Infosys Springboard Virtual Internship 6.0 Completion Report

Team Details

Batch Number 5

Start date 13-OCT-25

Names:

S.no	Name
1	Prashanti Hebbar
2	Anusuya B
3	Shreyas S N
4	Siva Sankar Seetharamaiah

Internship Duration: 8 Weeks

1. Project Title

GESTURE VOLUME CONTROL – VOLUME CONTROL WITH HAND GESTURES

2. Project Objective

The objective of this project is to develop a real-time, touch-free volume control system using computer vision techniques. The system aims to allow users to control system and microphone volume through hand gestures captured via a webcam, eliminating the need for physical interaction with hardware controls.

The key objectives include:

- Detecting and tracking hand gestures in real time using a webcam.
- Extracting accurate hand landmarks using MediaPipe Hands.
- Recognizing gestures through finger counting and distance-based methods.
- Mapping gestures to system and microphone volume levels dynamically.
- Providing a smooth, responsive, and user-friendly interaction experience.

3. Project description in detail

Gesture Volume Control is a computer vision–based application designed to control system audio using natural hand gestures. The application captures live video input from a webcam and processes each frame using OpenCV. MediaPipe Hands is used to detect the presence of a hand and extract 21 hand landmarks, each represented by normalized x, y, and z coordinates. The extracted landmarks are analyzed to determine finger positions and measure the distance between specific landmarks, such as the thumb tip and index finger tip. This distance is normalized using hand scale to ensure stable behavior when the hand moves closer to or farther from the camera.

Gesture interpretation logic converts these measurements into volume control signals. The system uses the PyCaw library to interface directly with the Windows Core Audio API, enabling real-time control of both system and microphone volume. A visual interface displays the live webcam feed, detected landmarks, and current volume level, providing immediate feedback to the user. The system operates continuously, ensuring low latency, smooth volume transitions, and reliable performance in real-time scenarios.

3. Timeline Overview

Week	Activities Planned	Activities Completed
Week 1	Project initiation, understanding problem statement, studying gesture-based human–computer interaction, and finalizing project objectives and scope.	Conducted project kickoff meeting, analyzed limitations of traditional volume control methods, finalized the idea of gesture-based volume control, and identified core technologies such as OpenCV, MediaPipe, and PyCaw.
Week 2	Webcam integration and basic computer vision setup using Python and OpenCV.	Integrated webcam input using OpenCV (cv2.VideoCapture), verified real-time frame capture, handled frame flipping for mirror view, and ensured stable video feed acquisition.
Week 3	Hand detection using MediaPipe and extraction of hand landmarks.	Implemented MediaPipe Hands for palm detection and landmark extraction, successfully detected hands in real time, and obtained 21 hand landmarks with normalized x, y, and z coordinates.
Week 4	Gesture recognition logic development including finger detection and finger counting.	Developed finger detection logic by comparing fingertip and joint landmarks, implemented finger counting mechanism, handled thumb detection separately, and validated gesture accuracy across different hand orientations.
Week 5	Distance-based gesture implementation and normalization logic for stable volume control.	Implemented Euclidean distance calculation between thumb tip and index finger tip, applied hand-scale normalization to prevent volume fluctuation due to depth changes, and tested continuous volume control behavior.
Week 6	System volume control integration using Windows audio APIs.	Integrated PyCaw library to control system volume, implemented volume mapping from gesture values to system-supported range (0.0–1.0), and ensured smooth volume transitions without abrupt jumps.
Week 7	Microphone volume control and real-time user interface developmet.	Implemented microphone volume control using PyCaw, added mute and unmute functionality, developed real-time user interface to display webcam feed, hand landmarks, and volume level.
Week 8	Performance optimization, testing, documentation, and final presentation preparation.	Optimized detection confidence values, restricted detection to a single hand for accuracy, tested system under different lighting and backgrounds, finalized documentation, and prepared project presentation and demo.

5a. Key Milestones

Milestone	**Description**	**Week Achieved**
Project Initiation & Requirement Analysis	Finalized the problem statement for gesture-based volume control, defined project objectives, and selected appropriate tools and technologies such as OpenCV, MediaPipe, PyCaw, and Python.	**Week 1**
Webcam Integration & Hand Detection	Integrated real-time webcam input using OpenCV and implemented MediaPipe Hands for palm detection and extraction of 21 hand landmarks.	**Week 2–3**
Gesture Recognition Development	Designed and implemented finger detection, finger counting, and distance-based gesture logic using landmark geometry for reliable gesture interpretation.	**Week 4–5**
Volume Control Implementation	Mapped recognized gestures to system and microphone volume levels, implemented normalized distance-based control, and ensured smooth volume transitions using PyCaw.	**Week 6**
User Interface & System Integration	Developed a real-time user interface displaying webcam feed, hand landmarks, and volume status, and integrated all modules into a single working system.	**Week 7**
Testing, Optimization & Final Submission	Performed performance testing under different conditions, optimized detection accuracy and stability, completed documentation, and prepared the final project demonstration and submission.	**Week 8**

5b. Project execution details

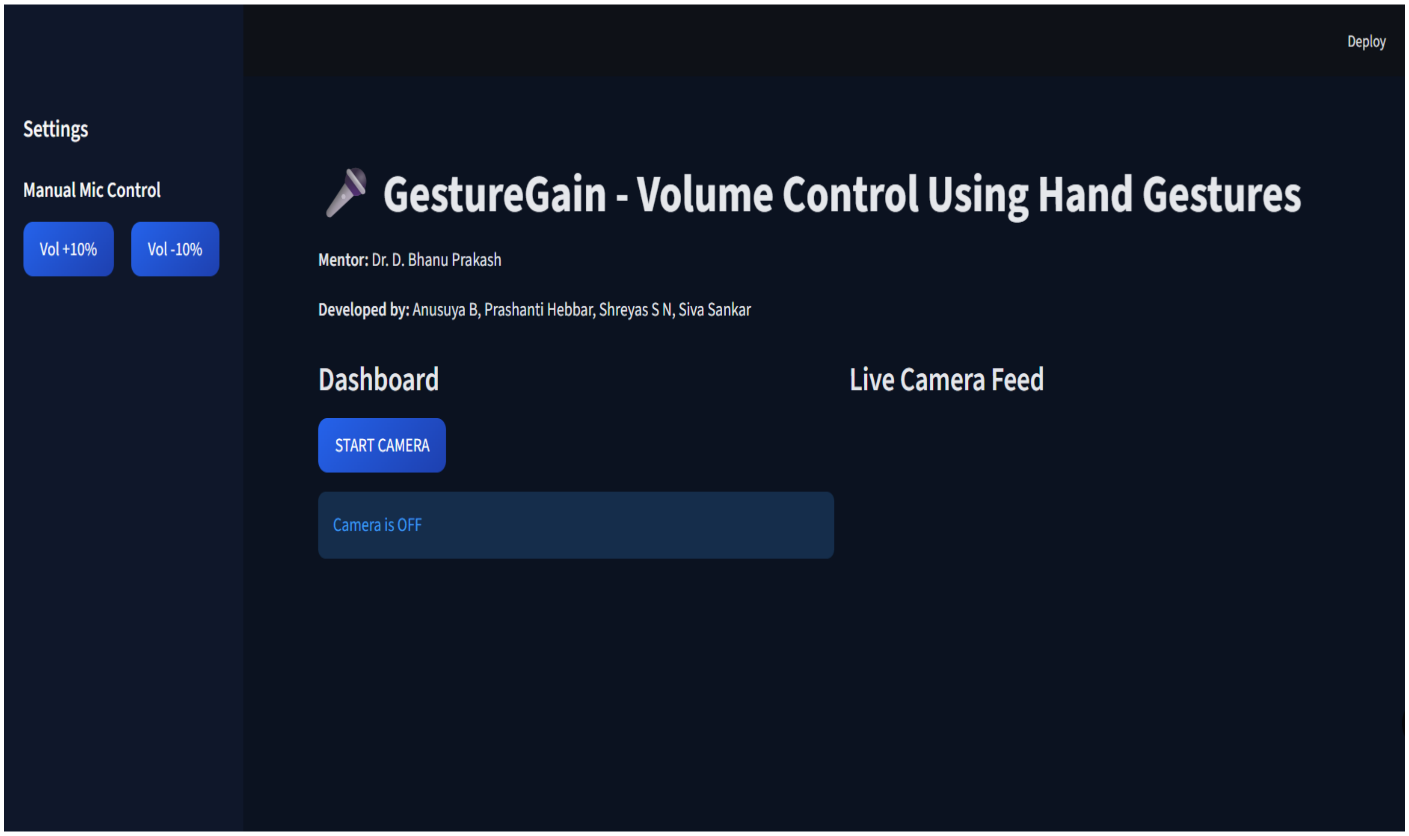
The project began with an initial planning phase, during which the problem of traditional volume control limitations was analyzed. Based on this analysis, a gesture-based, touch-free solution was finalized. The project scope, objectives, and functional requirements were clearly defined, and the core technology stack—Python, OpenCV, MediaPipe Hands, PyCaw, and Streamlit—was selected to support real-time gesture recognition and system-level audio control. In the next phase, webcam integration was implemented using OpenCV to capture live video frames. Frame acquisition was tested for stability and real-time performance. Additional preprocessing steps such as frame flipping were applied to provide a mirror view, making gesture interaction intuitive for users. This established a reliable computer vision pipeline for subsequent processing.

Following this, MediaPipe Hands was integrated to perform palm detection and hand landmark extraction. The system successfully detected the presence of a hand and extracted 21 normalized hand landmarks in real time. These landmarks provided detailed positional and depth information required for accurate gesture analysis, forming the foundation of the gesture recognition logic. Once landmark extraction was stable, gesture recognition logic was developed. Finger detection and finger counting mechanisms were implemented by comparing fingertip landmarks with corresponding joint landmarks. Special handling was introduced for thumb detection based on hand orientation to ensure consistent recognition for both left and right hands. This phase ensured reliable identification of gesture states in real time.

The project then progressed to implementing distance-based gesture control. The Euclidean distance between the thumb tip and index finger tip was calculated to enable continuous volume adjustment. To prevent volume fluctuations caused by hand movement toward or away from the camera, hand-scale normalization was applied using reference landmark distances. This ensured stable and smooth gesture-to-volume mapping. With gesture interpretation in place, system-level audio control was integrated using the PyCaw library. Gesture values were mapped to valid system volume ranges supported by the Windows Core Audio API. Volume transitions were carefully controlled to avoid abrupt changes, resulting in smooth and responsive audio adjustment. Subsequently, microphone volume control was implemented alongside system volume control. Using PyCaw, the default microphone device was identified and controlled programmatically. Mute and unmute functionality was added to enhance usability during online meetings and presentations. A real-time user interface was developed to display the live webcam feed, detected hand landmarks, and current volume status, providing immediate visual feedback to the user.

In the final phase, the system was optimized and thoroughly tested. Detection confidence thresholds were adjusted to improve accuracy, and processing was restricted to a single hand to enhance stability. The application was tested under different lighting conditions, backgrounds, and hand orientations to validate performance. Final documentation was prepared, and the complete system was packaged for demonstration and submission.

6. Snapshots / Screenshots



Deploy

Settings

Manual Mic Control

Vol +10% Vol -10%

GestureGain - Volume Control Using Hand Gestures

Mentor: Dr. D. Bhanu Prakash

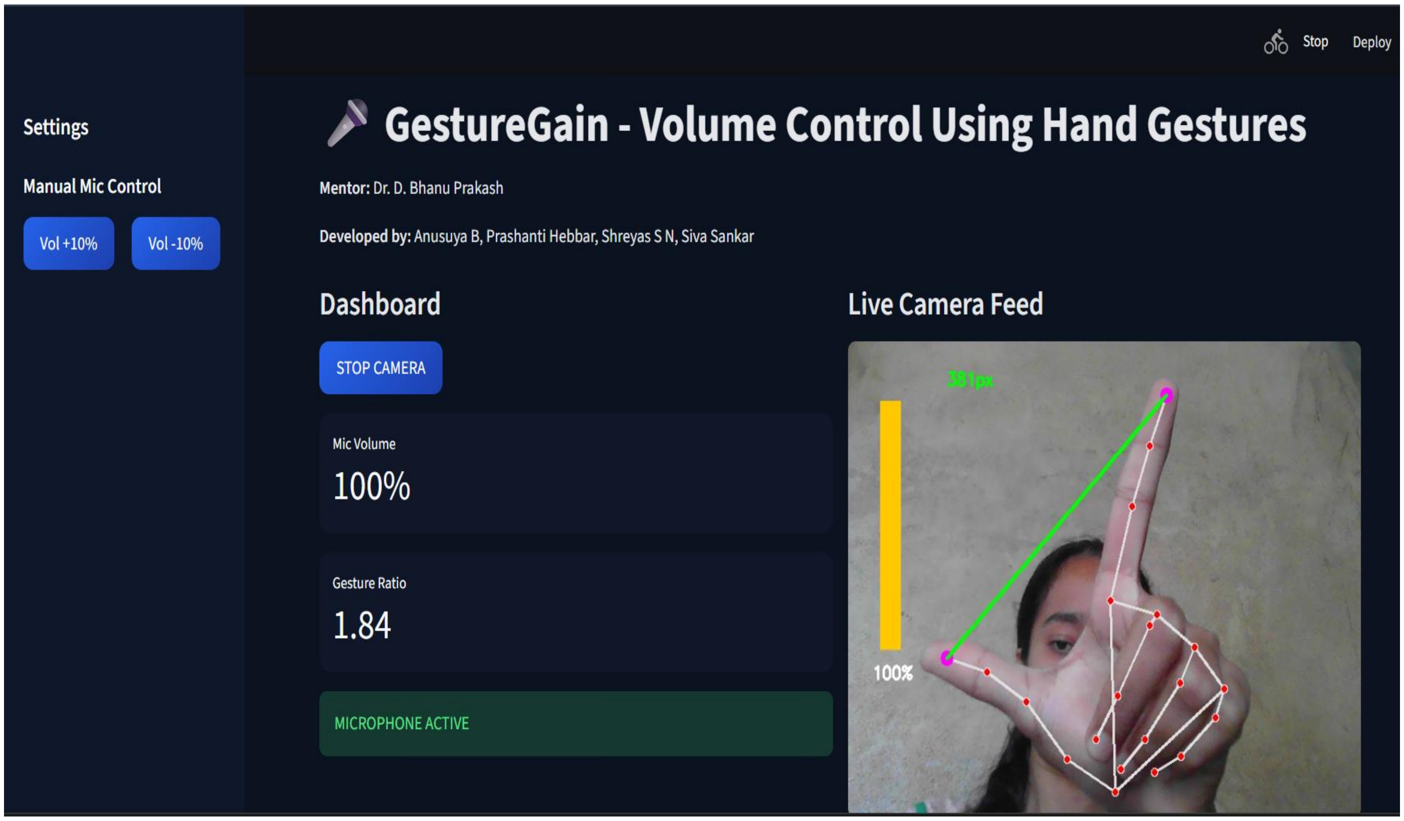
Developed by: Anusuya B, Prashanti Hebbar, Shreyas S N, Siva Sankar

Dashboard

START CAMERA

Camera is OFF

Live Camera Feed



Stop Deploy

Settings

Manual Mic Control

Vol +10% Vol -10%

GestureGain - Volume Control Using Hand Gestures

Mentor: Dr. D. Bhanu Prakash

Developed by: Anusuya B, Prashanti Hebbar, Shreyas S N, Siva Sankar

Dashboard

STOP CAMERA

Mic Volume

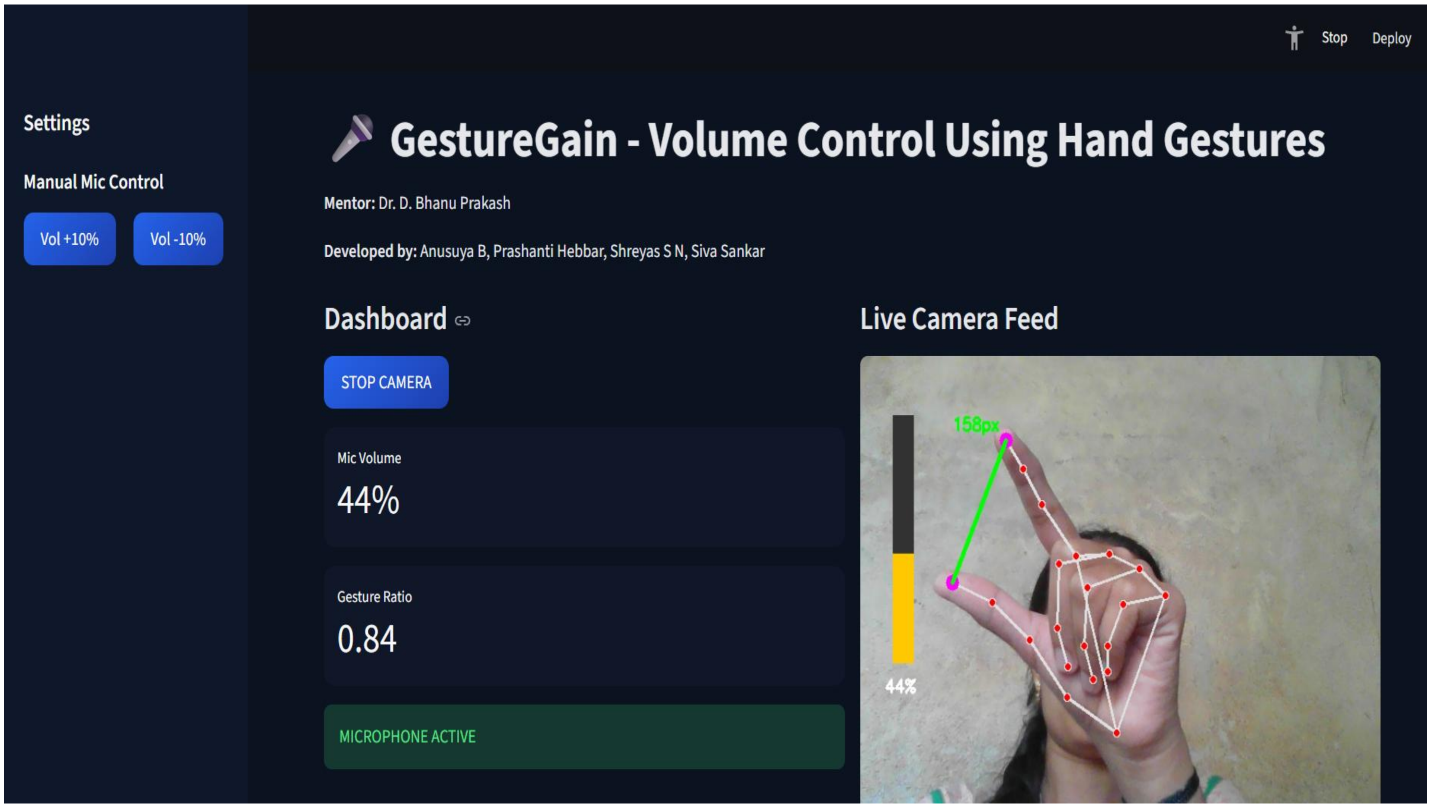
100%

Gesture Ratio

1.84

MICROPHONE ACTIVE

Live Camera Feed



Stop Deploy

Settings

Manual Mic Control

Vol +10% Vol -10%

GestureGain - Volume Control Using Hand Gestures

Mentor: Dr. D. Bhanu Prakash

Developed by: Anusuya B, Prashanti Hebbar, Shreyas S N, Siva Sankar

Dashboard

STOP CAMERA

Mic Volume

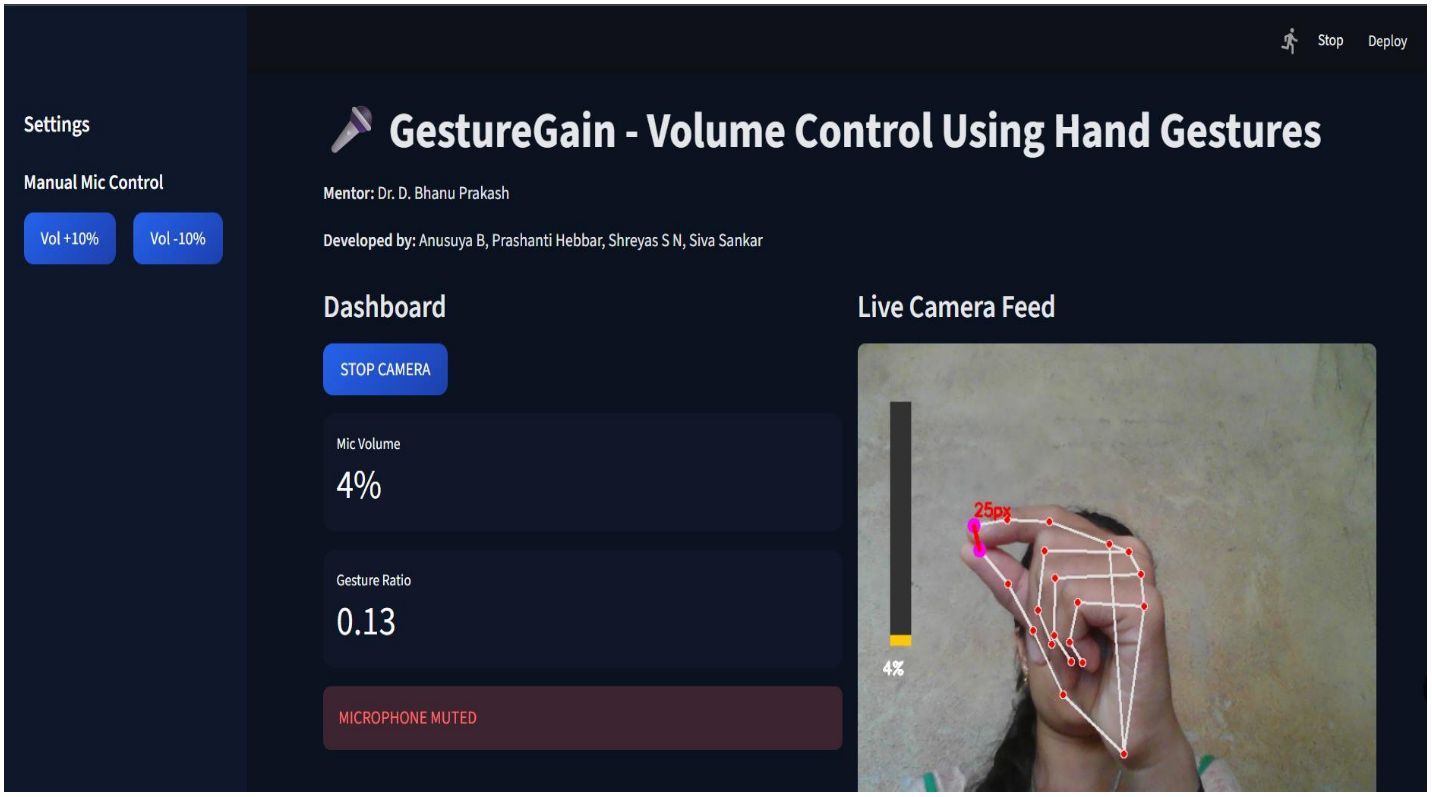
44%

Gesture Ratio

0.84

MICROPHONE ACTIVE

Live Camera Feed



Stop Deploy

Settings

Manual Mic Control

Vol +10% Vol -10%

GestureGain - Volume Control Using Hand Gestures

Mentor: Dr. D. Bhanu Prakash

Developed by: Anusuya B, Prashanti Hebbar, Shreyas S N, Siva Sankar

Dashboard

STOP CAMERA

Mic Volume

4%

Gesture Ratio

0.13

MICROPHONE MUTED

Live Camera Feed

7. Challenges Faced

During the implementation of the Gesture Volume Control – Volume Control with Hand Gestures project, several technical and practical challenges were encountered across different development stages. These challenges were carefully addressed to ensure reliable real-time performance.

Real-Time Webcam Processing Stability

Capturing continuous video frames from the webcam without delay was initially challenging. Minor latency and frame drops were observed during early testing, which affected gesture responsiveness. Ensuring stable frame acquisition while maintaining real-time processing speed required careful handling of camera initialization and frame processing logic.

Lighting and Environmental Variations

Hand detection accuracy varied significantly under different lighting conditions. Poor illumination or uneven lighting caused delayed palm detection and occasional loss of landmarks. Additionally, cluttered backgrounds sometimes interfered with reliable hand tracking, especially when background objects resembled skin tones.

Hand Landmark Detection Consistency

Maintaining consistent detection of all 21 hand landmarks during continuous hand movement was challenging. Rapid hand motion and extreme hand orientations occasionally caused temporary misdetections. Ensuring stable landmark tracking required tuning detection confidence values and restricting detection to a single hand.

Finger Detection and Gesture Interpretation

Accurate finger detection was difficult due to variations in hand size, orientation, and finger positioning. Thumb detection posed a specific challenge because of its sideways movement and dependence on left- or right-hand orientation. Separate logic was required to prevent incorrect finger counting and gesture misclassification.

Depth Sensitivity in Distance-Based Control

While implementing distance-based volume control, raw distance measurements between the thumb and index finger were affected by the hand's distance from the camera. This caused unintended volume changes when the hand moved forward or backward, even if finger spacing remained constant.

System and Microphone Volume Integration

Integrating system and microphone volume control using PyCaw introduced challenges related to Windows COM objects. Handling audio endpoint initialization and ensuring reliable communication with the Windows Core Audio API required careful error handling to prevent unexpected failures during runtime.

8. Learnings & Skills Acquired

The Gesture Volume Control project provided valuable hands-on experience in computer vision, real-time system development, and audio control integration. The key learnings and skills acquired during the internship are outlined below.

- **Computer Vision and Image Processing:** Gained a strong understanding of real-time image processing using OpenCV, including webcam integration, frame preprocessing, color space conversion, and frame transformations. Learned how to manage real-time video streams efficiently without introducing latency.

- **Hand Tracking and Gesture Recognition:** Developed practical experience in using MediaPipe Hands for palm detection and hand landmark extraction. Learned how to work with 21 normalized hand landmarks and apply geometric relationships for finger detection, finger counting, and gesture interpretation.

- **Distance-Based Gesture Control Logic:** Learned to implement Euclidean distance measurement between hand landmarks and apply scale normalization techniques to ensure stable gesture behavior irrespective of hand depth. This enhanced understanding of depth sensitivity in vision-based systems.

- **System and Microphone Audio Control:** Acquired knowledge of system-level audio control using PyCaw and the Windows Core Audio API. Gained experience working with COM objects and audio endpoints to control both system and microphone volume programmatically.

- **Real-Time Application Development:** Improved skills in designing and implementing real-time applications with continuous processing loops. Learned to manage synchronization between gesture detection, audio control, and user interface updates to ensure smooth interaction.

- **Debugging, Optimization, and Testing:** Developed strong debugging and optimization skills by addressing issues related to detection accuracy, performance stability, and smooth volume transitions. Learned to test the system under varying lighting conditions, backgrounds, and hand orientations.

- **Technical Documentation and Presentation:** Enhanced skills in technical documentation, report writing, and structured presentation of project work. Learned how to clearly explain system architecture, execution flow, challenges, and outcomes in a professional manner.

9. Testimonials from team

The Gesture Volume Control project served as a valuable collaborative learning experience for the entire team. Team members actively contributed to project discussions, development, testing, and documentation tasks. Regular communication and coordination helped ensure steady progress throughout the internship period.
The project encouraged teamwork, problem-solving, and adaptability, especially while managing real-time system challenges. The experience strengthened technical confidence and improved the team's ability to work effectively in a professional project environment while balancing academic responsibilities.

10. Conclusion

The Gesture Volume Control – Volume Control with Hand Gestures project successfully demonstrates the application of computer vision and gesture recognition techniques for real-time, touch-free system control. By enabling users to adjust system and microphone volume using natural hand gestures, the project enhances accessibility, usability, and convenience. The system achieves accurate hand detection, stable gesture interpretation, and smooth audio control using a standard webcam and widely available software libraries. This project establishes a strong foundation for future enhancements in gesture-based human–computer interaction and smart interface systems.

11. Acknowledgements

We sincerely express our gratitude to Infosys Springboard for providing the opportunity to participate in this virtual internship and gain hands-on experience in real-world project development. We would also like to thank Dr. D. Bhanu Prakash for continuous guidance, encouragement, and technical support throughout the project.
Their mentorship and feedback played a crucial role in shaping our understanding, improving our implementation, and ensuring the successful completion of this project.