# Software Requirements Specification

for

# AirBoard

(A Virtual Keyboard System Using Computer Vision)

Prepared by:

**Adhya Sharma and Shritama Acharyee**

Department of Computer Science and Engineering
Manipal University Jaipur

Under the Guidance of:

**Mr. Aditya Sinha**

**Date:** October 31, 2025

**Version 1.0**

# Contents

# 1 Introduction

## 1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to define a detailed description of the functionality, design constraints, and performance criteria of the **AirBoard** system. This document acts as a formal agreement between the development team and stakeholders, providing a shared understanding of what the system will do. The AirBoard system aims to offer an intuitive, touch-free virtual keyboard that utilizes computer vision to detect hand gestures and translate them into key presses, promoting accessibility, innovation, and hygienic computing environments.

## 1.2 Scope

AirBoard is a standalone computer vision-based desktop application designed to replace traditional keyboard input with gesture-based control. The software uses a webcam to detect hand motion, process the input frames in real-time, and convert gestures into corresponding keystrokes. It displays an interactive virtual keyboard that highlights and responds to finger movements, enabling a contactless typing experience.

Potential use cases include:

- Accessibility for differently-abled users.
- Touchless input in healthcare or public environments.
- Integration with AR/VR interfaces or smart classrooms.

The system's modular design allows future expansion, such as adding voice input or multi-language support.

## 1.3 Definitions, Acronyms, and Abbreviations

- **OpenCV:** Open Source Computer Vision Library for real-time image and video processing.
- **MediaPipe:** Google framework for hand and gesture tracking using machine learning.
- **cvzone:** A Python package simplifying OpenCV and MediaPipe integration.
- **GUI:** Graphical User Interface.
- **ROI:** Region of Interest – the selected area of the image being analyzed.
- **FPS:** Frames Per Second – a measure of system responsiveness.

### 1.4 Overview

This SRS document includes a detailed overview of the system architecture, features, design diagrams, and performance requirements. It aims to serve as a comprehensive guide for developers, testers, and stakeholders to ensure consistent understanding and successful implementation of the AirBoard system.

# 2 Overall Description

## 2.1 Product Perspective

AirBoard is an independent application that relies on the computer's webcam as the input device. The system captures live video frames and processes them using the MediaPipe hand-tracking framework to detect hand landmarks. Each frame is analyzed to identify the position of the fingertips relative to virtual keyboard coordinates. When a gesture meets the threshold of a "key press," the corresponding key is activated and displayed on the virtual screen.

The architecture is composed of four main modules:

- **Input Module:** Captures real-time webcam video feed.
- **Processing Module:** Uses MediaPipe and OpenCV to analyze hand landmarks.
- **Keyboard Module:** Displays a virtual keyboard and manages key states.
- **Display Module:** Outputs typed text and provides user feedback.

## 2.2 Product Functions

The key functional capabilities include:

- Real-time hand detection and gesture tracking.
- Visual keyboard rendering on the screen.
- Key highlighting and selection based on finger proximity.
- Text rendering corresponding to detected key presses.
- Gesture-based controls such as space, backspace, clear, and enter.
- System calibration to adjust for lighting and distance variations.

## 2.3 User Characteristics

The expected users include:

- **General Users:** Individuals interested in touchless interaction.
- **Students and Developers:** Exploring computer vision and HCI (Human-Computer Interaction).
- **Researchers:** Investigating gesture recognition systems.
- **Accessibility Users:** Individuals with mobility limitations who require non-physical input methods.

### 2.4 Operating Environment

- Hardware: Webcam-enabled computer (minimum 720p resolution).
- Software: Windows/Linux/macOS.
- Libraries: Python 3.x, OpenCV, MediaPipe, cvzone, NumPy.
- Lighting: Minimum 250 lux illumination for accurate detection.

### 2.5 Constraints

- Requires stable lighting for accurate hand tracking.
- Dependent on webcam resolution and frame rate.
- Limited multitasking due to real-time processing.
- Cannot detect gestures reliably if background objects interfere.

### 2.6 Assumptions and Dependencies

- The user has Python installed with all necessary libraries.
- The webcam is functional and provides a continuous stream.
- The application assumes one active user in front of the camera.

# 3 System Features

## 3.1 Feature 1: Hand Detection and Tracking

**Description:** The system captures video input from the webcam and detects the hand using MediaPipe's hand landmark model. **Inputs:** Live video frames. **Outputs:** Coordinates of hand landmarks and finger tips. **Process:**

- Capture frames using OpenCV.
- Detect hands and extract 21 landmarks using MediaPipe.
- Identify fingertip and thumb positions for gesture analysis.

**Precondition:** The webcam must be active and the hand should be visible. **Postcondition:** Hand coordinates are updated for gesture interpretation.

## 3.2 Feature 2: Virtual Keyboard Interface

Displays an interactive keyboard on the screen using OpenCV rectangles. Each key has a coordinate region and responds to hover and press actions. **Functions include:**

- Key hover highlighting.
- Key press feedback.
- Dynamic text rendering for user input.

## 3.3 Feature 3: Gesture-based Key Press

Recognizes specific gestures (e.g., thumb-index proximity) as a key press. The system checks whether the fingertip position lies within a key region to register an input.

## 3.4 Feature 4: Display of Typed Text

Displays all pressed keys on a text area within the application window. This acts as a virtual typing surface for real-time visual feedback.

## 3.5 Feature 5: System Controls

Special gestures are assigned for:

- **Clear Text:** Both hands open gesture.
- **Space:** Two-finger spread gesture.
- **Backspace:** Quick flick motion.

# 4 System Design Diagrams

## 4.1 Use Case Diagram

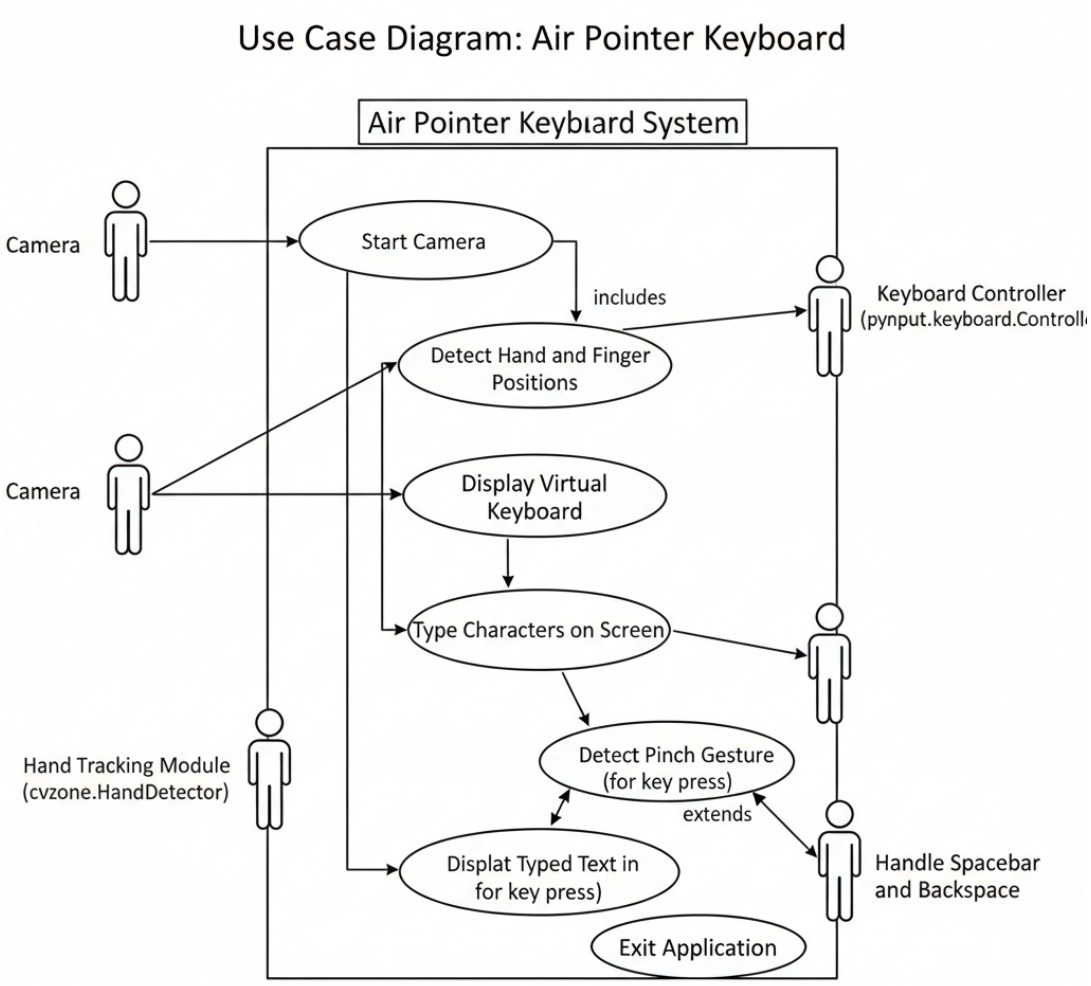


Figure 1: Use Case Diagram for Air Pointer Keyboard System

## 4.2 Data Flow Diagram

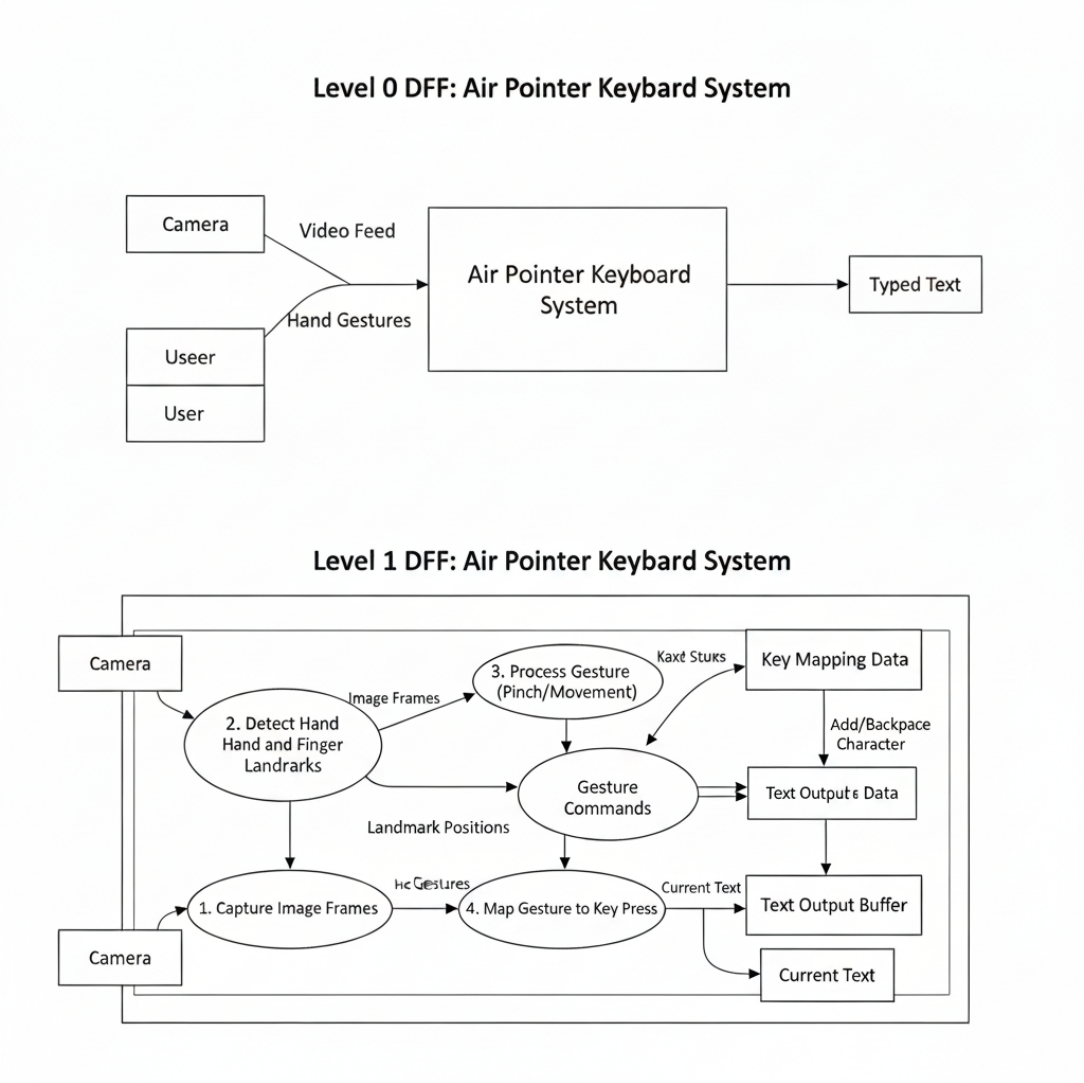


Figure 2: Data Flow Diagram for Air Pointer Keyboard System

### 4.3 Entity-Relationship Diagram

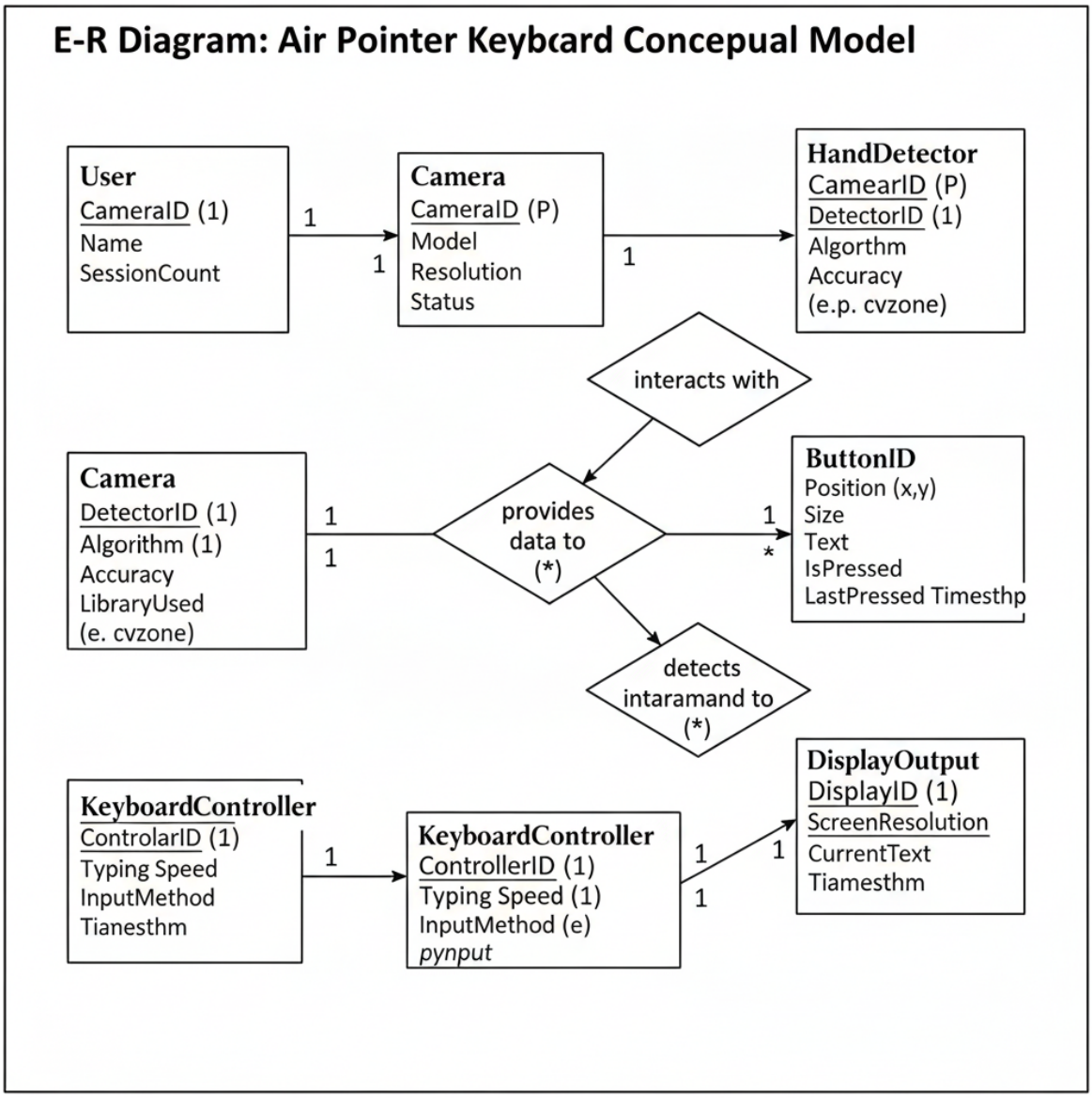


Figure 3: ER Diagram for Air Pointer Keyboard System

### 4.4 Class Diagram

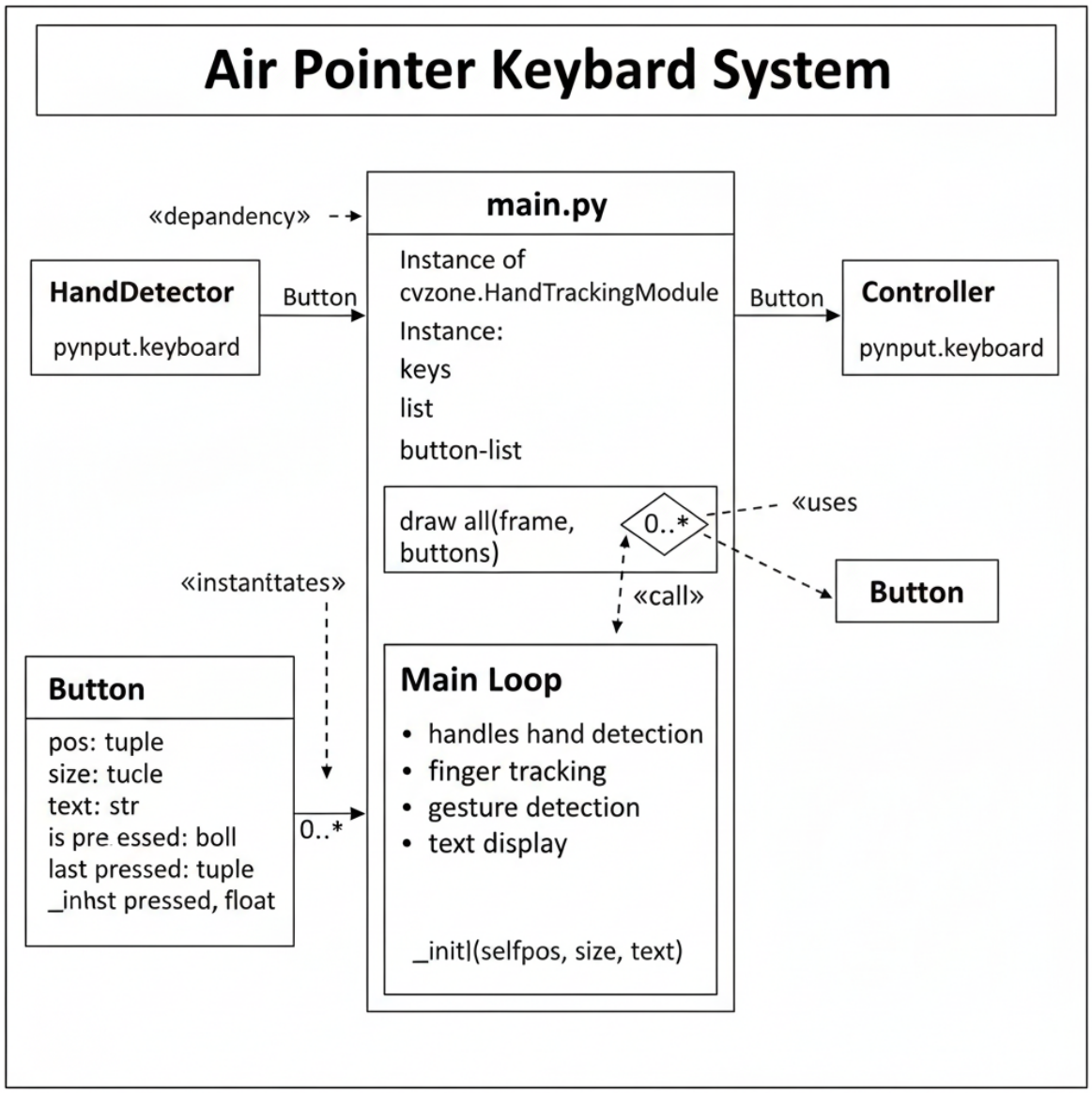


Figure 4: Class Diagram for Air Pointer Keyboard System

### 4.5 Sequence Diagram

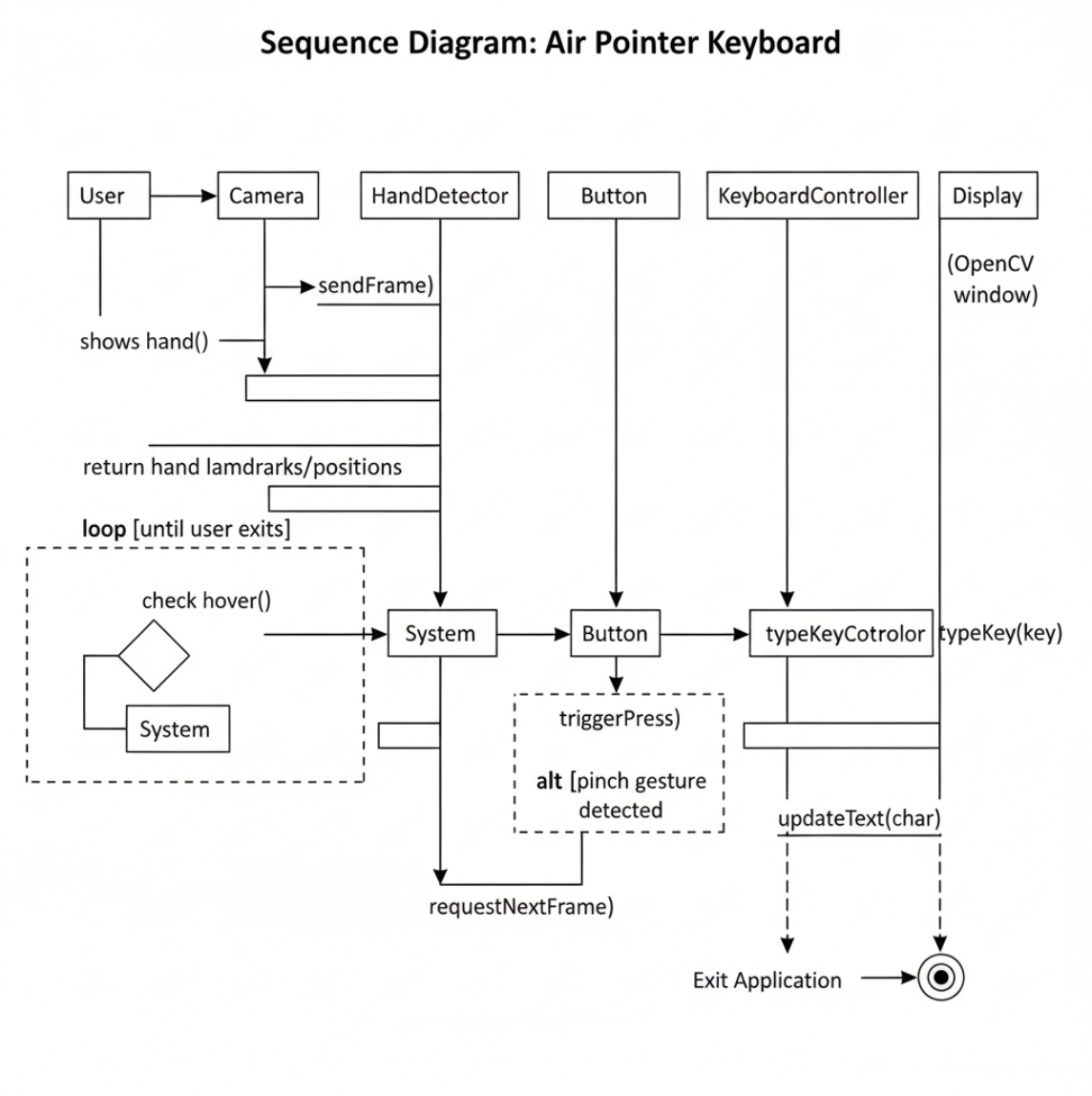


Figure 5: Sequence Diagram for Air Pointer Keyboard System

### 4.6 Activity Diagram

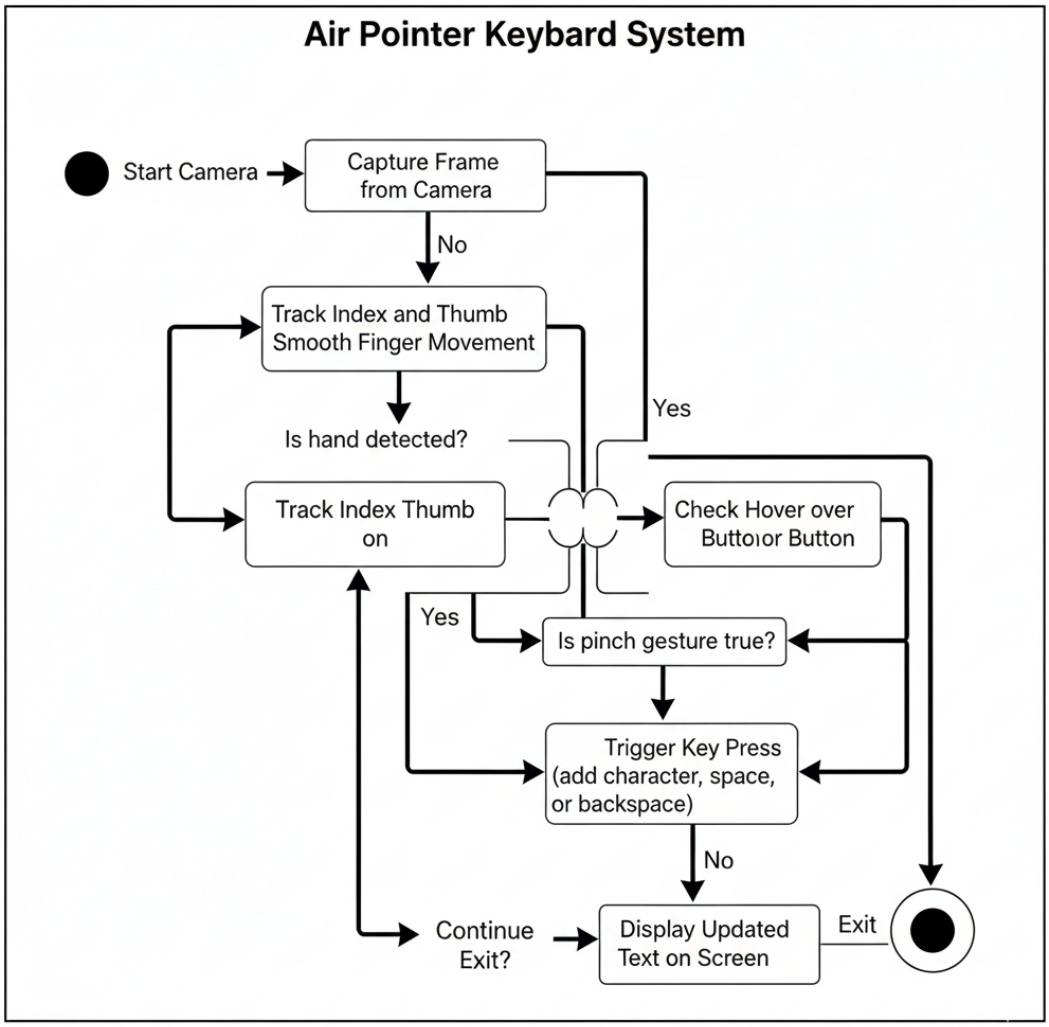


Figure 6: Activity Diagram for Air Pointer Keyboard System

### 4.7 State Chart Diagram

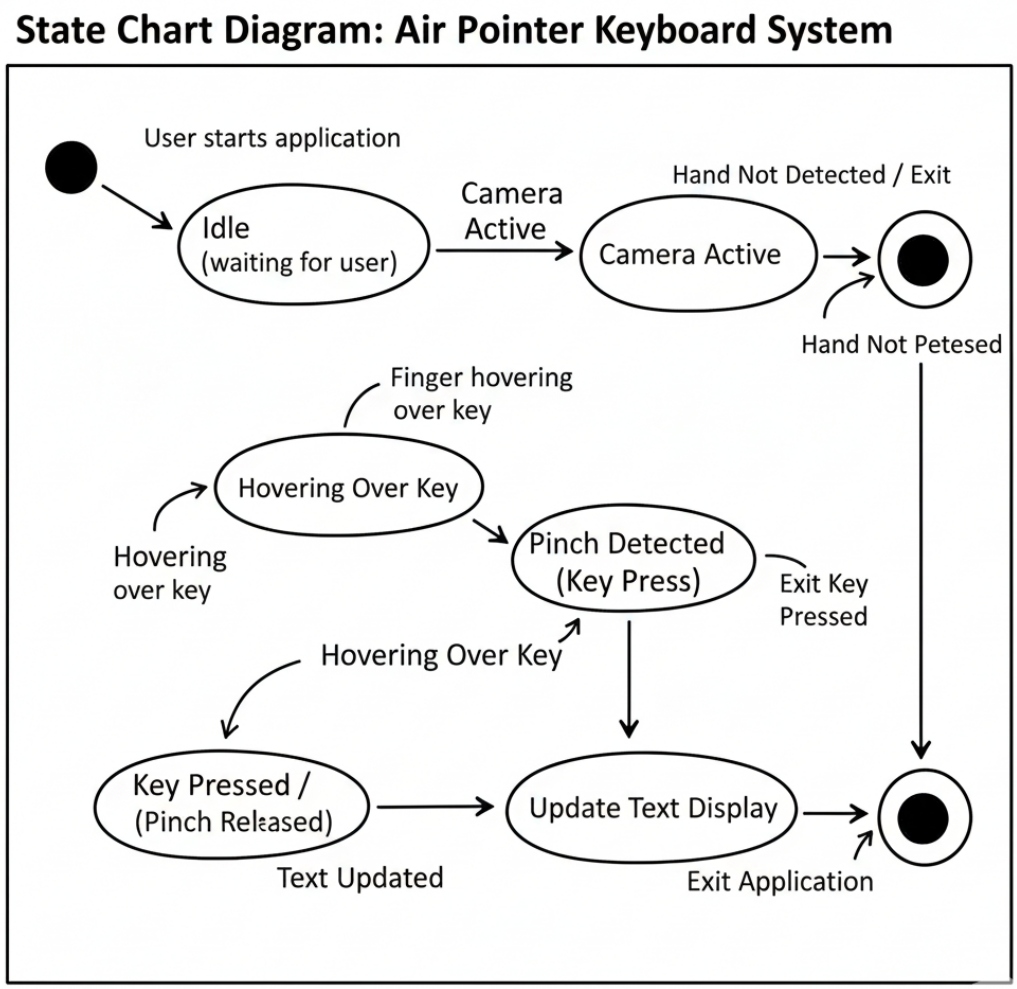


Figure 7: State Chart Diagram for Air Pointer Keyboard System

# 5 Non-Functional Requirements

## 5.1 Performance Requirements

- The system should maintain a minimum of 15 FPS for smooth operation.
- Gesture recognition accuracy must exceed 90% under optimal conditions.
- Latency between gesture and key response should be under 200 ms.

## 5.2 Security Requirements

- All processing is done locally without storing or transmitting user data.
- Webcam access must require explicit user permission.

## 5.3 Software Quality Attributes

- **Usability:** Easy to install and intuitive to use.
- **Reliability:** Should work consistently under varied lighting.
- **Portability:** Cross-platform support.
- **Maintainability:** Modular and well-documented source code.
- **Scalability:** Can be extended to support multiple gestures or users.

## 5.4 Environmental Requirements

- Adequate lighting (at least 250 lux).
- Stable camera mount for accuracy.

# 6 Conclusion

The AirBoard system successfully demonstrates the fusion of computer vision and human-computer interaction for creating a contactless input interface. It minimizes dependency on physical devices while promoting hygiene and accessibility. The modular structure and open-source libraries make it adaptable for future enhancements like predictive typing, multi-language input, or AI-based gesture learning.