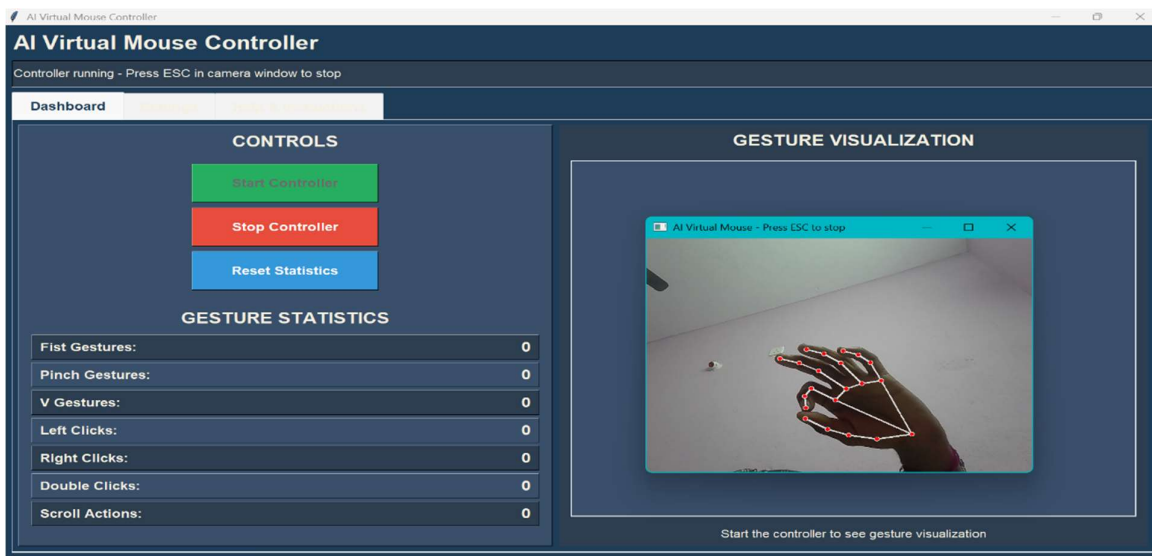


Fig: 8.2.7 Gesture for the computer to perform scroll up function.

Scroll down

On the same note, the scrolling down is performed with the two fingers to be unbundled and scrolled downwards. This movement is read by the system and the scroll-down command is issued using PyAutoGUI.

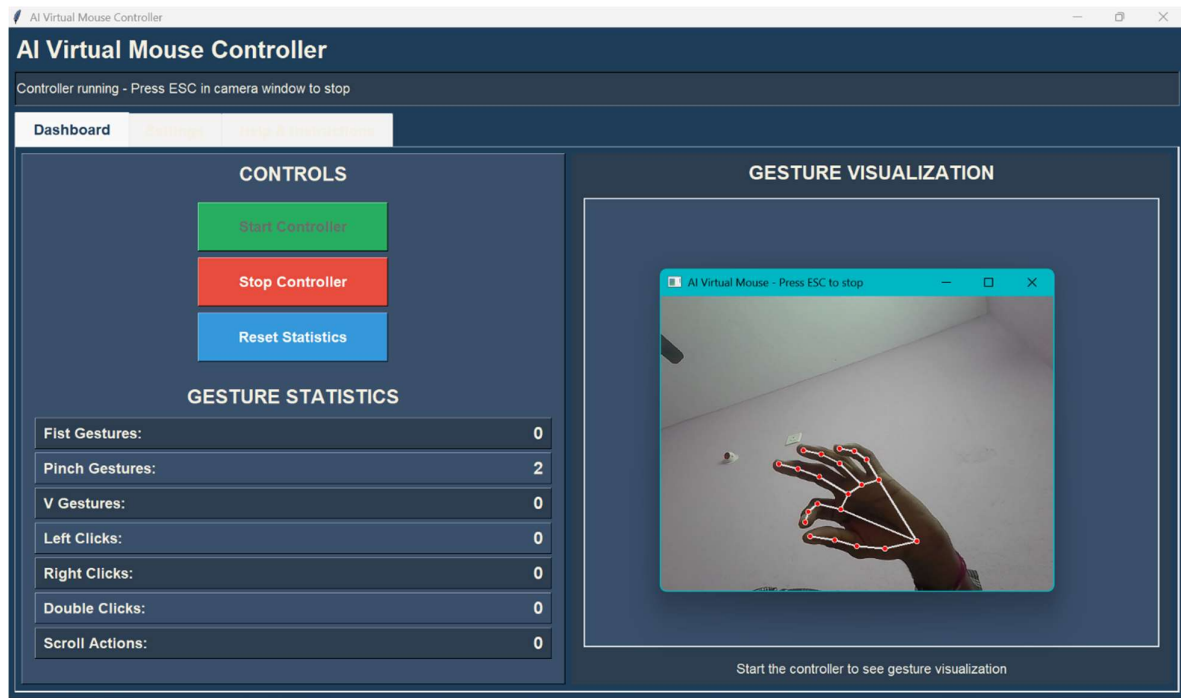


Fig: 8.2.8 Gesture for the computer to perform scroll down function.

No Action State

The system gets into a neutral position when all the fingers are extended and no mouse actions are performed. This will help avoid unwanted commands when the hand of the user is not busy or resting.

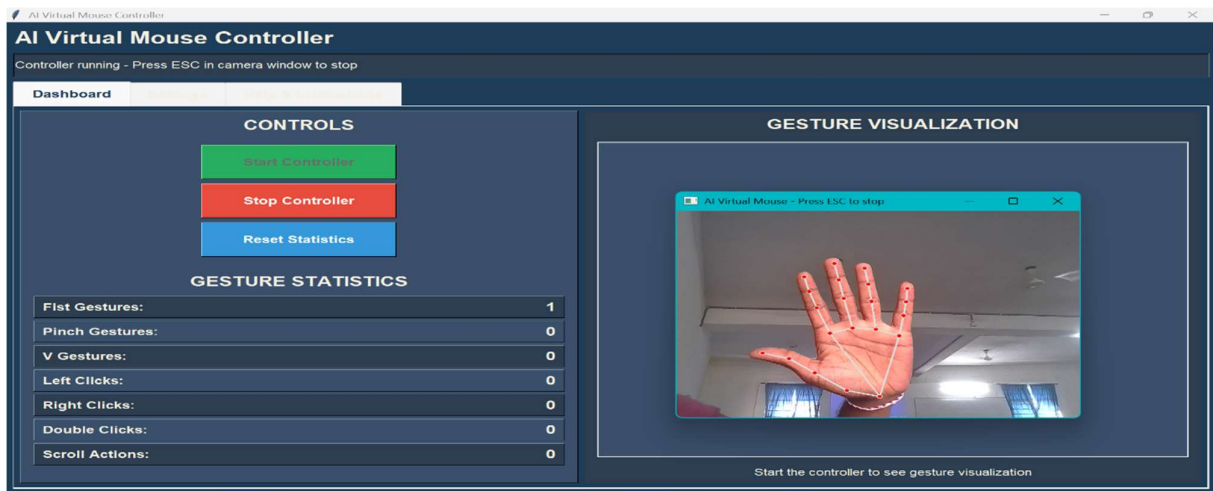


Fig: 8.2.9 Gesture for the computer to perform no action.

8.3 PERFORMANCE ANALYSIS AND MEASURES:

Speed and Responsiveness:

- ❖ Measure the time taken for each step in your system, such as eye detection, gesture recognition, and cursor control.
- ❖ Assess the responsiveness of the system to user inputs, such as hand gestures or voice commands, by measuring the latency between input and action.
- ❖ Optimize algorithms and implementations to reduce latency and improve real-time performance.

Accuracy:

- ❖ Evaluate the accuracy of eye detection and tracking algorithms by comparing the detected eye positions with ground truth data.
- ❖ Measure the precision of gesture recognition by comparing the recognized gestures with the intended gestures performed by the user.
- ❖ Implement calibration routines to fine-tune the accuracy of cursor control based on the user's eye movements and gestures.

Robustness:

- ❖ Test the system under various lighting conditions, camera angles, and user environments to assess its robustness.
- ❖ Evaluate the system's ability to handle occlusions, such as partial obstruction of the user's face or hands, without significant degradation in performance.
- ❖ Implement error handling mechanisms to gracefully handle unexpected inputs or failures, ensuring the system remains stable and responsive.

Resource Utilization:

- ❖ Monitor the system's resource utilization, such as CPU and memory usage, to ensure efficient use of hardware resources.
- ❖ Profile the performance of individual components, such as image processing algorithms and gesture recognition modules, to identify bottlenecks and optimize resource usage.

User Experience:

- ❖ Gather feedback from users through surveys, interviews, or usability testing sessions to assess their satisfaction with the system's performance.
- ❖ Identify areas for improvement based on user feedback, such as interface design, responsiveness, and ease of use.
- ❖ Iterate on the design and implementation based on user feedback to enhance the overall user experience.

Scalability:

- ❖ Assess the scalability of the system to handle varying levels of user interactions, such as increasing the number of simultaneous users or supporting different input modalities.
- ❖ Evaluate the system's performance under load and identify any scalability limitations or constraints.
- ❖ Implement scalability enhancements, such as distributed processing or parallelization, to improve the system's ability to handle increased workload.

By conducting thorough performance analysis and measurement across these dimensions, you can ensure that your project meets its performance goals and delivers a reliable and responsive user experience.

A hand-gesture- controlled virtual mouse could provide an alternative method for people with disabilities who may have difficulty using a traditional mouse or keyboard. This technology can make it easier for them to interact with computers and other devices. A hand gesture-controlled virtual mouse could also be useful for people who prefer to work or play games without being tethered to a physical mouse touchpad. This model would allow them to control their devices without the need for a physical interface.

Depending on the technology used, a hand gesture-controlled virtual mouse may offer a higher degree of accuracy and speed than traditional mice or video editing. The success of this technology will depend on the user experience it provides. If the technology is easy to use, reliable and provides an intuitive interface, likely to be well-received. However, if the technology is difficult to use, unreliable, or unintuitive, users may quickly abandon it.

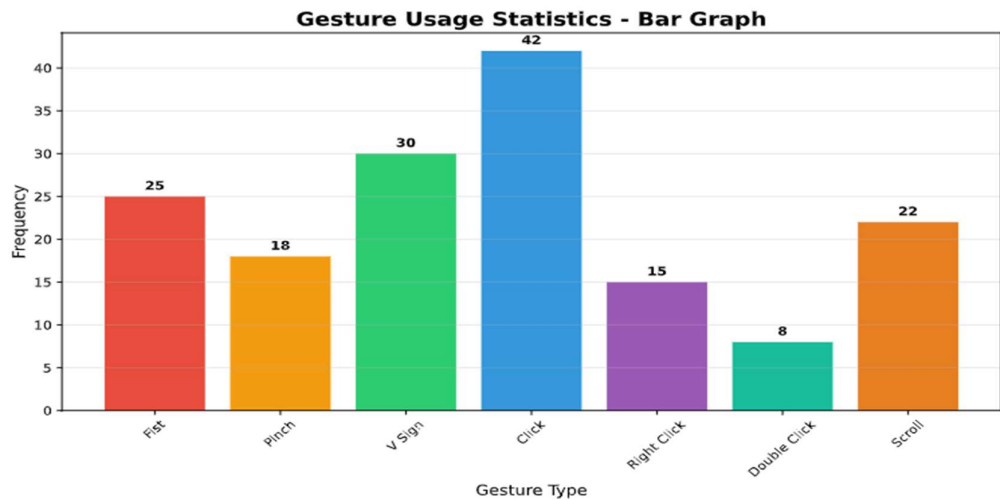


Fig: 8.3.1 Gesture Usage Statistics.

9. TESTING

Testing an AI Virtual Mouse System involves a systematic approach to ensure accuracy, responsiveness, and reliability across various scenarios and environments. Here's a structured guide to testing your AI virtual mouse:

Setup and Environment Preparation

❖ Hardware Requirements:

A functional webcam or depth sensor with suitable resolution.

A computer with sufficient processing power and GPU capability (if required).

❖ Software Setup:

Install the necessary AI models, libraries (e.g., OpenCV, TensorFlow, or PyTorch), and drivers.

Ensure the testing environment matches the intended deployment platform (e.g. Windows, macOS, Linux).

❖ Lighting and Background:

Test under different lighting conditions (low light, bright light, natural light).

Use varied backgrounds to check for noise handling and robustness.

Testing Scenarios

Cursor Movement

❖ Smoothness and Precision:

Move your hand or gesture slowly and rapidly to test the system's responsiveness.

Evaluate whether the cursor aligns accurately with your gestures.

❖ Boundary Testing:

Test the cursor near screen edges to ensure it doesn't overshoot or lag.

Gesture Recognition

❖ Basic Gestures:

Test core gestures like left-click, right-click, double-click, drag, and scroll.

Ensure gestures are consistently recognized across multiple attempts.

❖ **Complex Gestures:**

Include multi-step or compound gestures to see if the AI differentiates them effectively.

Speed and Latency

❖ **Response Time:**

Measure the time between a gesture and the corresponding cursor movement.

Identify if delays occur under specific conditions.

User Adaptability

❖ **Multiple Users:**

Test with users of different hand sizes, skin tones, and movement styles.

Evaluate the system's ability to adapt or learn user-specific variations.

Edge Cases

❖ **Occlusions:**

Test scenarios where parts of the hand or face are temporarily obscured.

Assess recovery speed and accuracy after occlusion.

❖ **Distractions:**

Introduce external movements (e.g., waving objects) to test the system's robustness.

Ensure the AI focuses on intended gestures.

❖ **Unintended Gestures:**

Perform random movements to check if the system misinterprets them as valid command.

Performance Metrics

❖ **Accuracy Rate:**

Percentage of correctly recognized gestures.

❖ **False Positive/Negative Rate:**

Rate of unintended actions or missed gestures.

❖ **Latency:**

Average time taken to process input and reflect output.

❖ **Usability Rating:**

User feedback on ease of use, comfort, and intuitive design.

Real-World Usability Testing

Deploy the system in actual usage scenarios such as:

- i. Navigating websites.
- ii. Playing games.
- iii. Managing tasks like dragging files or interacting with productivity software.
- iv. Gather feedback from real users to refine and improve the system.

Debugging and Optimization

- i. Analyze logs to identify frequent failures or lag sources.
- ii. Optimize algorithms for better performance on diverse hardware setups.
- iii. Retrain the AI model with additional data if recognition accuracy is low.

Testing ensures your AI virtual mouse provides a seamless, intuitive, and inclusive experience for all users.

10. RESULT

Enhanced Security System

We successfully implemented a robust user authentication mechanism that requires username and password verification before accessing the virtual mouse controller. This security layer protects against unauthorized usage and ensures only registered users can operate the system.

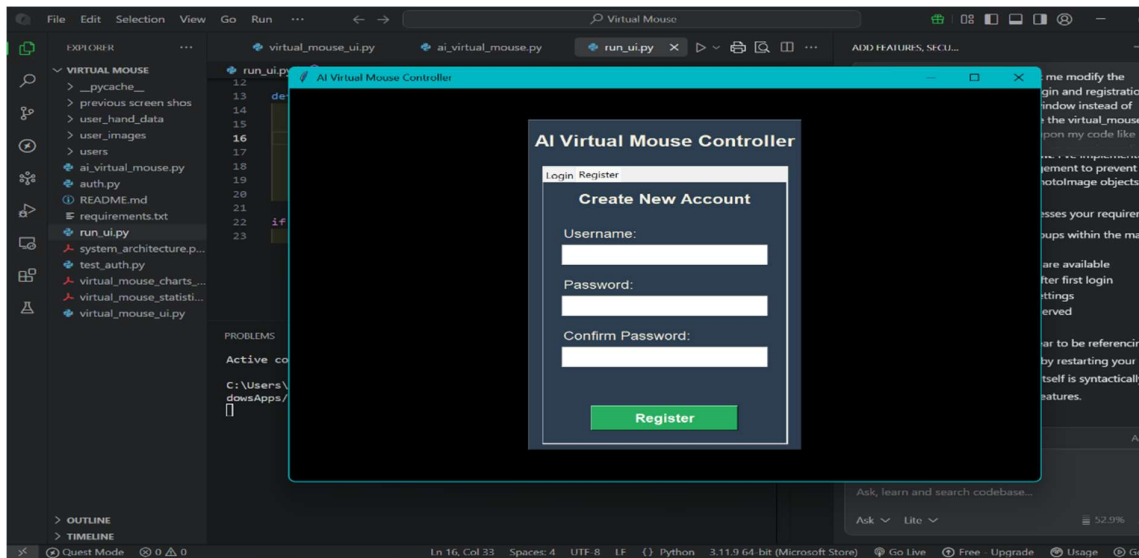


Fig: 10.1 Login page

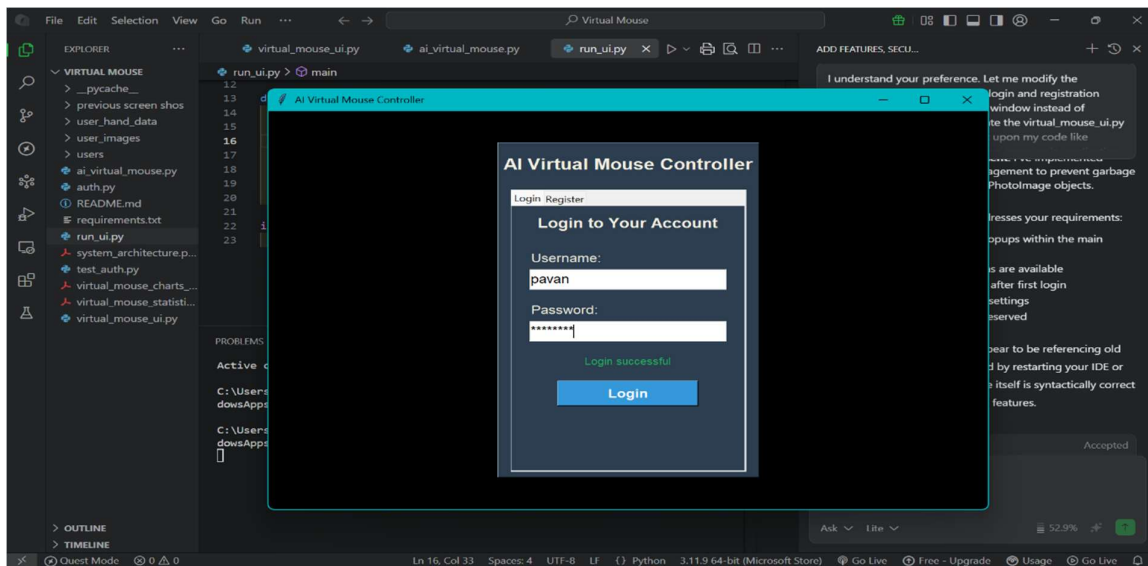


Fig: 10.2 Successful Login

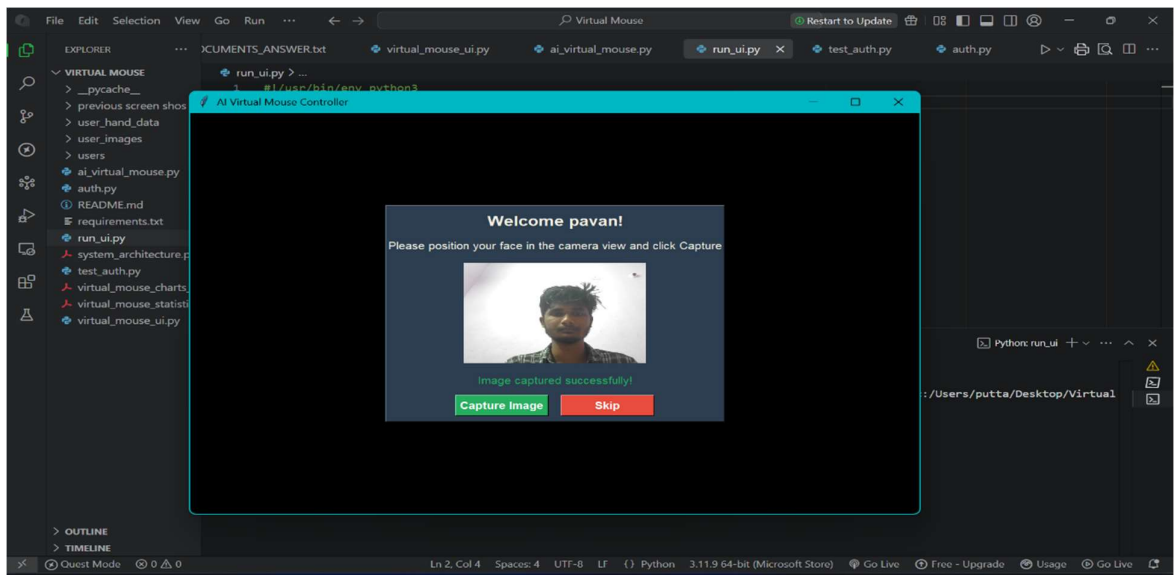


Fig: 10.3 User Image Capturing and Face Recognition

Personalized Hand Recognition

The system now uniquely identifies and responds only to the authenticated user's hand gestures, even when multiple hands are detected simultaneously. This personalized recognition prevents interference from other users and maintains precise control accuracy.

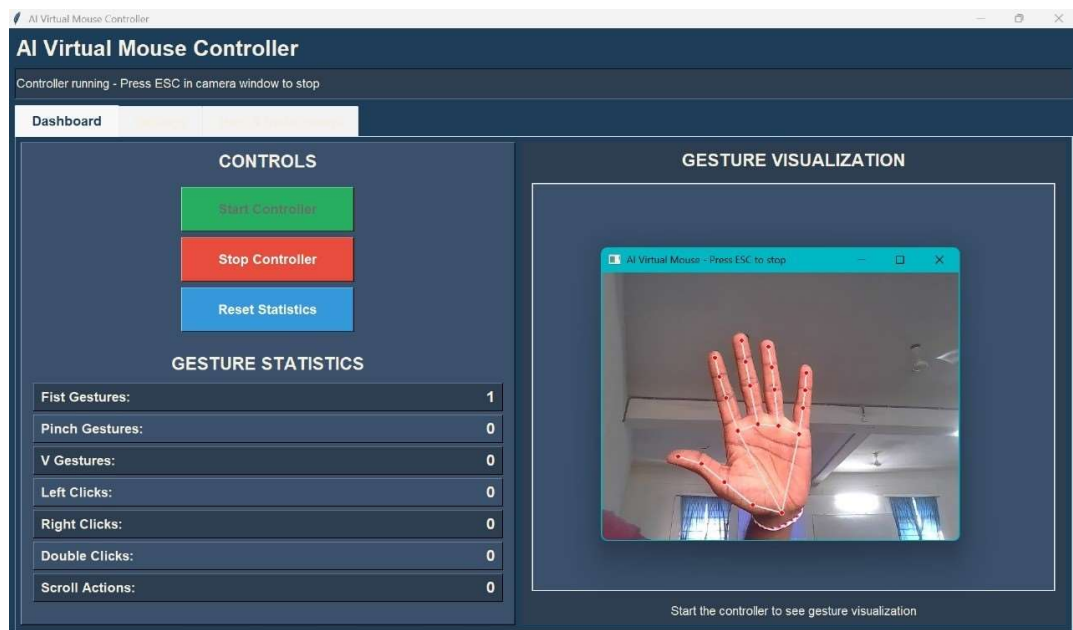


Fig: 10.4 Hand Gesture Recognition

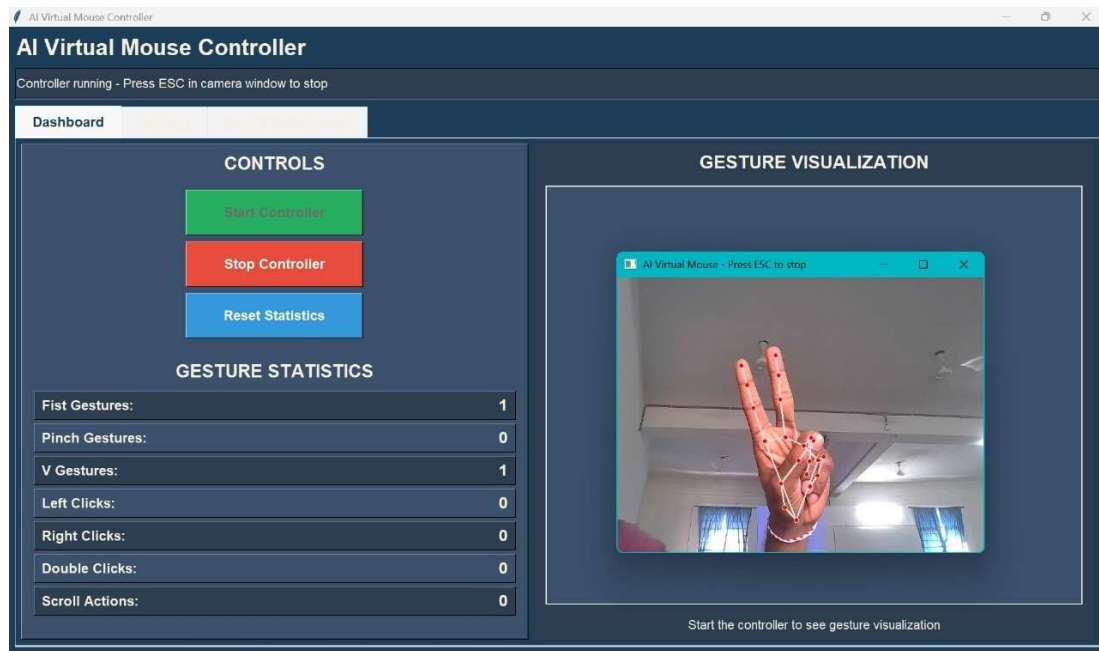


Fig: 10.5 Gesture for moving cursor

Intuitive User Interface

We developed a colorful, tab-based dashboard interface with Dashboard, Settings, and Help sections that are clearly visible and easy to navigate. The interface features customizable themes and a clean layout that enhances user experience.

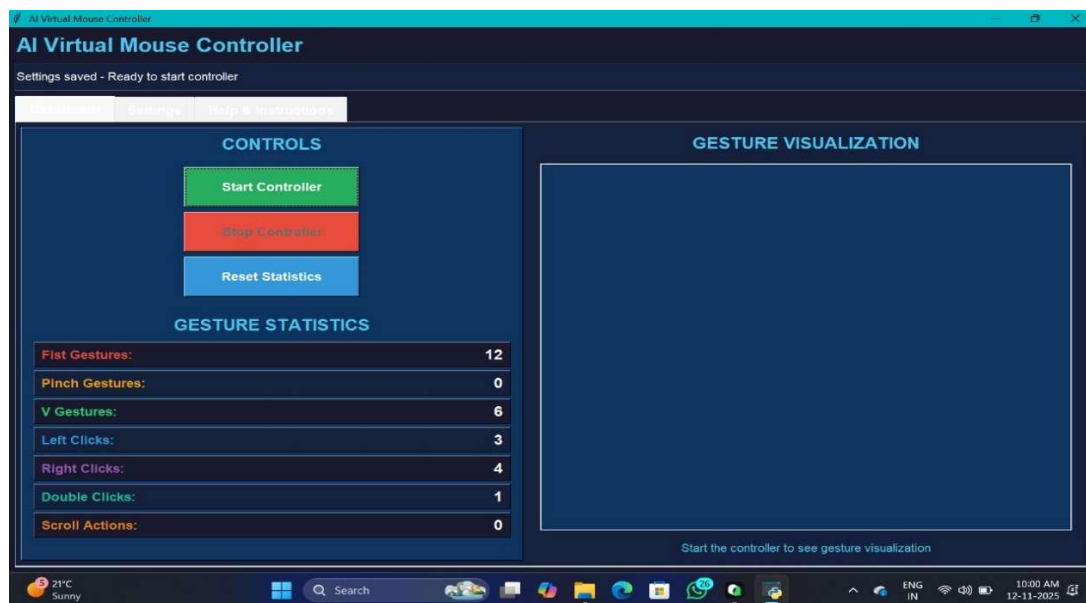


Fig: 10.6 Controller and Gesture Statistics

Comprehensive Gesture Control

The virtual mouse supports a full range of gesture-based controls including cursor movement, single and double-clicking, right-clicking, scrolling, and system adjustments for brightness and volume. These gestures provide complete computer control without a physical mouse.

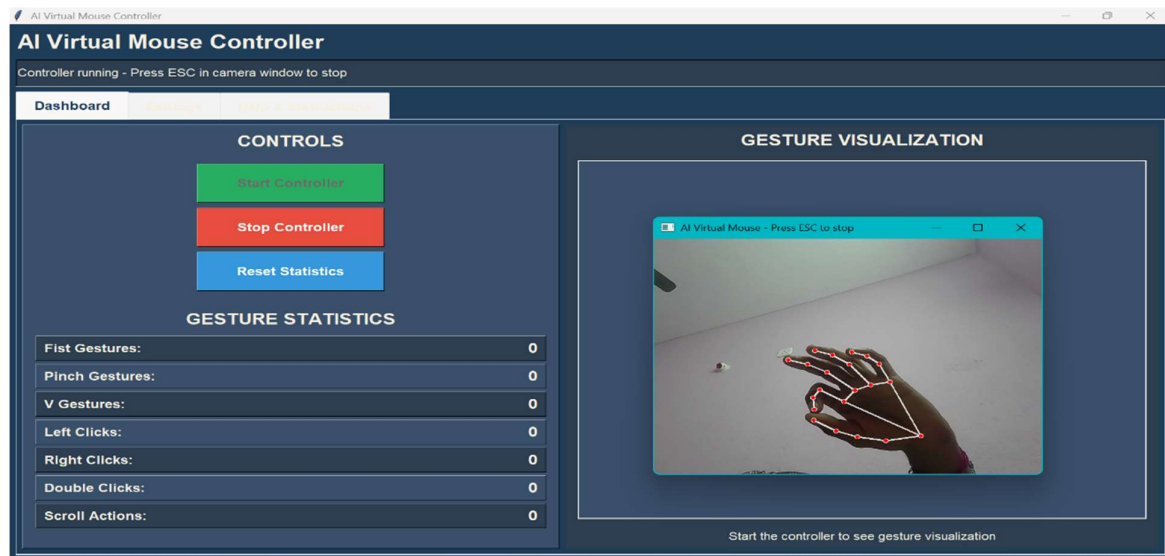


Fig: 10.7 Pinch gesture for Select and Scroll

Customizable Performance Settings

Users can fine-tune the system performance through adjustable parameters such as mouse sensitivity, scroll speed, click delay, and hand detection confidence. Additional options include gesture mode selection and theme customization for a personalized experience.

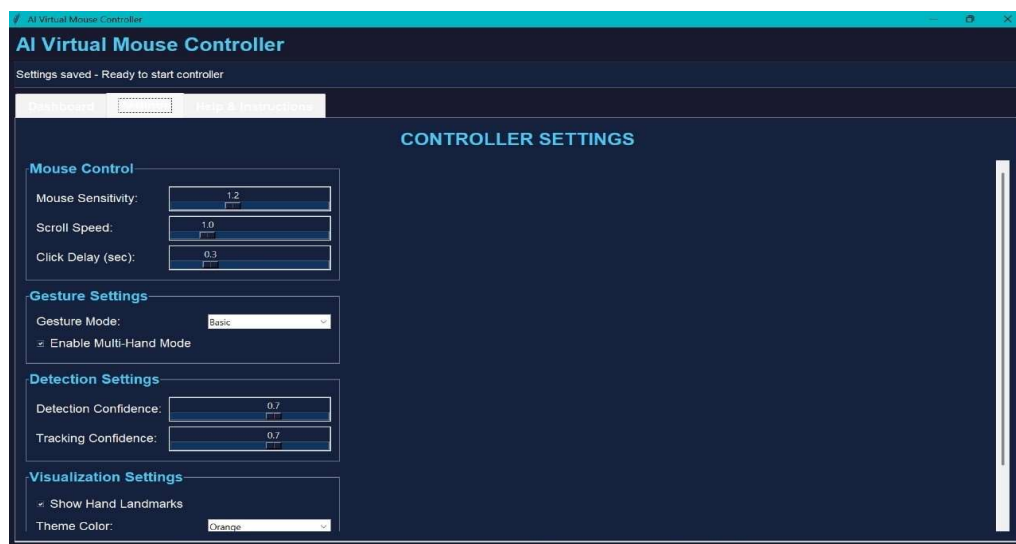


Fig: 10.8 Controller Settings

Detailed Analytics and Visualization

The dashboard provides real-time statistics and visual feedback of gesture usage through charts and graphs. Users can track their interaction patterns and monitor system performance with detailed metrics.

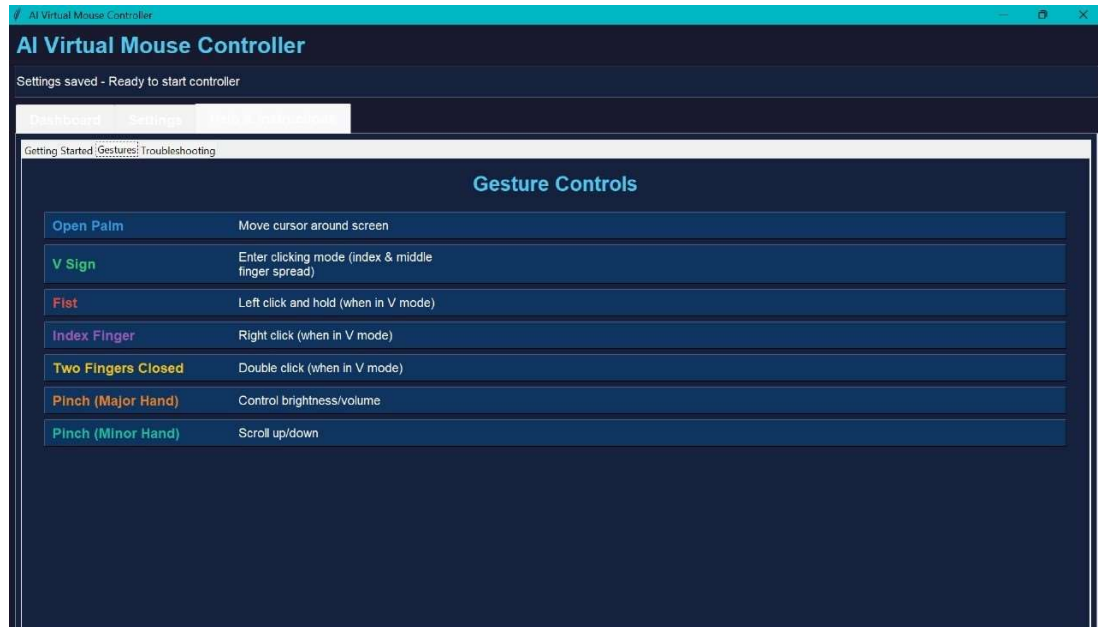


Fig: 10.9 Gesture Controls Information

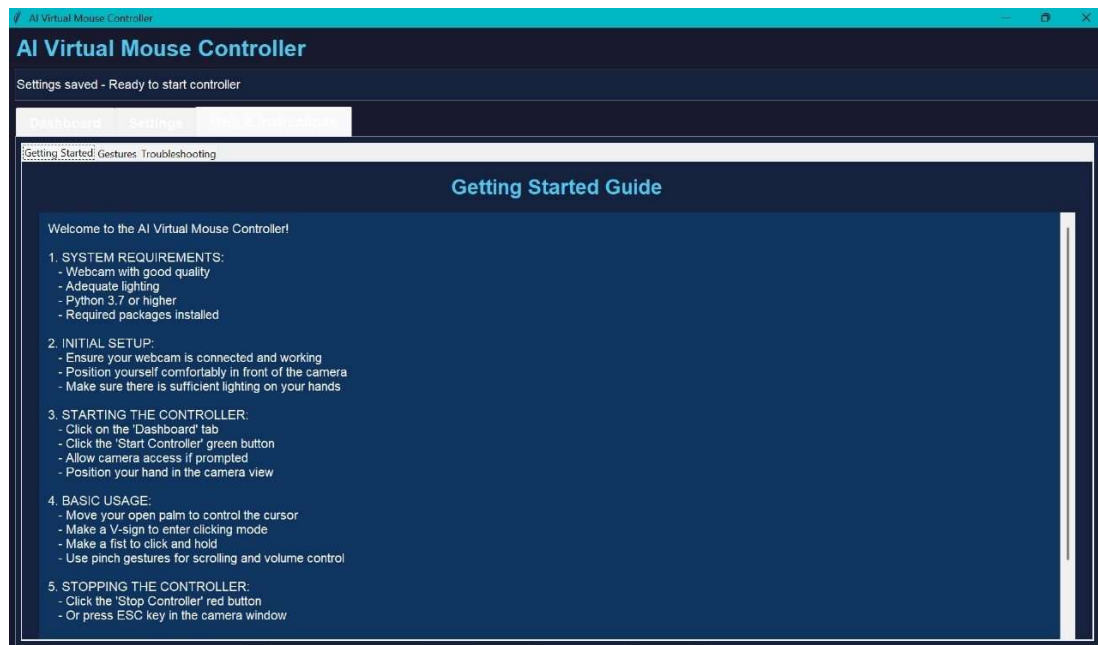


Fig: 10.10 Guide for AI Virtual Mouse Controller

11. FUTURE ENHANCEMENTS

The application needs a number of features and enhancements to be more versatile, accurate, and user-friendly in a variety of settings. The required upgrades and functionalities are described as follows:

Intelligent Recognition Algorithm

An adjustable zoom-in/out function is needed to increase the covered range because the existing recognition mechanism can only detect objects within a 25 cm radius. This function can automatically modify the focus rate dependent on the proximity between both the operator and the video camera.

Increased Efficiency

The hardware of the device, which comprises the processor's processing speed and the amount of the memory space, has a significant impact on the download speed. RAM and the webcam's capabilities. As a result, the program might run more effectively when it's installed on a good computer with a webcam that works well in various lighting conditions.

12. CONCLUSION

The AI-Based Virtual Mouse shows how computer vision and hand-gesture recognition can create an alternative to most traditional hardware devices used to interact with computers on a daily basis. The system uses a standard webcam to perform hand-tracking in real time, recognize finger gestures, and convert them into accurate mouse actions like moving the cursor, clicking, scrolling, and dragging while the user interacts with the system. This project illustrates that touch-free interaction is realistic without the need for any auxiliary devices or specialized sensors, making it practical for everyone by being affordable, accessible, and non-disruptive to the user.

The system proved its reliability throughout the development process under normal lighting conditions while simultaneously responding smoothly to the user's gestures. The modular design also allowed every component in the system (hand-detection, gesture analysis, cursor mapping, and UI feedback) to work autonomously and collaboratively, creating the opportunity to easily address performance challenges and suggestions for enhancements in future iterations. Moreover, the proposal highlights the ability of gesture-based interfaces to be used in health-sensitive settings (e.g., hospital and treatment settings), disability (adaptable and accessible to any user), and settings where traditional interface devices may not be available or realistic (e.g., laboratory settings).

Overall, this project provides a practical and creative approach to human-computer interface. By furthering recognition accuracy of gestures, feature performance for multiple hands or bodies, and greater user discretion through customization, the AI Virtual Mouse framework could evolve to be a sophisticated alternative to traditional input devices, while also reinforcing computing scenarios based on more natural, intuitive, and hands-free experiences.

13. BIBLIOGRAPHY

1. OpenCV Documentation: <https://opencv.org/documentation/>
2. PyAutoGUI Documentation: <https://pyautogui.readthedocs.io/en/latest/index.html>
3. "Python Programming for Beginners: An Introduction to the Python Computer Language and Computer Programming" by Jason Cannon, Independently published, 2019.
4. Quam, D.L., et.al. (1990). Gesture Recognition with a Dataglove. In IEEE conference on Aerospace and Electronics (pp. 755-760).
5. Guoli Wang., et.al. (2015). Optical Mouse Sensor-Based Laser Spot Tracking for HCI input, Proceedings of the Chinese Intelligent Systems Conference (pp. 329-340).
6. Baldauf, M., and Frohlich, p. (2013). Supporting Hand Gesture Manipulation of Projected Content with mobile phones. In the European conference on computer vision (pp. 381-390).
7. Roshnee Matlani., Roshan Dadlani., Sharv Dumbre., Shruti Mishra., & Abha Tewari. (2021). Virtual Mouse Hand Gestures. In the International Conference on Technology Advancements and innovations (pp. 340-345).
8. Mayur, Yeshi., Pradeep, Kale., Bhushan, Yeshi., & Vinod Sonawane. (2016). Hand Gesture Recognition for Human-Computer Interaction. In the international journal of scientific development and research (pp. 9-13).
9. Shriram, S., Nagaraj, B., Jaya, J., Sankar, S., & Ajay, P. (2021). Deep Learning Based Real Time AI Virtual Mouse System Using Computer Vision to Avoid COVID-19 Spread. In the Journal of Healthcare Engineering (pp. 3076-3083).
10. Steven Raj, N., Veeresh Gobbur, S., Praveen., Rahul Patil., & Veerendra Naik. (2020). Implementing Hand Gesture Mouse Using OpenCV. In the International Research Journal of Engineering and Technology (pp. 4257-4261).
11. Sneha, U., Monika, B., & Ashwini, M. (2013). Cursor Control System Using Hand Gesture Recognition. In the International Journal of Advanced Research in Computer and Communication Engineering (pp. 2278-1021).