System Requirements Specification (SRS) Document for Hand Gesture Control System

---

1. Introduction

1.1 Purpose

The purpose of this project is to develop a system that uses a webcam and hand gestures to control computer functionalities such as scrolling, brightness adjustment, volume control, and mouse interactions. The system leverages computer vision and hand tracking technologies.

1.2 Scope

The Gesture Control System allows users to interact with their computer using hand gestures captured through a webcam. The gestures are processed using Mediapipe's hand-tracking library, and various actions (e.g., mouse clicks, scrolling, brightness/volume control) are executed accordingly.

1.3 Definitions, Acronyms, and Abbreviations

- **Mediapipe**: A framework for building machine learning pipelines for media processing.
- **Gesture**: A movement of the hand that conveys a specific command.
- **PyAutoGUI**: A library used for controlling mouse and keyboard programmatically.
- **Brightness Control**: Adjusting the screen brightness.
- **Volume Control**: Adjusting the system sound volume.

1.4 References

- Mediapipe documentation: https://mediapipe.dev
- OpenCV documentation: https://opencv.org
- PyAutoGUI documentation: https://pyautogui.readthedocs.io
- Screen Brightness Control Library: https://pypi.org/project/screen-brightness-control/

---

2. System Overview

The system uses a webcam to capture hand movements. It processes the video feed to detect hand landmarks and recognizes specific gestures using predefined criteria. Based on these gestures, appropriate commands are sent to the operating system.

---

3. System Requirements

3.1 Functional Requirements

1. **Gesture Detection:**
   - Detect and classify hand gestures using Mediapipe.
   - Support gestures such as:
     - **V Gesture**: Moves the mouse cursor.
     - **Fist Gesture**: Enables drag-and-drop functionality.
     - **Pinch Gesture**: Adjusts system brightness or volume based on the direction.
     - **Two-Finger Closed Gesture**: Scrolls vertically or horizontally.
2. **Command Execution:**
   - Control mouse cursor movement.
   - Simulate left-click, right-click, and double-click actions.
   - Adjust system brightness using the pinch gesture.
   - Adjust system volume using the pinch gesture.
   - Scroll horizontally or vertically using gestures.
3. **Hand Tracking:**
   - Use Mediapipe to track hand landmarks in real-time.
   - Support multi-hand tracking (major and minor hands).
4. **GUI Integration:**
   - Display a live video feed with hand landmarks overlaid for debugging.
5. **User Interaction:**
   - Provide a smooth and natural experience by stabilizing cursor movement and avoiding jitter.
6. **Gesture Stabilization:**
   - Implement noise handling to avoid unintended gestures.

3.2 Non-Functional Requirements

1. **Performance:**
   - The system should process at least **30 frames per second** for smooth gesture recognition.
2. **Usability:**
   - The system should be intuitive and easy to use with minimal setup.
3. **Compatibility:**
   - Compatible with Windows operating systems.
   - Works with any webcam with a minimum resolution of **640x480**.
4. **Scalability:**
   - Should allow additional gestures or functionalities to be added in the future.
5. **Reliability:**
   - Should handle edge cases like partial hand visibility or overlapping hands gracefully.
6. **Security:**
   - The system should not store or transmit video data for privacy reasons.

---

4. System Architecture

4.1 Hardware Requirements

Component	Minimum Requirement
Processor	Intel i3 or equivalent
RAM	4 GB
Storage	1 GB free space
Webcam	Resolution: 640x480, 30 FPS
Display	Screen with adjustable brightness

4.2 Software Requirements

Component	Minimum Requirement
Operating System	Windows 10 or higher
Python Version	Python 3.8 or higher
Libraries	OpenCV, Mediapipe, PyAutoGUI, Pycaw, Screen Brightness Control, Google Protobuf

---

5. Functional Flow

1. **Capture Video Feed:**
   - Access the webcam using OpenCV.
   - Flip and preprocess the video feed.
2. **Hand Detection and Tracking:**
   - Use Mediapipe to detect hand landmarks.
   - Classify hands as **Major (right)** or **Minor (left)**.
3. **Gesture Recognition:**
   - Recognize gestures based on the relative positions of hand landmarks.
   - Map recognized gestures to predefined actions.
4. **Command Execution:**
   - Perform corresponding actions such as mouse movement, clicks, scrolling, or adjusting brightness/volume.
5. **Feedback Loop:**
   - Stabilize commands by processing gestures over multiple frames to minimize noise.

---

6. Constraints

1. Requires good lighting conditions for accurate hand detection.
2. May not work well if hands are partially obscured or out of the camera's field of view.
3. The system relies on the CPU; performance may degrade on lower-end hardware.

---

7. Use Cases

1. **Mouse Control:**
    - Use gestures to move the mouse, click, or drag files.
2. **Brightness/Volume Control:**
    - Adjust system brightness or volume using pinch gestures.
3. **Scrolling:**
    - Scroll through documents or webpages using minor hand gestures.

---

8. Future Enhancements

1. Add support for custom gestures defined by the user.
2. Support gesture-based text input or keyboard shortcuts.
3. Extend compatibility to other operating systems like macOS or Linux.
4. Integrate voice commands alongside gesture recognition.

---

9. Appendix

9.1 Dependencies

- OpenCV: Real-time video processing.
- Mediapipe: Hand tracking and gesture recognition.
- PyAutoGUI: Simulate mouse and keyboard actions.
- Screen Brightness Control: Adjust system brightness.
- Pycaw: Control system audio.

9.2 Limitations

- System performance is dependent on the webcam quality and lighting conditions.
- Gesture recognition accuracy may decrease for non-standard hand movements.