Infosys Springboard Virtual Internship 6.0
Completion Report

Team Details

BATCH-A

START:- 12-OCT-2025

Names:

S.NO	NAME
1.	SNEHIL GHOSH
2.	AMRUTHA VERSHINI GARI
4.	GAUTAM N CHIPKAR
5.	AYUSH GORGE

Internship Duration: 8 weeks

Infosys Springboard Virtual Internship 6.0
Completion Report

1. Project Title

Controlling Volume with Hand Gestures Using a Webcam

2. Project Objective

The objective of this project is to design and develop a computer vision based system that enables users to control microphone or system volume using predefined hand gestures captured through a webcam.

The project aims to:

- Detect and track human hand landmarks in real time using MediaPipe Hands.
- Interpret geometric relationships between key hand landmarks such as thumb tip and index finger tip.
- Translate recognized gestures into corresponding volume control commands.
- Enable touch-free and intuitive interaction with system audio controls.
- Improve accessibility and usability in smart environments and assistive technology scenarios.

3. Project description in detail

The “Controlling Volume with Hand Gestures” project is a real-time gesture recognition system that leverages computer vision and machine learning techniques to replace traditional physical volume controls.

Using a live webcam feed, the system captures video frames through OpenCV and processes them using MediaPipe Hands, which detects 21 hand landmarks per hand. These landmarks are analyzed to determine gesture states such as pinch distance, finger openness, and hand posture. Based on these detected gestures, volume levels are dynamically adjusted.

To ensure stable and safe interaction with the Windows audio system, microphone volume control is implemented using Pycaw and Windows Audio Session API (WASAPI). Due to strict COM threading requirements, all audio-related operations are isolated into a dedicated child process, improving system reliability and preventing crashes.

Infosys Springboard Virtual Internship 6.0 Completion Report

A Tkinter-based graphical user interface (GUI) serves as a heads-up display (HUD), displaying real-time camera previews, detected gestures, and volume levels. The application delivers smooth performance at real-time frame rates and provides an intuitive, contactless user experience.

4. Timeline Overview

Week	Activities Planned	Activities Completed
Week 1	Project understanding, requirement analysis and tool research	Finalized project scope and studied MediaPipe and OpenCV
Week 2	Webcam integration and hand detection	Implemented hand landmark detection
Week 3	Gesture logic development	Calculated pinch distance and finger states
Week 4	Volume mapping and audio control	Integrated Pycaw with an isolated child process
Week 5	GUI design	Developed Tkinter-based HUD
Week 6	System integration	Connected gesture pipeline with audio control
Week 7	Testing and optimization	Improved stability and gesture smoothing
Week 8	Documentation and presentation	Prepared final report and PPT

5a. Key Milestones

Milestone	Description	Timeline
Project Kickoff	Finalized project scope, objectives, and selected tools such as OpenCV, MediaPipe, and Pycaw.	Week 1
Gesture Detection Ready	Implemented hand landmark detection and thumb–index finger distance–based gesture recognition.	Week 4
Audio Control Integrated	Enabled microphone volume control, mute, and unmute using Pycaw and Windows audio APIs.	Week 6

Infosys Springboard Virtual Internship 6.0
Completion Report

Stable Architecture Achieved	Introduced the child subprocess to isolate COM and Pycaw for stable real-time operation.	Week 8
Final Submission	Completed system integration, testing, documentation, and project presentation.	Week 8

5b. Project execution details

The project was executed using a modular, pipeline-based architecture that clearly separated vision processing, gesture interpretation, audio control, and user interface components.

Real-time video input was captured from the system webcam using OpenCV, which handled camera access, frame capture, preprocessing, and colour space conversion from BGR to RGB for MediaPipe compatibility. Frames were processed continuously to maintain stable frame rates and responsive gesture detection.

MediaPipe Hands was used for real-time hand detection and tracking. The framework identifies 21 hand landmarks, including fingertip and joint coordinates. These landmarks were analyzed to compute geometric parameters such as thumb–index finger distance, finger count, and hand dimensions. Gesture logic classified these measurements into control signals representing microphone volume levels.

Gesture outputs were mapped to proportional volume values, such as finger count–based scaling (0–5 fingers mapped to 0–100% volume). This ensured intuitive and consistent user interaction.

Microphone volume control was implemented using Pycaw, which interfaces with the Windows Audio Session API (WASAPI) to access and modify audio endpoint properties, including volume level and mute state. Since Pycaw relies on Windows COM interfaces with strict threading requirements, direct integration within GUI and gesture threads resulted in instability.

To resolve this, a dedicated child subprocess architecture was introduced. All COM and Pycaw operations were isolated into a separate Python process, where COM was initialized exactly once. The main application communicated with the child process via standard input and output streams.

A MicController communication layer managed command transmission, response handling, thread locking, and timeout control. The graphical user interface was developed using Tkinter,

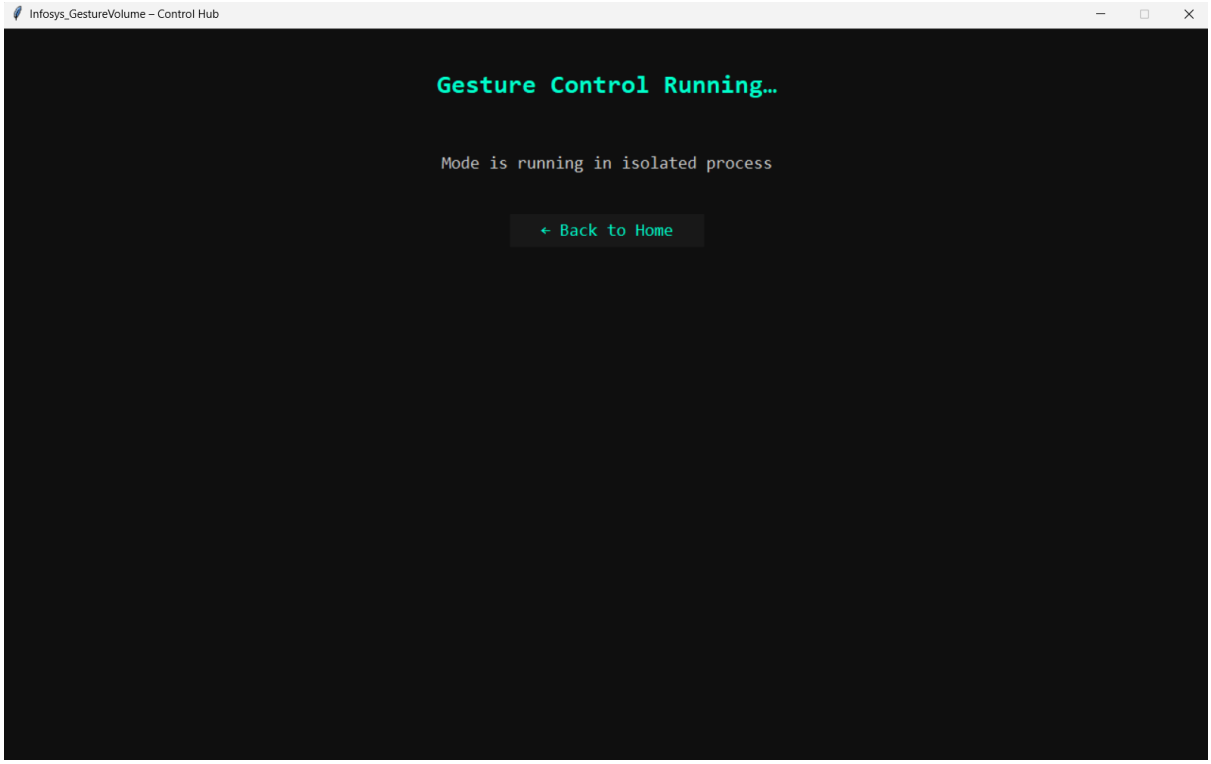
Infosys Springboard Virtual Internship 6.0
Completion Report

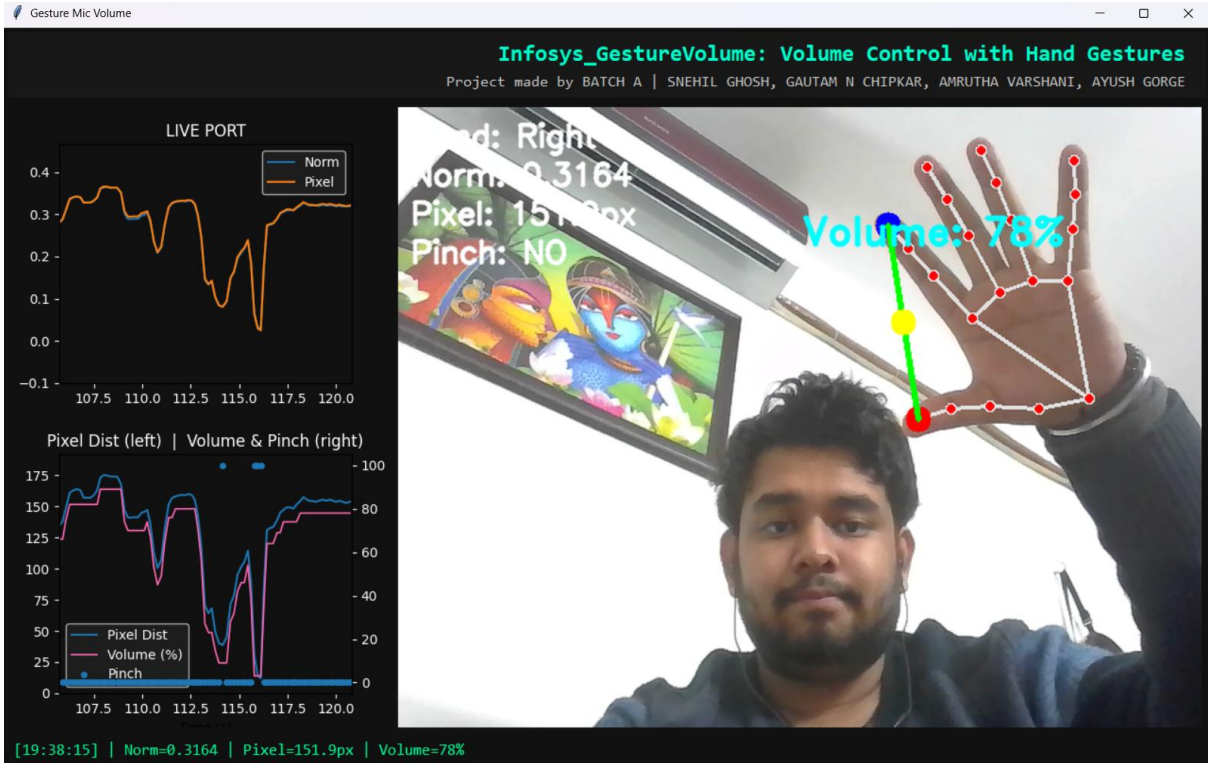
displaying the live video feed, detected gestures, and volume indicators. Matplotlib was embedded within the GUI to visualize volume levels and proximity metrics in real time.

This architecture ensured stability, real-time performance, and reliable system-level microphone control.

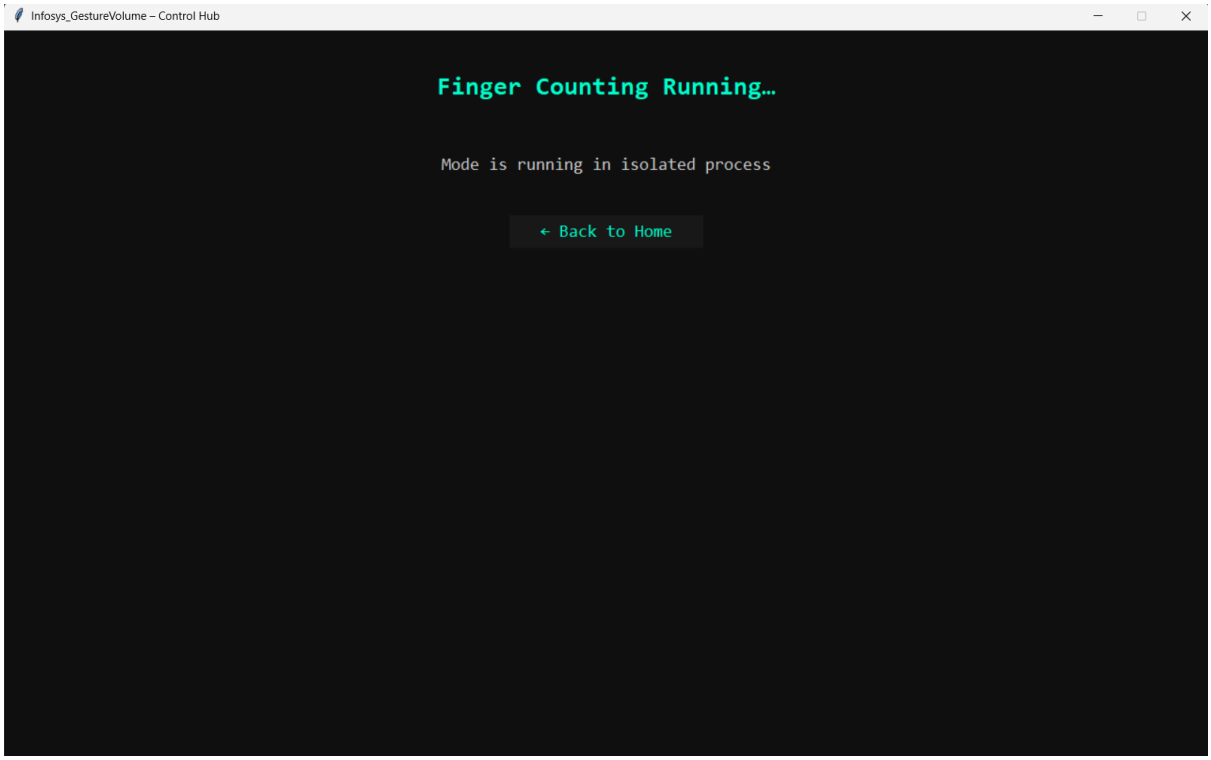
6. Snapshots / Screenshots

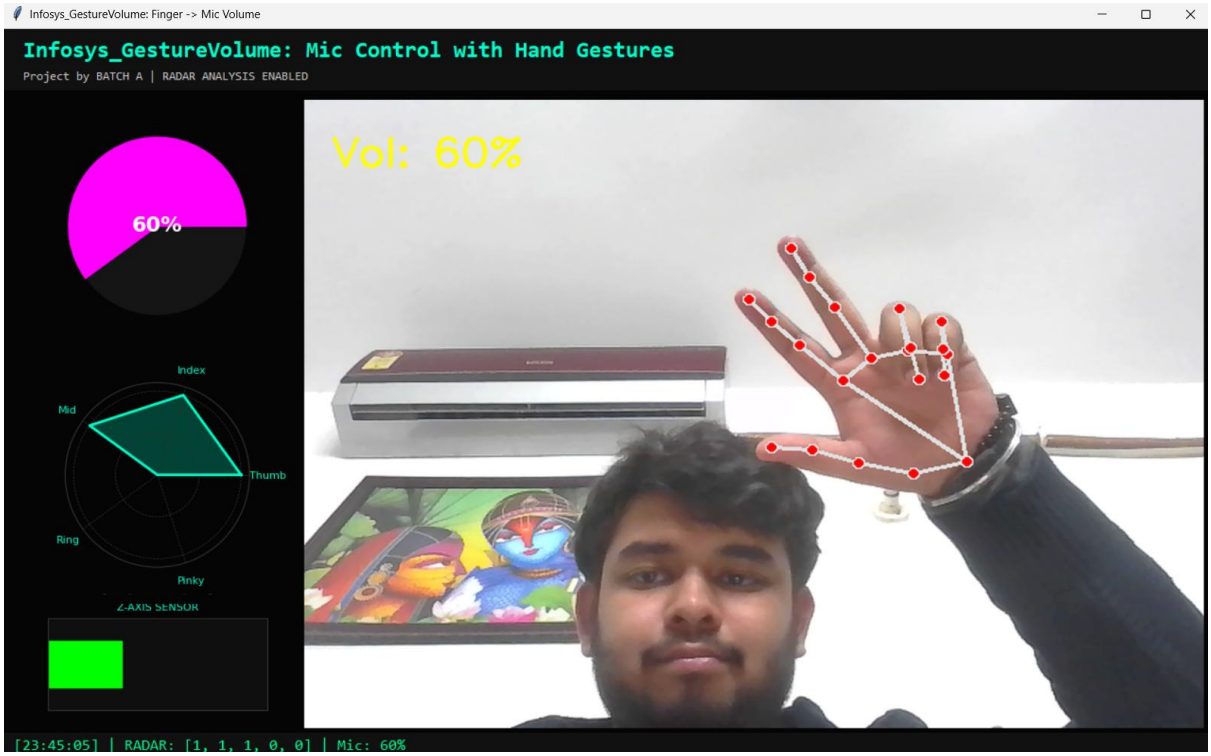
Infosys Springboard Virtual Internship 6.0
Completion Report





Infosys Springboard Virtual Internship 6.0
Completion Report





Infosys Springboard Virtual Internship 6.0
Completion Report

7. Challenges Face

During the development of the gesture-controlled microphone volume system, multiple challenges were encountered while integrating real-time computer vision, Windows audio APIs, and graphical user interfaces. Key challenges and their resolutions are summarized below:

- **Unstable Hand Landmark Detection:**
  MediaPipe Hands occasionally lost hand tracking in low-light conditions or unclear visibility, resulting in inconsistent gesture recognition. This was due to low detection and tracking confidence. The issue was resolved by tuning min_detection_confidence and min_tracking_confidence parameters, improving lighting conditions, and applying smoothing logic to landmark measurements, which significantly improved detection stability.

- **Inconsistent Gesture to Volume Mapping:**
  Directly using raw thumb–index finger distance values caused sudden volume jumps due to camera perspective variation and frame-to-frame noise. This was addressed by normalizing distance values using numpy.interp() and mapping them to a controlled volume range, ensuring smooth and predictable volume transitions.

- **Pycaw and COM Threading Instability:**
  Pycaw interfaces with Windows Core Audio through COM, which enforces strict threading and initialization rules. When Pycaw was executed directly within the Tkinter GUI and gesture-processing threads, it resulted in COM initialization errors, application crashes, and unstable microphone control.

  To resolve this, a dedicated child subprocess architecture was introduced. All COM and Pycaw operations were isolated into a separate Python process, where COM was initialized exactly once using CoInitialize(). This separation eliminated thread interference, stabilized audio control, and ensured reliable microphone volume updates.

- **GUI Framework Selection (Tkinter vs Streamlit):**
  Initial consideration of Streamlit revealed limitations for real-time gesture-based applications. Streamlit's script rerun behaviour, browser-based rendering, and event-driven execution model caused latency, repeated hotkey registration, and reduced control over threading and real-time loops. In contrast, Tkinter provided a native desktop event loop, precise thread coordination, low-latency rendering, and seamless

Infosys Springboard Virtual Internship 6.0 Completion Report

integration with OpenCV and the child subprocess architecture. Therefore, Tkinter was chosen to ensure stable real-time performance and system-level integration.

9. Learnings & Skills Acquired

The development of the gesture-controlled microphone volume system provided valuable hands-on experience across computer vision, system-level programming, and GUI development. The key learnings and skills acquired during this project are summarized below:

- **Computer Vision with OpenCV:**
  Gained practical experience in capturing real-time webcam input, performing frame preprocessing, managing color space conversions (BGR to RGB), and maintaining stable frame rates for real-time applications.
- **Hand Tracking and Gesture Recognition (MediaPipe Hands):**
  Learned to work with MediaPipe Hands for detecting and tracking 21 hand landmarks in real time. Developed an understanding of landmark-based gesture interpretation using geometric relationships such as fingertip distance, finger count, and hand dimensions.
- **Gesture-to-Control Mapping Logic:**
  Acquired experience in converting raw gesture measurements into meaningful control signals. Implemented normalization and interpolation techniques to map gesture values smoothly to microphone volume levels.
- **System-Level Audio Control (Pycaw & WASAPI):**
  Developed a strong understanding of Windows Audio Session API (WASAPI) and Pycaw for controlling microphone volume and mute states programmatically. Learned the importance of proper COM initialization and cleanup.
- **COM Architecture and Process Isolation:**
  Gained insight into Windows COM threading constraints and implemented a child subprocess architecture to isolate COM-dependent audio operations, ensuring application stability and reliability.
- **GUI Development with Tkinter:**
  Designed and implemented a responsive desktop GUI using Tkinter, integrating live video feeds, gesture indicators, and real-time system feedback.
- **Multithreading and Real-Time System Design:**
  Learned to coordinate multiple components such as video capture, gesture processing,

Infosys Springboard Virtual Internship 6.0 Completion Report

GUI updates, and inter-process communication without blocking the main application loop

9. Testimonials from Team

The gesture-controlled volume project provided a strong practical learning experience for the team. Throughout the internship, the team adapted quickly to computer vision concepts, system-level programming, and real-time application development. Regular participation in discussions, consistent progress updates, and collaborative problem-solving helped ensure steady project advancement. The team demonstrated commitment and professionalism while balancing academic responsibilities alongside project deliverables.

10. Conclusion

Overall, the gesture-controlled microphone volume project served as an effective learning platform, strengthening both technical and analytical skills. The team gained hands-on experience in computer vision using OpenCV and MediaPipe, system-level audio control through Pycaw and Windows Audio APIs, GUI development with Tkinter, and stable architecture design using process isolation. The project enhanced understanding of real-time systems, multi-threading, and platform-specific constraints, while building confidence in designing reliable and scalable applications. This experience has laid a solid foundation for future work in computer vision and interactive system development.

11. Acknowledgements

We sincerely thank our mentor for their continuous guidance, encouragement, and technical support throughout the internship. Their insights, patience, and constructive feedback played a vital role in shaping our understanding and improving the overall quality of the project. We are also grateful to the Infosys Springboard team for providing this valuable learning opportunity and a structured environment for skill development.