**Real-time Stabilization and 3D Reconstruction of Hand Gestures and Finger Movement Traces Using LED-Equipped Gloves**



Software Requirements Specification (SRS)

**Version No. 1.0**

**Project Document Revision History**

| **VersionNumber** | **Date** | **Revision Author** | **Description of Revision** |
| --- | --- | --- | --- |
| 1.0 | 10/31/2024 | Team 5 | Original Published Version |
| 1.1 | 11/10/2024 | Team 5 | Updated Published Version |

# I. Table of Contents

[**I. Table of Contents 3**](#_heading=h.96w65dfa41xf)

[**II. GRAPHICAL NOTATIONS USED 6**](#_heading=h.t3d4luw7ffyp)

[**1. Introduction 7**](#_heading=h.x89sgshlrep9)

[1.1 Goals and Objectives 7](#_heading=h.u292beqjx1f9)

[1.2 System Statement of Scope 7](#_heading=h.lmm8yyihtlx4)

[Outputs 8](#_heading=h.v2hazx6h6uzq)

[1.2.1 General Requirements 8](#_heading=h.gvaxriyug22l)

[1.2.2 Extended Enhancement 8](#_heading=h.nyizf28z38gq)

[1.3 System Context 9](#_heading=h.w26s0lk44y77)

[1.4 Major Constraints 9](#_heading=h.f8p75eu4qfn3)

[**2.0 Usage Scenario 9**](#_heading=h.nfqq5zk45fx0)

[2.1 User Profiles 9](#_heading=h.jw21jnxi7eg0)

[2.2 User Stories 10](#_heading=h.gv6oxf6duv2p)

[**3.0 Data Model and Description 10**](#_heading=h.x8yj5seq83yg)

[3.1 Data Description 10](#_heading=h.mbgok9nq5o0m)

[3.1.1 Data Objects 10](#_heading=h.wn8alssfywd3)

[3.1.2 Relationships 11](#_heading=h.l7oyqnma0pw4)

[3.1.3 Complete Data Model 12](#_heading=h.1iczg59p039j)

[3.1.4 Data Dictionary 12](#_heading=h.u873rwose59m)

[**4.0 Functional Model and Description 13**](#_heading=h.ng7toihuxkct)

[4.1 MA-01: Mobile Application 13](#_heading=h.ohjg7ifp4ke4)

[4.1.1 Use Case Name 13](#_heading=h.wdzxfgbsy8cp)

[4.1.2 Actors 13](#_heading=h.h49z9rxm2sdc)

[4.1.3 Preconditions 14](#_heading=h.6pxkyzyjk07s)

[4.1.4 Triggers 14](#_heading=h.4he20zd08fkd)

[4.1.5 Scenario Description 14](#_heading=h.9pneq96gmq04)

[4.1.6 Post Conditions 14](#_heading=h.wobprv1jdhf4)

[4.1.7 Exceptions 15](#_heading=h.iemn8nbi99u)

[4.1 IP-01: Image Processing 15](#_heading=h.walg3i6bepom)

[4.1.1 Use Case Name 15](#_heading=h.gery5xcyg730)

[4.1.2 Actors 15](#_heading=h.90rsdoz4fbqz)

[4.1.3 Preconditions 16](#_heading=h.tbpof1xz770)

[4.1.4 Triggers 16](#_heading=h.kbawm7663xbe)

[4.1.5 Scenario Description 16](#_heading=h.z0cock2k00a)

[4.1.6 Post Conditions 17](#_heading=h.4yarq1t56vjf)

[4.1.7 Exceptions 17](#_heading=h.cwigfe46q2iz)

[4.1 ML-01: Machine Learning 17](#_heading=h.vf5rbaf3y3h8)

[4.1.1 Use Case Name 17](#_heading=h.9ap68b3gqq0c)

[4.1.2 Actors 18](#_heading=h.j3y384sys8d6)

[4.1.3 Preconditions 18](#_heading=h.dbo9xb66exs7)

[4.1.4 Triggers 18](#_heading=h.h8uuyejf1v40)

[4.1.5 Scenario Description 18](#_heading=h.dnbmabtyarl1)

[4.1.6 Post Conditions 18](#_heading=h.cdd3hmvw69v4)

[4.1.7 Exceptions 18](#_heading=h.i0h83fguaccm)

[4.1 DV-01: 3D Finger Tracking for Air Writing Recognition 19](#_heading=h.x0w9yk32qm7o)

[4.1.1 Use Case Name 19](#_heading=h.jlp8djsu2a8)

[4.1.2 Actors 19](#_heading=h.8vwb7vnjzmcs)

[4.1.3 Preconditions 19](#_heading=h.4mqcvg1xp3l)

[4.1.4 Triggers 19](#_heading=h.2qrn9evnogj)

[4.1.5 Scenario Description 19](#_heading=h.jfwq8xioy86i)

[4.1.6 Post Conditions 19](#_heading=h.aky0xsuqpc4c)

[4.1.7 Exceptions 19](#_heading=h.190otaxzp4fe)

[4.1 AA-01: AR/VR 20](#_heading=h.hpns8gwtqml7)

[4.1.1 Use Case Name 20](#_heading=h.93mbpw2hvhzx)

[4.1.2 Actors 20](#_heading=h.115r2pgpkvm7)

[4.1.3 Preconditions 20](#_heading=h.ndgus1tjheg5)

[4.1.4 Triggers 21](#_heading=h.8bmbsxet1xb8)

[4.1.5 Scenario Description 21](#_heading=h.ynarvyv550cy)

[4.1.6 Post Conditions 21](#_heading=h.pqt09yee6m3j)

[4.1.7 Exceptions 21](#_heading=h.qxqwiibwhuvl)

[4.2 Software Interface Description 22](#_heading=h.rq0avlfuorok)

[4.2.1 External machine interfaces 22](#_heading=h.fojsdqfyy2gi)

[4.2.2 External system interfaces 22](#_heading=h.kdyz9n9pzz10)

[4.2.3 Human interface 23](#_heading=h.y9rcjs4blxif)

[4.3 Use Case Diagrams 23](#_heading=h.o7iv9l1hb76o)

[4.3.1 Mobile Application 24](#_heading=h.sr6cur72o9li)

[4.3.2 Image Processing 24](#_heading=h.ma3016gxy50s)

[4.3.3 Machine Learning 25](#_heading=h.mfkmky5t5boz)

[4.3.4 Finger Tracking for Air Writing Recognition 25](#_heading=h.4wo4ms8v7l6l)

[4.3.5 AR/VR 26](#_heading=h.x6tha2fvr821)

[4.4 Sequence Diagrams 26](#_heading=h.t7e4mi9nrp7p)

[4.5 Communication Diagrams 26](#_heading=h.gjdgxs)

[4.5.1 Gesture Tracking, Gesture Smoothing and 3D Visualization 26](#_heading=h.nch9hzk1bvrm)

[**5.0 Behavioral Model and Description 27**](#_heading=h.tby1x9kfbr3f)

[5.1 Events, States and Activities - Subsystem Flow Diagrams 27](#_heading=h.d8d08h5525yy)

[5.1.1 Events 27](#_heading=h.a1hq4d2p1dik)

[5.1.2 States 28](#_heading=h.ab5oj9j4zc9p)

[5.2 State Transition Diagrams 28](#_heading=h.4pjxlneb75bh)

[5.3 Activity Diagram 28](#_heading=h.f96yfp4k6kgs)

[**6.0 Restrictions, Limitations and Constraints 28**](#_heading=h.izjm2kr2txv0)

[**7.0 Validation Criteria 29**](#_heading=h.bg4d2g6haidl)

[7.1 Testing Methodology 29](#_heading=h.s6r4ck4n3ir8)

[7.1 Classes of Tests 29](#_heading=h.dhaxf56ugaab)

[7.2 Expected Software Response 30](#_heading=h.lh90zgo0xyc1)

[7.3 Performance bounds 30](#_heading=h.47lfc4o6m1xh)

[7.3.1 Latency 31](#_heading=h.f9rqqdz8fkga)

[7.3.2 Frame Rate 31](#_heading=h.elfduli7oym1)

[7.3.3 Gesture Smoothing and Stabilization 31](#_heading=h.96d1v9t9whn)

[7.3.4 Environmental Adaptation Response 31](#_heading=h.uxrcupkqmaps)

[7.3.5 Battery and Resource Efficiency 31](#_heading=h.oby80fshpxcv)

[7.3.6 Data Storage and Transfer 32](#_heading=h.47fk9jtxl9tz)

[**8.0 Appendices 32**](#_heading=h.vezgtjp2m95p)

[8.1 System traceability matrix 32](#_heading=h.gljk5kp8ukjj)

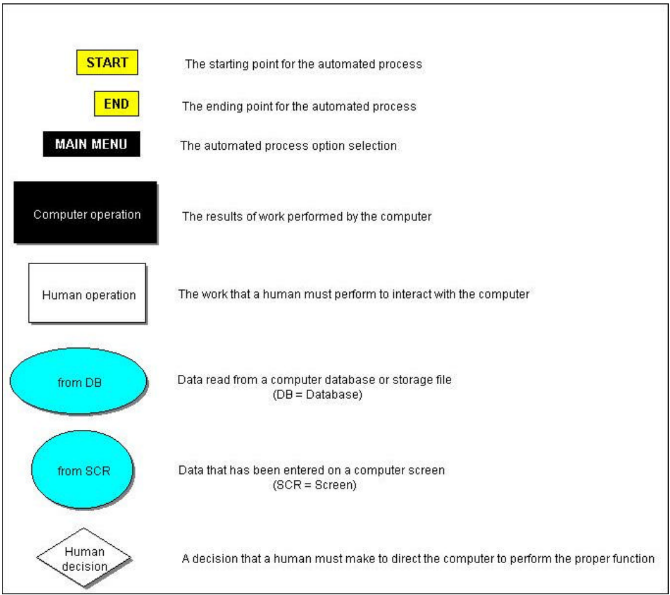
[8.2 Product Strategies 33](#_heading=h.mzvuma7okhm4)

[8.3 Analysis metrics to be used 34](#_heading=h.i2vy8st5yill)

[8.4 Supplementary information 34](#_heading=h.nhbhrka3jll)

# II. GRAPHICAL NOTATIONS USED

The following diagram defines the graphical notation used to document the business rules for automation of the Inspection to Compliance Process.



# 1. Introduction

This section outlines the requirements document, detailing all data, functional, and behavioral requirements needed for the Adaptive HCI software to meet project objectives effectively.

## 1.1 Goals and Objectives

The primary goal of the **Adaptive HCI** project is to develop a mobile application that provides stable, accurate, and real-time gesture tracking for users, specifically targeting patients with Parkinson's disease who experience tremors. This project aims to improve accessibility and ease of interaction in VR/AR and other interactive platforms by minimizing hand instability in gesture-based systems. Key objectives are:

* **To create an intuitive and accessible gesture-based interface** that stabilizes and smooths hand movements, particularly for virtual writing.
* **To develop a reliable gesture recognition system** that supports users in VR/AR environments and other interfaces requiring precise control.
* **To provide a robust system for real-time gesture processing** that minimizes the impact of tremors and enhances hand gesture accuracy.

## 1.2 System Statement of Scope

The **Adaptive HCI** software is a mobile application designed to facilitate real-time gesture tracking and smoothing, specifically for individuals experiencing hand instability, such as patients with Parkinson’s disease. The application enables users to interact seamlessly in VR/AR and other gesture-based environments by stabilizing hand movements and minimizing tremors.

**Major Inputs**

* **LED-Equipped Glove Data**: Captures and transmits hand movements, including LED position (x, y, z coordinates) in real-time, allowing precise gesture tracking.
* **Video Stream from Device Camera**: Provides a live feed to detect LED positions and trace hand gestures in 3D space using advanced image processing techniques.
* **User Commands and Settings**: Allows users to start/stop tracking, adjust gesture smoothing levels, and switch between VR/AR modes.

**Processing Functionality**

* **Gesture Tracking and Detection**: Analyzes LED positions to recognize hand gestures and map them accurately in real time.
* **Real-Time Smoothing and Stabilization**: Applies machine learning algorithms (CNN-LSTM) to reduce tremors and smooth gesture paths, creating stable, continuous traces.
* **Adaptive Lighting Adjustment**: Modifies tracking parameters based on ambient light to ensure consistent LED detection and accurate tracking across various lighting conditions.
* **3D Visualization for VR/AR**: Renders stabilized hand gestures in a 3D space, allowing users to view their gestures in an immersive, interactive environment.

#### **Outputs**

* **Real-Time Gesture Path Display**: Displays the user’s smoothed gesture path on the smartphone screen or connected VR/AR device, allowing immediate visual feedback.
* **Session Data for Further Analysis**: Stores processed gesture data for future reference or further analysis, providing valuable insights for personalized model adaptation.
* **VR/AR Interactive Experience**: Offers an immersive 3D visualization of gesture paths within a virtual environment, enabling users to interact with virtual objects using stabilized gestures.

The Adaptive HCI system aims to deliver a smooth, responsive, and adaptive gesture-based interface, enhancing user experience and accessibility for individuals with hand instability.

### 1.2.1 General Requirements

The **Adaptive HCI** system includes several key features and modules designed to enhance gesture tracking for users with hand instability:

* **Gesture Tracking with LED Gloves**: Integration of LED-equipped gloves that emit a steady, trackable light for accurate hand gesture capture.
* **Real-Time Stabilization**: A processing module that reduces hand tremors in real time using Kalman filters, bicubic interpolation, and thresholding.
* **Machine Learning-Driven Gesture Smoothing**: A hybrid CNN-LSTM model processes gesture data to predict and correct unstable movement, creating smooth and accurate 3D gesture traces.
* **VR/AR Integration**: Capability for gesture output to be visualized and interacted with in virtual and augmented reality environments.
* **User Interface**: A user-friendly mobile application that enables users to start/stop gesture tracking, adjust settings, and monitor feedback from the gesture recognition system.

### 1.2.2 Extended Enhancement

* **Gesture Customization**: Future versions could include customization options allowing users to personalize gesture sensitivity and smoothing intensity.
* **Smart Home Integration**: Extended development may incorporate smart home device control, further expanding functionality beyond VR/AR environments.

## 1.3 System Context

The **Adaptive HCI** system will be accessible through a mobile app and primarily run on smartphones equipped with rolling shutter cameras. Since this project involves real-time tracking, **low-latency processing** and **multi-user scalability** are critical. Additionally, the project is designed with future expansion in mind, allowing integration with different platforms (e.g., VR headsets, IoT devices) for users with Parkinson's disease and potentially other mobility impairments.

## 1.4 Major Constraints

* **Time Constraints**: This project has a planned completion within an academic semester, necessitating tight project milestones and efficient time allocation.
* **Hardware Limitations**: The application must support rolling shutter cameras on various smartphones, but certain image processing techniques might be hardware-intensive and require optimizations for lower-end devices.
* **Privacy**: The system must ensure that all data is processed locally or securely stored, protecting user privacy and filtering out non-relevant background information.
* **Accuracy and Responsiveness**: The system must balance smoothing with responsiveness, maintaining user control without excessive delay or unintended gesture modifications.

# 2.0 Usage Scenario

This section outlines usage scenarios for the Adaptive HCI software, including user profiles, key use cases, and special requirements impacting software use.

## 2.1 User Profiles

There are two potential user profiles for the **Adaptive HCI** mobile application:

1. **Local/Guest User (Default)**:
   * This is the default mode, allowing users to access all application features without needing an account.
   * The application includes a **pre-trained neural network model** that provides immediate gesture smoothing, and it continues to fine-tune its performance based on locally captured user data.
   * All data remains on the device, with ongoing model training adapting to the user’s gestures over time, enhancing the personalized smoothing effect.
2. **Registered User (Potential Future Enhancement)**:
   * In a future version, users may optionally create an account, allowing their personalized neural network model settings to be linked to their account.
   * **Account Benefits**: For registered users, the customized preferences or weights of the neural network—adapted through local use—would be linked to the account. This enables easy transfer of the personalized model to a new device, ensuring a consistent user experience across devices.

## 2.2 User Stories

This section lists all user stories defining the use cases for the Adaptive HCI software, as described by the users:

* As a user, I want to start and stop finger tracking easily on the mobile application.
* As a user, I want to set up the camera and LED gloves for gesture tracking.
* As a user, I want to see my virtual writing smoothed by the machine learning model.
* As a user, I want to view my 3D hand movements in VR/AR.
* As a user, I want to monitor real-time hand tremors and calibration data in VR/AR.
* As a user, I want to start and stop real-time gesture recording.
* As a user, I want to see my virtual writing rendered in 3D for accurate feedback.

2.3 Special usage considerations

To fully utilize the software, users must have the LED-equipped glove, an Android smartphone, and a compatible VR/AR headset. These components are essential to ensure compatibility, precise gesture tracking, and immersive VR/AR integration.

# 3.0 Data Model and Description

This section outlines the Adaptive HCI information domain, including data descriptions, key objects, relationships, a data model, and a data dictionary.

## 3.1 Data Description

This section describes the key data objects managed and manipulated by the Adaptive HCI software, detailing their roles within the system and how they interact to support functionality.

### 3.1.1 Data Objects

This section includes key data objects and classes for the system, such as User Data for session details and preferences, Gesture Tracking Data for real-time gesture capture, Visualization Data for VR/AR display, and Performance Metrics for tracking system performance. These classes collectively enable accurate and customized gesture tracking and visualization.

**User Data**

* **Attributes:** 
  + **userID**: String
  + **sessionStartTime**: DateTime
  + **sessionEndTime**: DateTime
  + **neuralNetworkWeights**: Array<Float>

**Gesture Tracking Data**

* **Attributes:**
  + **gestureID**: String
  + **timestamp**: DateTime
  + **positionX:** Float
  + **positionY:** Float
  + **positionZ:** Float
  + **PWMRate:** Integer

**Visualization Data**

* **Attributes:**
  + **path3DData:** Array<Float>
  + **deviceType:** String
  + **renderingSettings:** String

**Performance Metrics**

* **Attributes:**
  + **frameRate:** Float
  + **processingTime:** Float
  + **accuracy**: Float

### 3.1.2 Relationships

**User Data** ↔ **Gesture Tracking Data**:

* Each user’s session is linked to gesture data collected during that session, allowing for personalized neural network adjustments.

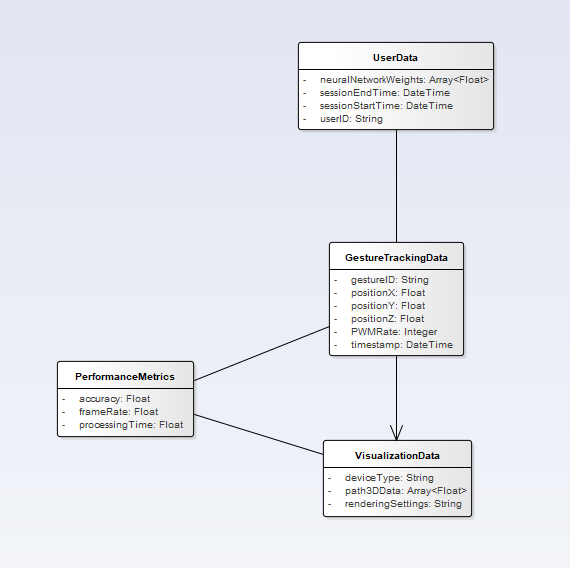
**Gesture Tracking Data** ↔ **Visualization Data**:

* Gesture tracking data flows into the 3D path visualization module for real-time display, either on the smartphone or VR set.

**Performance Metrics** ↔ **Gesture Tracking Data and Visualization Data**:

* Performance metrics are tied to each session’s gesture tracking and visualization data, enabling analysis of app responsiveness under various conditions.

### 3.1.3 Complete Data Model



### 3.1.4 Data Dictionary

**User Data**

* **userID**: A unique string identifier assigned to each user upon account creation, ensuring system-wide uniqueness.
* **sessionStartTime**: Timestamp (DateTime) marking the beginning of a user session, used to log interaction duration.
* **sessionEndTime**: Timestamp (DateTime) marking the end of a user session.
* **neuralNetworkWeights**: An array of floating-point values representing personalized neural network weights for individual users, used for gesture customization and consistency across devices.

**Gesture Tracking Data**

* **gestureID**: A unique string identifier for each captured gesture.
* **timestamp**: A DateTime record for each frame in the gesture, stored as a 4D array (x, y, z, time) to maintain continuity of gesture flow.
* **positionX, positionY, positionZ**: Float values representing the 3D coordinates of the LED glove in space, tracking the path of each gesture.
* **PWM Signature**: Pulse Width Modulation rate (Integer) associated with the LED glove, used to uniquely identify and differentiate it from other light sources in the environment.
* **Environmental Conditions**: Describes ambient lighting conditions as a string to adjust gesture tracking parameters accordingly. These conditions help calibrate tracking algorithms for optimal performance.

**Visualization Data**

* **path3DData**: An array of floating-point values representing the smoothed 3D coordinates of the gesture path, processed for VR/AR visualization.
* **deviceType**: A string that specifies the type of device used to ensure compatibility and appropriate display adjustments.
* **renderingSettings**: A string that includes visual quality and responsiveness settings based on device capabilities, adjusting for frame rate, resolution, or latency as required.

**Performance Metrics**

* **frameRate**: A floating-point value representing the rate of video capture during gesture tracking, measured in frames per second (fps).
* **processingTime**: A float representing the time taken to process each frame for gesture recognition, smoothing, and visualization.
* **accuracy**: A float indicating the percentage of correct predictions made by the machine learning model for gesture recognition.

# 4.0 Functional Model and Description

This section provides an overview of primary software functions for the system, supported by the Use Case, sequence, and communication diagrams to illustrate system interactions and workflow.

## 4.1 MA-01: Mobile Application

This section provides a detailed description of the Mobile Application use case, specifically focusing on starting and stopping finger tracking within the app.

### 4.1.1 Use Case Name

Mobile Application (MA-01).

### 4.1.2 Actors

**User**

* Interacts with the mobile application to initiate and stop gesture tracking by performing hand movements.

### 4.1.3 Preconditions

1. **Mobile Application**
   1. The mobile application must be installed and running on the user's device, with all necessary permissions granted for accessing the camera and Bluetooth functionality.
2. **Camera**
   1. The rolling shutter camera must be properly connected to the mobile app, and it should be powered on and ready to capture hand movements.
3. **LED Gloves**
   1. The LED gloves must be charged, powered on, and paired with the mobile app, ensuring that all sensors are functioning correctly for gesture tracking.

### 4.1.4 Triggers

This function is utilized when the user opens the mobile application and selects the "Start Tracking" option to begin gesture tracking, or "Stop Tracking" to terminate the session.

### 4.1.5 Scenario Description

1. The user opens the mobile app and clicks the "Start Tracking" button.
2. The mobile app triggers the rolling shutter camera and LED gloves to start tracking the user's hand and finger movements.
3. The camera begins capturing real-time frames of the user’s hand gestures and sends the data to the app.
4. The LED gloves' sensors detect the precise finger positions and movements, sending this data to the app via Bluetooth or a wired connection.
5. The mobile app processes the captured data and combines the input from the camera and gloves to create a real-time simulation of the hand movements.
6. The simulated gestures are displayed on the mobile app interface, reflecting the user’s hand and finger movements in real-time.
7. The user can stop tracking at any time by clicking the "Stop Tracking" button, which saves the captured gesture data for further use or analysis.

### 4.1.6 Post Conditions

**Successful Gesture Simulation**

* After the use case is completed, either on a mobile phone or AR/VR device, the system has successfully captured and processed the user’s hand gestures using the LED gloves. The gestures are simulated in real-time, and the data is saved or ready for further analysis.

### 4.1.7 Exceptions

**Server Exception**

1. The mobile app attempts to communicate with the server for data processing but encounters an issue.
2. If a Server Processing Error occurs, the app notifies the user that the server cannot process the request and suggests trying again later.
3. If No Internet Connection is detected, the app informs the user of the lack of connectivity and prompts them to reconnect.
4. If the Connection is Interrupted during data transmission, the app attempts to reconnect. If unsuccessful, it notifies the user and pauses further operations.

**Camera Exception**

1. The app attempts to initialize the camera, but a hardware or connection issue arises.
2. In the case of a Camera Hardware Error, the app informs the user that the camera has malfunctioned and suggests checking or reconnecting the device.
3. If the Camera is Not Detected, the app notifies the user that the camera is not connected and advises them to verify the connection.
4. If the Camera is Not Compatible, the app informs the user that the camera does not meet the system requirements and suggests using a compatible device.

**Glove Exception**

1. The app attempts to pair or use the LED gloves, but an error occurs.
2. If a Glove Hardware Error is detected, the app notifies the user of a malfunction and recommends checking or restarting the gloves.
3. If the Glove is Not Detected, the app alerts the user to reconnect the gloves and ensures proper pairing.
4. If the Glove is Not Compatible, the app informs the user that the gloves are incompatible and suggests using a different, supported pair of gloves.

## 4.1 IP-01: Image Processing

This section provides a detailed description of the Image Processing function within the project, focusing on real-time stabilization and 3D reconstruction of hand gestures using LED-equipped gloves.

### 4.1.1 Use Case Name

Image Processing (IP-01).

### 4.1.2 Actors

**Mobile Application Module**

* Responsible for interfacing with the user, initiating gesture tracking, and managing camera access.

**Image Processing Module**

* Processes captured frames, extracts key features, and performs pre-processing for gesture tracking.

**Machine Learning Module**

* Analyzes the processed gesture data for pattern recognition and further refinement of gesture accuracy.

**Rolling Shutter Camera**

* Captures frames in real-time, providing raw input for the Image Processing Module.

**LED-Equipped Gloves**

* Provides visible tracking points (LEDs) used by the camera to track hand movements in 3D space.

**User**

* Interacts with the mobile application to initiate and stop gesture tracking by performing hand movements.

### 4.1.3 Preconditions

**Mobile Application**

* The mobile application must be installed and configured, with access to the Camera to capture real-time frames.

**LED Visibility Given Ambient Lighting**

* The LED-equipped gloves must have their lights visible and clearly detectable, ensuring they are distinguishable from ambient lighting conditions for accurate gesture tracking.

### 4.1.4 Triggers

**Gesture Tracking Initiation**

* The Image Processing Module use case is initiated by the Mobile Application Module after the user clicks the "Record Gestures" button.

**Real-Time Frame Data Capture**

* The Mobile Application Module continuously feeds real-time camera frame data to the Image Processing Module.

### 4.1.5 Scenario Description

1. The user opens the mobile app and clicks the "Record Gestures" button.
2. The Mobile App Module triggers the rolling shutter camera to capture real-time frames of the user's hand movements.
3. The camera begins capturing frames using the Camera2 API and feeds them to the Image Processing Module.
4. The captured frames are processed in real-time:
   1. The frames undergo pre-processing, which includes sharpening, denoising, edge detection, and thresholding to enhance image quality.
   2. Key features such as 2D position (X, Y), depth (Z), and time (T) are extracted from each frame using OpenCV.
5. The Image Processing Module compiles the extracted data into a 4D array (X, Y, Z, T) representing the LED's position and movement over time.
6. The gesture path is smoothed using Kalman filters, moving average filters, and spline interpolation to refine the output data.
7. The final smoothed data is output as a 4D NumPy array, which is passed to the Machine Learning Module for further analysis and recognition.

### 4.1.6 Post Conditions

**Successful Gesture Data Output**

* The Image Processing Module successfully processes the incoming frame data and outputs a 4D NumPy array containing the position (X, Y, Z) and time (T) values for the LED's path in 3D space, ready for further analysis by the Machine Learning Module.

### 4.1.7 Exceptions

**Hardware Exception (Invalid Input Data)**

1. The camera fails to capture frames due to a hardware malfunction or poor ambient lighting.
2. The Mobile App Module detects the issue and displays an error message to the user.
3. The user can retry or exit the recording session.

**Software/Data Exception (Invalid Data Processing)**

1. The input data is available, but the Image Processing Module fails to detect the LED or encounters an error during pre-processing or feature extraction.
2. The system logs the error and prompts the user to adjust the environment (e.g., improve lighting or visibility of the LED).
3. The user can retry or exit the session.

## 4.1 ML-01: Machine Learning

This section provides a detailed description of the Machine Learning function within the project, focusing on smoothing and reconstructing 3D virtual writing gestures in real-time.

### 4.1.1 Use Case Name

Machine Learning (ML-01).

### 4.1.2 Actors

**User**

* The end-user who interacts with the application.

**System**

* The entity that processes inputs and returns the outputs to the user.

### 4.1.3 Preconditions

1. The user must wear the gloves designed for tracking virtual writing gestures.
2. The user must access the mobile app to enable data processing.
3. The user must perform virtual writing gestures in front of the camera.
4. The image processor must successfully convert the virtual writing gestures into 3D trajectories for further analysis.

### 4.1.4 Triggers

The use case initiates when the system receives raw gesture data from the image processing phase, triggering the start of data standardization, noise reduction, and feature extraction for machine learning processing.

### 4.1.5 Scenario Description

1. Raw virtual writing gesture data is received from the image processing phase.
2. The system adjusts the data to standardize the length of the input signal.
3. The system checks the data for noise and reduces it to create a smoother trajectory.
4. Key features, including kinematic, geometric, and frequency-based attributes, are extracted from the trajectory.
5. The extracted features are normalized for consistent input into the machine learning model.
6. The processed data is input into a neural network model, which identifies spatial and temporal patterns within the virtual writing.

### 4.1.6 Post Conditions

1. The virtual writing gesture is smoothed and accurately rendered in real-time as a 3D trajectory within the mobile app.
2. The smoothed virtual writing is displayed in 3D space, available for interaction and review on the mobile app interface.
3. The 3D virtual writing output is processed and available for viewing in AR/VR, allowing the user to engage with the writing in an immersive environment.

### 4.1.7 Exceptions

1. The system receives raw virtual writing gesture data from the image processing phase.
2. The system attempts to standardize the input signal length.
3. The system detects that the data quality is insufficient.
4. The system prompts the user to repeat the virtual writing gesture to capture improved data quality.
5. Once improved data is captured, the system resumes the main flow at the trajectory smoothing step.

## 4.1 DV-01: 3D Finger Tracking for Air Writing Recognition

This section provides a detailed description of the 3D Finger Tracking for Air Writing Recognition use case within the project, focusing on real-time tracking and interpretation of finger movements for character recognition.

### 4.1.1 Use Case Name

3D Finger Tracking for Air Writing Recognition (DV-01).

### 4.1.2 Actors

**User**

* Moves their finger in front of the camera to write a set of characters

**Mobile App**

* Captures and tracks finger movements, reconstructs them as 3D models, and interprets them as writing.

**OpenGL**

* Visualizes the finger paths in 3D, rendering them on the screen in real-time.

### 4.1.3 Preconditions

1. The user must be positioned correctly in front of the mobile app, with fingers within the camera’s field of view as well as within a certain distance.
2. The app must be running during this period.

### 4.1.4 Triggers

1. The use case is initiated when the user begins writing in the air in front of the mobile app.
2. The mobile app captures and processes the finger movements in real-time, reconstructing the path for character recognition.

### 4.1.5 Scenario Description

* Describe the flow of events needed to accomplish the use case

### 4.1.6 Post Conditions

1. The 3D finger path is visualized on the screen in real-time.
2. The recognized characters are sent to the system for further interaction.

### 4.1.7 Exceptions

**Invalid Finger Movement Data**

* If the captured finger movement data is incomplete or incorrect, an error message is displayed, and no rendering occurs.

**Unrecognized Character**

* If the app does not recognize the finger path as a valid character, no input is sent to the text system, and a notification is displayed.

**Invalid positioning of finger**

* If the hand/finger is not positioned properly in front of the camera, the app cannot read the information properly, and an error message is displayed

## 4.1 AA-01: AR/VR

This section provides a detailed description of the AR/VR use case within the project, focusing on real-time interaction within a virtual environment using LED gloves and a mobile phone’s camera.

### 4.1.1 Use Case Name

VR/AR (AA-01).

### 4.1.2 Actors

**User**

Uses LED gloves and the mobile phone's camera to track hand movements and interact with virtual objects in the AR/VR environment.

### 4.1.3 Preconditions

**Mobile Application**

* The mobile application must be installed and running on the user's device, with all necessary permissions granted for accessing the camera and Bluetooth functionality.

**Camera**

* The rolling shutter camera must be properly connected to the mobile app, and it should be powered on and ready to capture hand movements.

**LED Gloves**

* The LED gloves must be charged, powered on, and paired with the mobile app, ensuring that all sensors are functioning correctly for gesture tracking.

**AR/VR Glasses**

* The AR/VR glasses must be properly connected and calibrated to ensure they provide the user with an immersive virtual environment where gesture tracking and interactions can take place smoothly.

### 4.1.4 Triggers

**Camera Initialization**

* The camera is activated, and the system checks for proper connection and calibration, ensuring it is ready to capture hand and finger movements accurately.

**LED Glove Initialization**

* The LED gloves are powered on and paired with the app, with the system confirming that all sensors are working correctly and are ready for use in gesture tracking.

**Smart Phone Initialization**

* The smartphone’s system is initialized, ensuring all required components (camera, Bluetooth, or wired connections) are active and ready to support the AR/VR interaction.

### 4.1.5 Scenario Description

1. The user puts on the LED gloves and opens the AR/VR system, accessing the graphical user interface (GUI).
2. The user initiates the interaction by clicking the "Start Interaction" button within the AR/VR system.
3. The system activates the mobile phone's camera and LED gloves, beginning real-time tracking of the user's hand and finger movements.
4. The camera captures hand gestures while the LED gloves detect finger movements, sending this data to the AR/VR system.
5. The AR/VR system processes the input, allowing the user to interact with virtual objects, buttons, and icons in the GUI through gestures.
6. The system displays the real-time interaction within the AR/VR environment, reflecting the user’s movements precisely.
7. The user can end the interaction at any time by selecting the "Stop Interaction" button, which saves the interaction data for future use or analysis.

### 4.1.6 Post Conditions

**Successful Human-Computer Interaction**

The user interacts with the virtual environment using gestures, manipulating objects, buttons, and icons.

### 4.1.7 Exceptions

**Server Exception**

1. The user attempts to interact with the AR/VR system, but the server fails to process the request.
2. In the case of a Server Processing Error, the system notifies the user and attempts to retry the request or suggests an alternative action.
3. If there is No Internet Connection, the system alerts the user of the connectivity issue and pauses further interaction until the connection is restored.
4. If the Connection is Interrupted during use, the system will notify the user and attempt to reestablish the connection, pausing the interaction if necessary.

**Hardware Exception**

1. The user initiates the interaction, but a hardware issue occurs with the camera or LED gloves.
2. If there is a Hardware Error with the gloves or camera, the system will notify the user to check the device and retry the connection.
3. If the hardware is Not Detected, the system will prompt the user to ensure all components are properly connected and powered on.
4. If the hardware is Not Compatible, the system will inform the user that the connected devices do not meet the requirements for proper interaction, suggesting compatible alternatives.

## 4.2 Software Interface Description

This section describes the software interfaces of the **Adaptive HCI** mobile application, covering external machine interfaces, external system interfaces, and the human interface.

### 4.2.1 External machine interfaces

The **Adaptive HCI** application interfaces with external devices for gesture visualization and tracking. Key machine interfaces include:

* **Smartphone/VR Set Integration**: The app is designed to run on smartphones and VR headsets to display real-time gesture tracking and smoothing. The system must interface seamlessly with these devices to enable smooth, low-latency visualization.
* **LED-Equipped Gloves**: The application is designed to recognize and track the LED lights on specially equipped gloves. This connection involves interpreting light sources based on unique PWM (Pulse Width Modulation) signatures, ensuring the application isolates and tracks the correct source in real-time.

### 4.2.2 External system interfaces

The **Adaptive HCI** application requires minimal external system integration. The use of an external cloud database is optional and will be explored as a potential enhancement if time permits. Some external interfaces support additional functionalities:

* **Cloud Storage (For Registered Users)**: For users who create accounts, the app can interface with a cloud storage service (e.g., Firebase) to save and retrieve neural network weights and preferences. This enables users to transfer their customized gesture model across devices, retaining their unique smoothing adjustments and neural network adaptation.

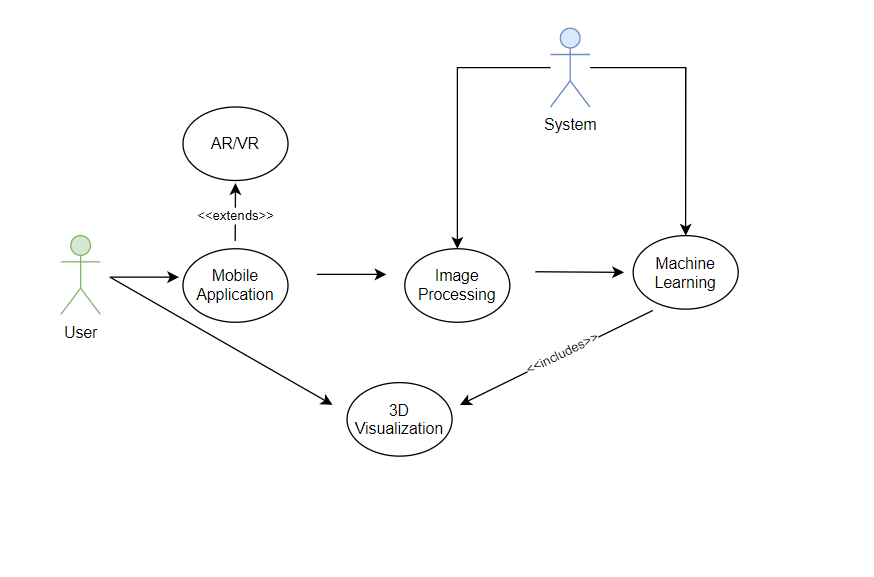
### 4.2.3 Human interface

The **Adaptive HCI** application provides an intuitive interface designed for real-time gesture tracking and visualization. Key human interface components include:

* **Home Screen**:
  + **Start Tracking**: Begins gesture tracking, capturing video feed and processing gesture smoothing in real-time.
  + **Settings**: Allows adjustments for gesture smoothing intensity, light filtering, and visualization mode.
  + **VR/AR Mode**: Connects the app to a VR headset or AR-capable display, enabling immersive visualization.
  + **Help**: Offers guidance on app usage and troubleshooting.
* **Tracking Screen**:
  + **Gesture Path Visualization**: Displays the real-time 3D path of gestures, continuously updated as the user moves.
  + **Pause/Stop Tracking**: Enables users to pause or end the tracking session as needed.
  + **Save Session**: Saves the current gesture data for later analysis or review.
* **Settings Screen**:
  + **Smoothing Intensity**: Allows users to control the amount of smoothing applied to gestures.
  + **Environment Adaptation**: Users can select between nominal and boundary lighting settings for optimal tracking.
  + **Neural Network Training Toggle**: For registered users, enables or disables continued adaptation of the neural network based on session data.
* **VR/AR Visualization Screen**:
  + **VR Mode Activation**: Connects the app to a VR headset, providing immersive 3D gesture visualization.
  + **AR Display**: Renders gestures in augmented reality using the phone screen, overlaying gestures on the real-world environment.

## 4.3 Use CaseDiagram

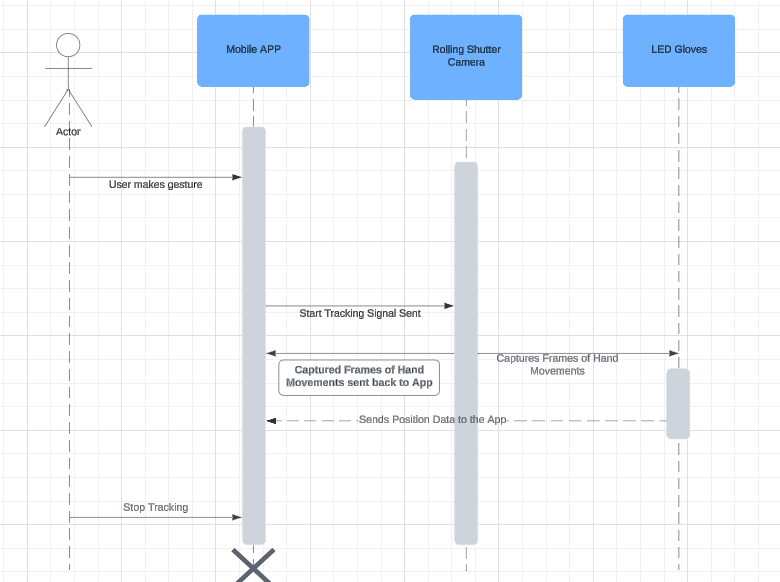
The use case diagram for **Adaptive HCI** outlines the system's control flow, covering key functionalities like the mobile application, image processing, machine learning, and 3D finger tracking for air writing recognition (3D Visualization.

****

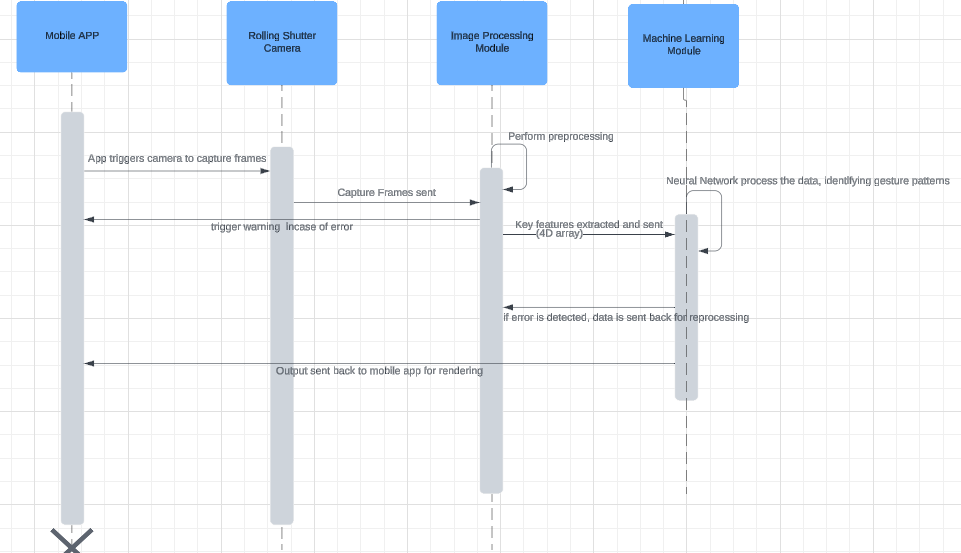
## 4.4 Sequence Diagrams

Used to model the class interactions needed for the use cases.

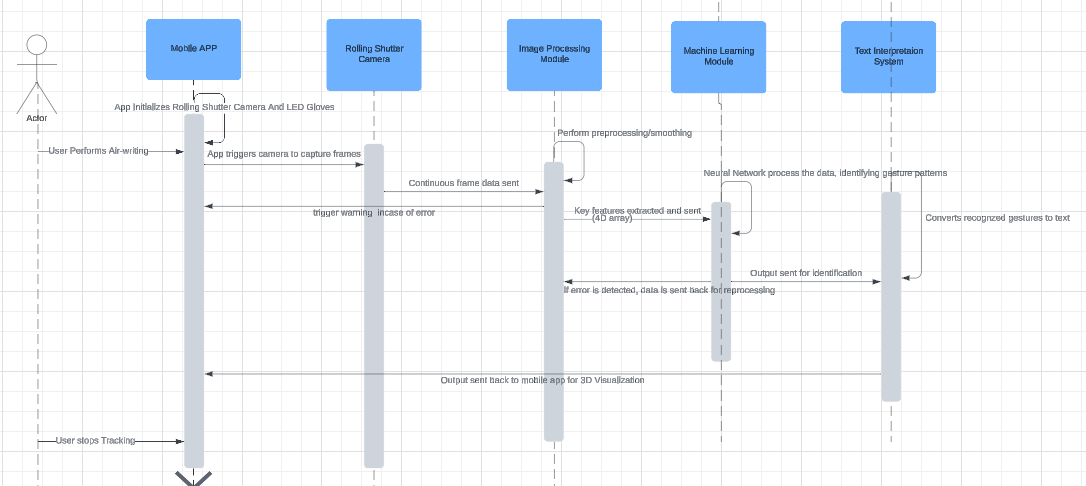
### 4.4.1 Mobile Application



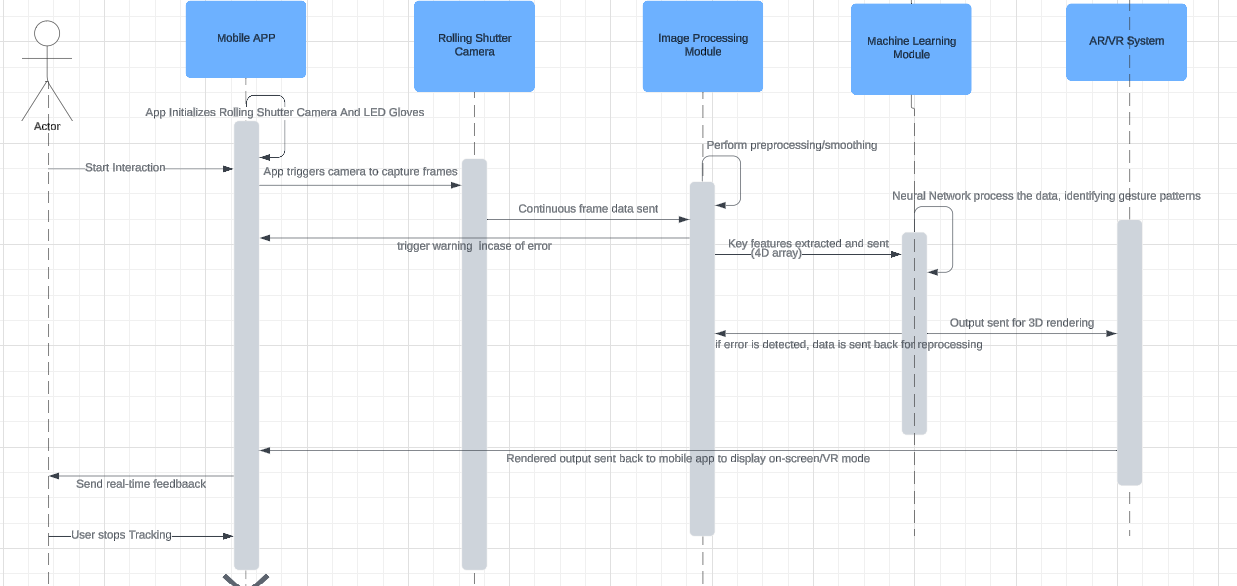
### 4.4.2 and 4.4.3 Image processing and Machine Learning



### 4.4.4 3D Finger Tracking for Air-Writing Recognition



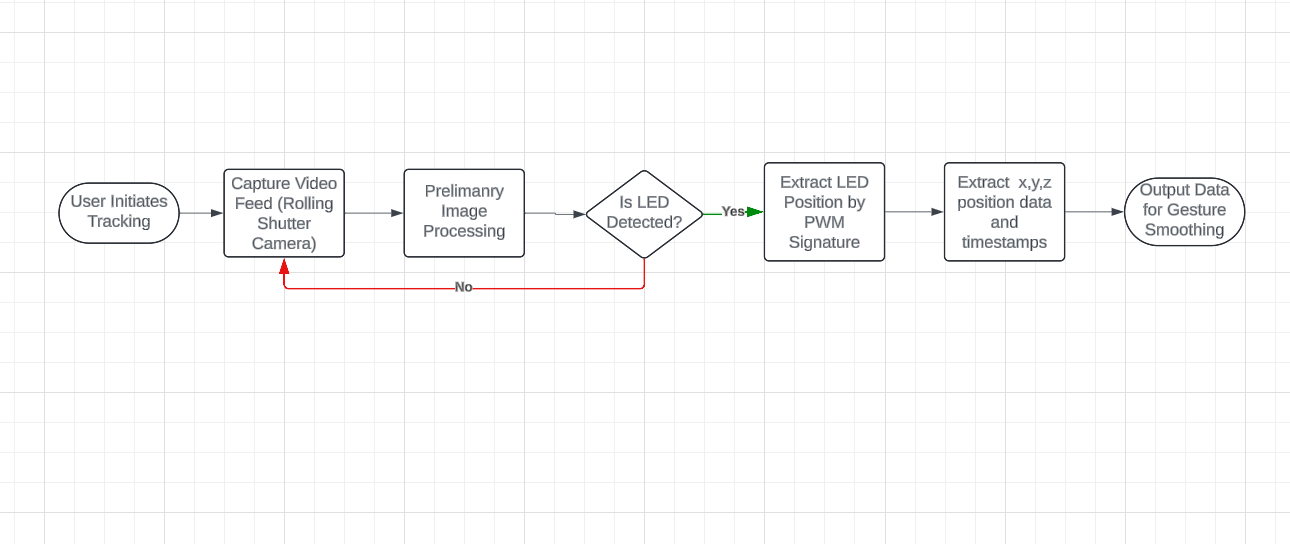
### 4.4.5 AR/VR



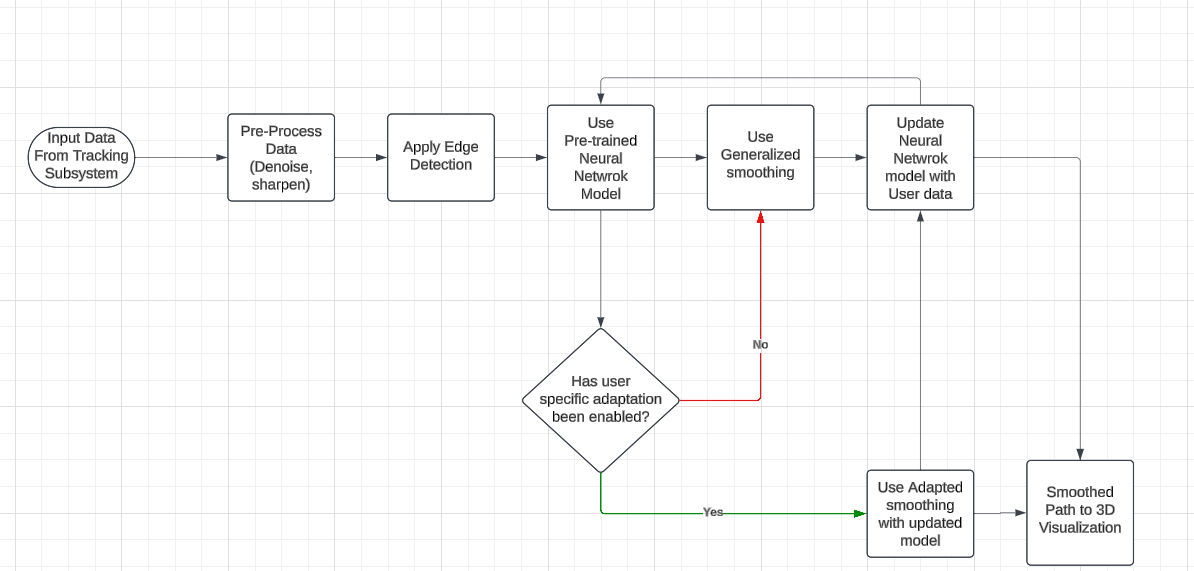
## 4.5 Communication Diagrams

The communication diagrams for **Adaptive HCI** illustrate the message-passing structure that enables system functions like gesture tracking, gesture smoothing, and 3D visualization.

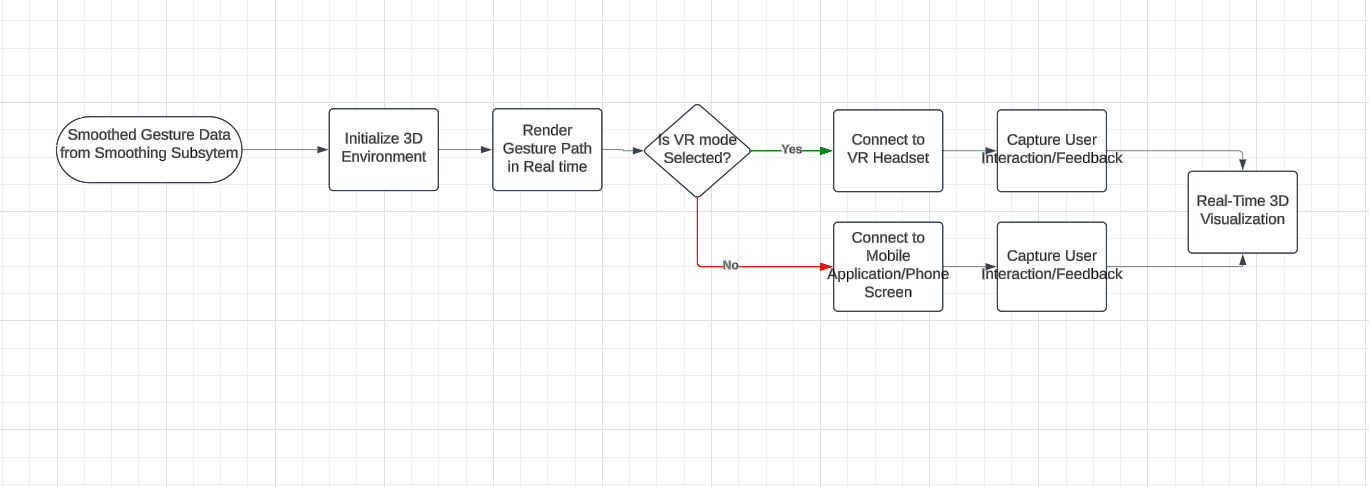
### 4.5.1 Gesture Tracking



### 4.5.2 Gesture Smoothing



### 4.5.3 3D Visualization



# 5.0 Behavioral Model and Description

This section describes the behavior of the **Adaptive HCI** software, including events, states, and transitions that shape its responsiveness, adaptability, and user experience.

## 5.1 Events, States and Activities - Subsystem Flow Diagrams

### 5.1.1 Events

Events are interactions or conditions that trigger behavioral changes within the system. The primary events include:

* **Start Tracking**: Initiates video capture, gesture recognition, and smoothing processes.
* **Stop Tracking**: Ends the current tracking session, saving any recorded gesture data if specified.
* **Pause Tracking**: Temporarily halts tracking while maintaining session state, allowing resumption without restarting.
* **Switch to VR/AR Mode**: Transitions the visualization output to a connected VR headset or AR display.
* **Environment Change Detection**: Detects ambient lighting changes, adjusting tracking and filtering parameters based on nominal or boundary conditions.
* **User Account Login**: Allows registered users to load personalized neural network settings, enabling continuity across devices.
* **Settings Adjustment**: Triggers recalibration of parameters such as smoothing intensity or lighting adaptation, applied immediately during tracking.

### 5.1.2 States

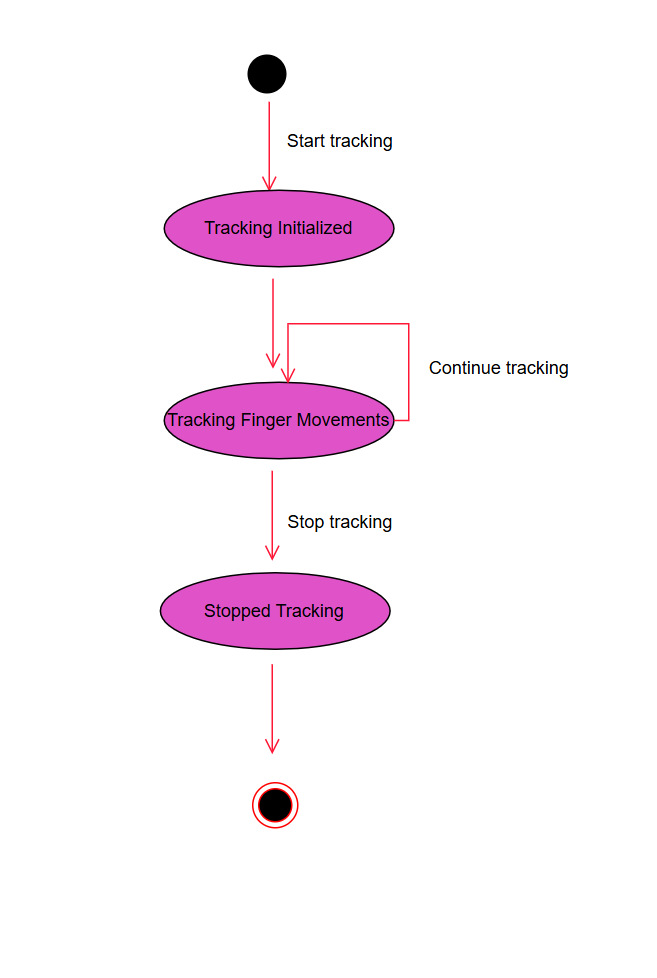
The following states represent distinct modes of behavior in response to system events:

* **Idle**: Default state when the app is inactive, awaiting user input to start tracking.
* **Tracking Active**: State during real-time video capture and gesture tracking, maintaining active updates to the display.
* **Paused**: The system maintains the current session data, suspending real-time tracking until resumed.
* **VR/AR Mode**: Active state when gesture visualization is displayed in VR or AR mode, adjusting display parameters for immersive experience.
* **Environment Adaptation Active**: The system is actively filtering and adjusting tracking settings based on detected ambient lighting conditions.
* **Settings Adjustment Mode**: Temporarily enables user-defined configuration changes (e.g., smoothing, lighting) before resuming the current state.
* **Adaptive Learning Active**: For registered users, the neural network fine-tunes in real time based on user-specific gesture data.

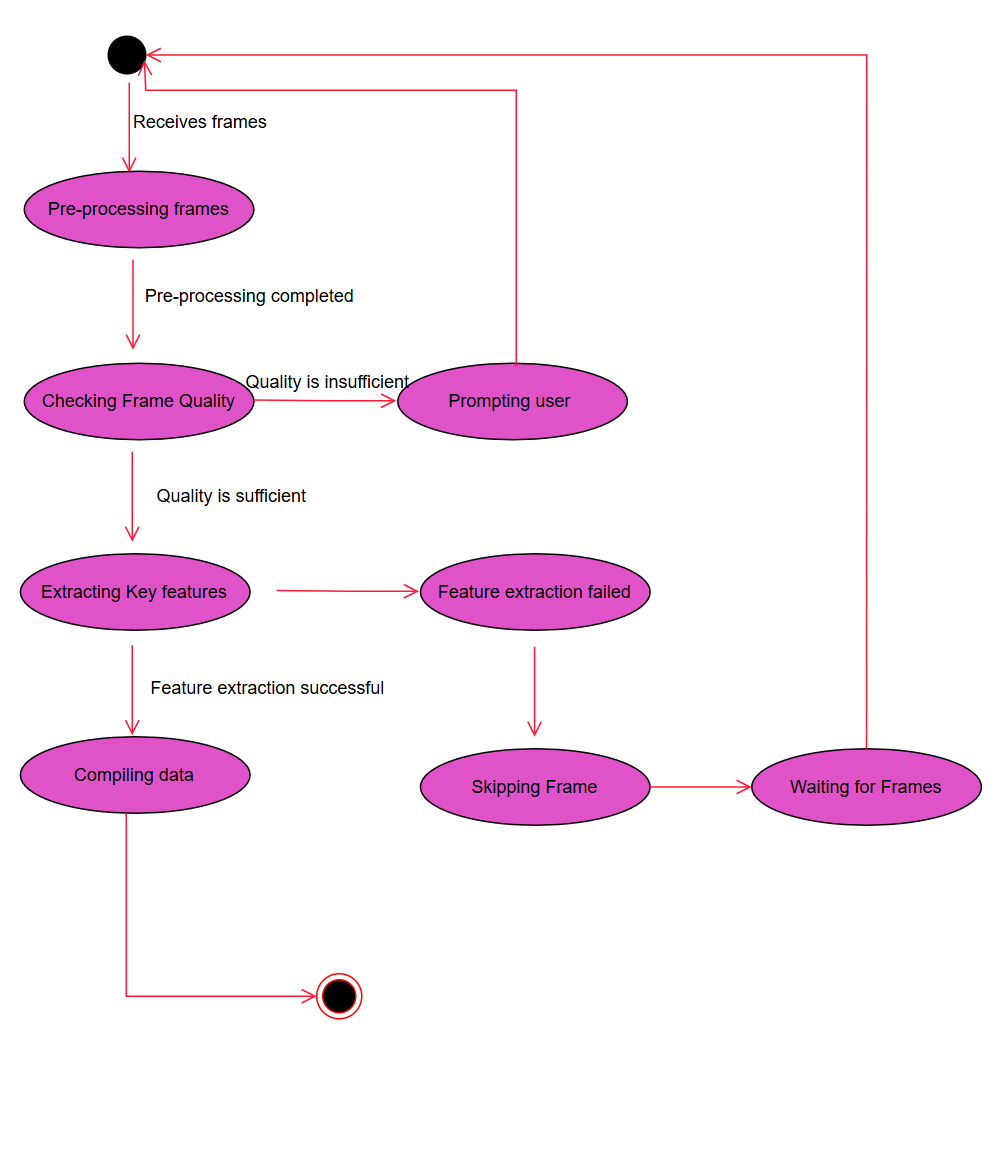
## 5.2 State Transition Diagrams

State diagrams for the Adaptive HCI project illustrate the different system states involved in processing and rendering user gestures, showing transitions based on data validation, error handling, and interactions within the AR/VR environment.

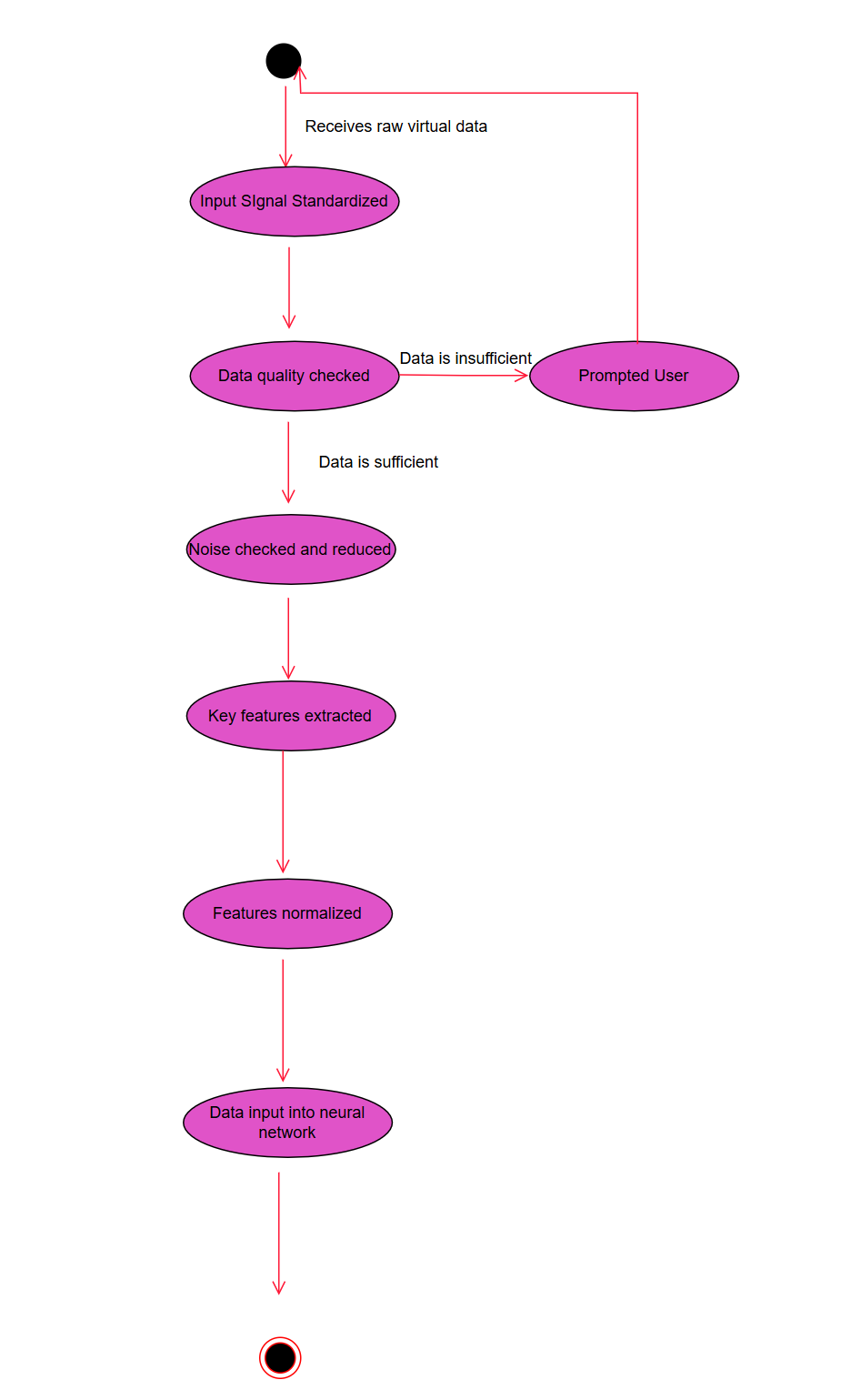
### 5.2.1 Mobile Application



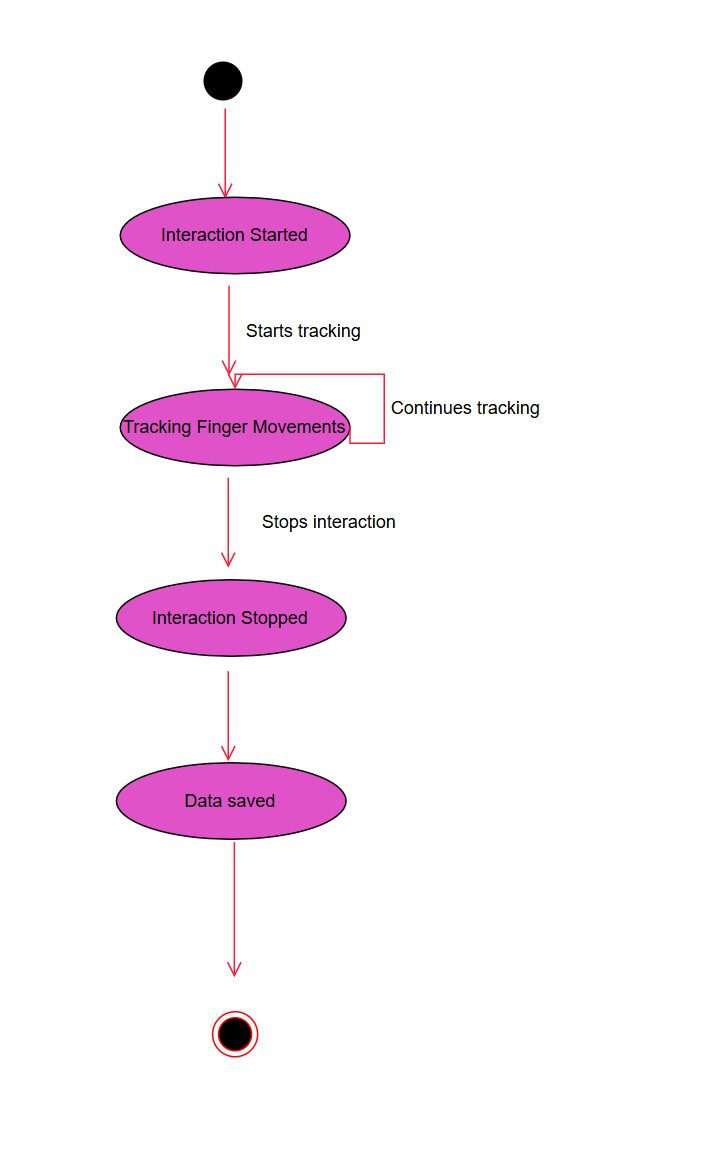
### 5.2.2 Image Processing



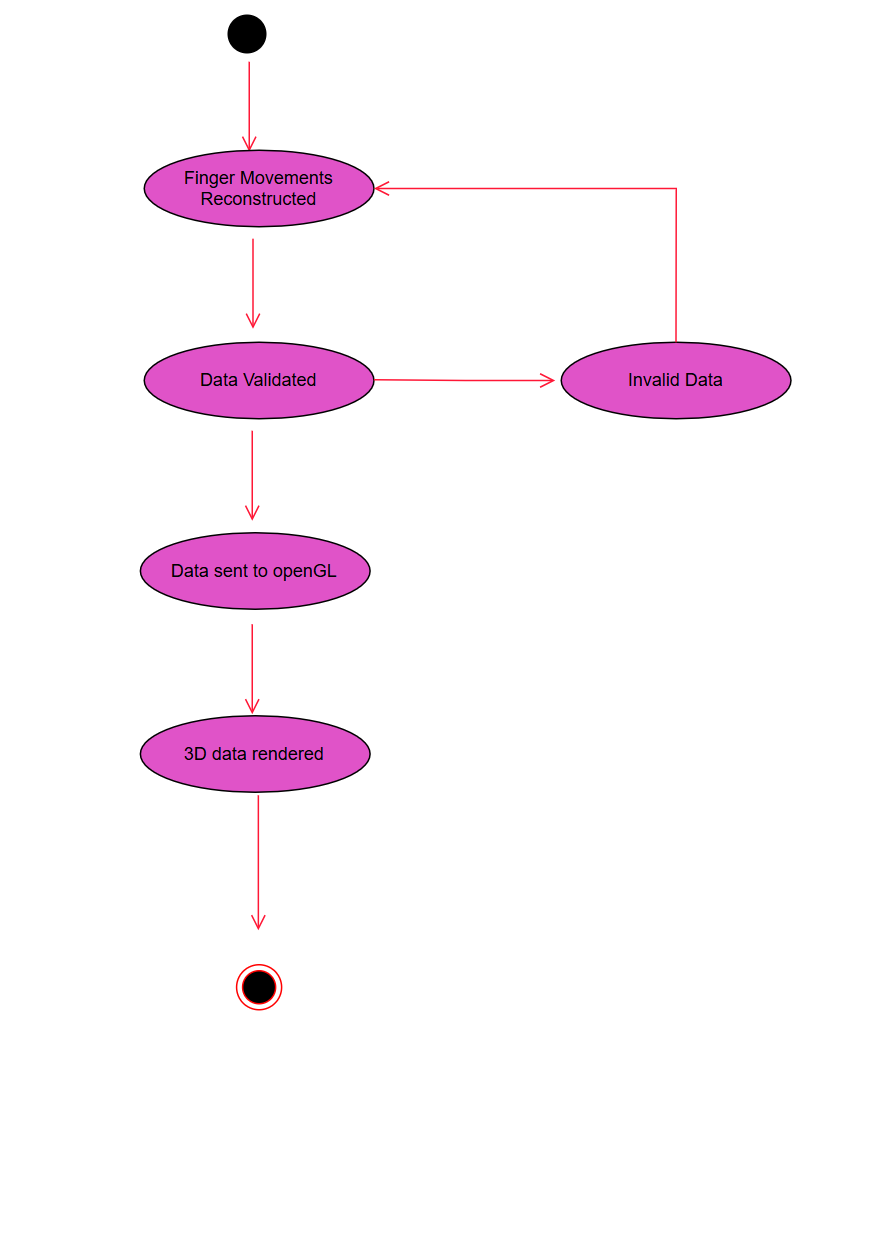
### 5.2.3 Machine Learning



### 5.2.4 AR/VR



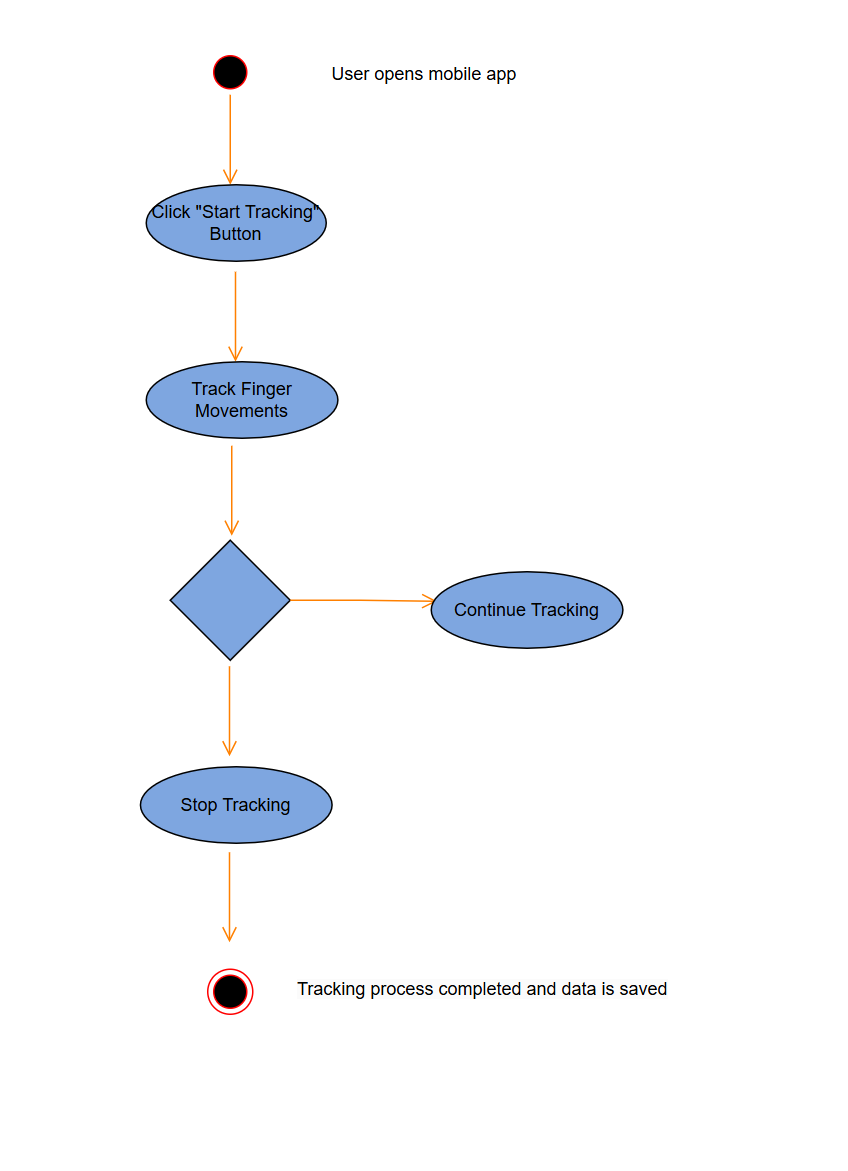
### 5.2.5 3D Finger Tracking for Air Writing Recognition



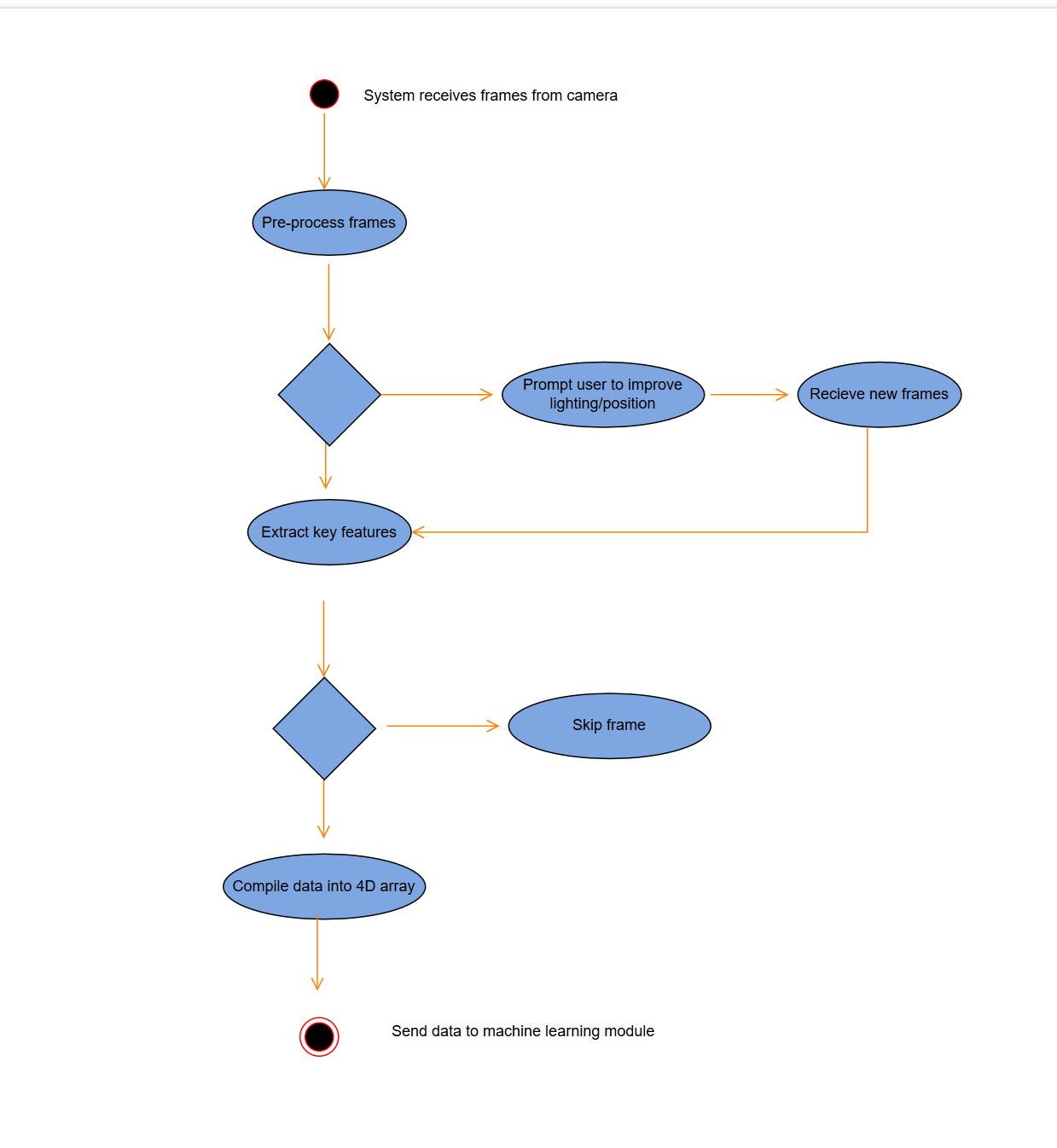
## 5.3 Activity Diagrams

Activity diagrams for the **Adaptive HCI** project illustrate the workflow and sequence of actions for capturing, processing, and displaying smoothed gesture data in real-time.

### 5.3.1 Mobile Application



### 5.3.2 Image Processing

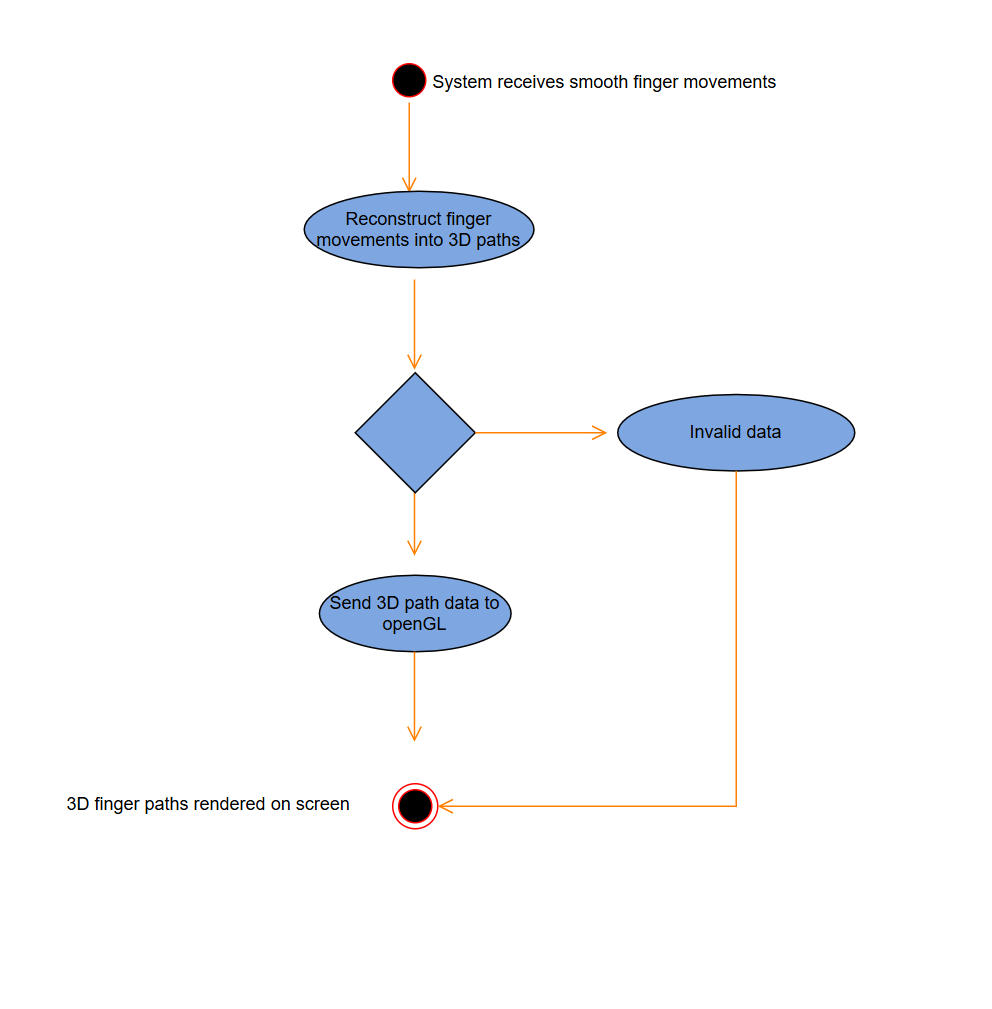


### 5.3.3 Machine Learning

### 5.3.4 AR/VR

### 

### 5.3.5 3D Finger Tracking for Air Writing Recognition



# 6.0 Restrictions, Limitations and Constraints

**Special issues which impact the specification, design, or implementation of the software are noted here.**

**Time**

This project has a significant time constraint, with a limited window to complete all documentation, development, testing, and refinement tasks. The ambitious goals of real-time gesture tracking, smoothing, and visualization, along with publishing requirements, add to the time pressure. While additional enhancements and features could be beneficial, they may be challenging to implement within the allotted time frame.

**Client Support**

The client is highly supportive, providing essential resources and guidance throughout the project. This includes:

* Access to required hardware, such as smartphones and VR sets, for testing and development.
* Ongoing support and advice on technical challenges, particularly in image processing, machine learning, and hardware integration.
* Assistance in drafting, reviewing, and preparing materials for publication in the client’s journal, ensuring the project’s research goals are met.

# 7.0 Validation Criteria

The **Adaptive HCI** mobile application is developed with the goal of real-time video recording, smoothing, and visualization, providing users with a seamless and responsive experience. The app should allow users to draw or write gestures in the air, with instant feedback displayed on either a smartphone screen or a VR set, closely mirroring the movement without noticeable delay.

## 7.1 Testing Methodology

To validate the app’s ability to perform in real time, we will conduct **black-box testing** focused on the application’s core features: gesture detection, real-time processing, smoothing, and visualization. Validation will include measuring processing speed, responsiveness, and the fluidity of feedback displayed on the user’s device (phone or VR set).

## 7.1 Classes of Tests

The following classes of tests will be conducted to validate core functionality, emphasizing black-box testing to evaluate real-time gesture tracking, processing, smoothing, and visualization:

1. **Real-Time Responsiveness Tests**:
   * **Objective**: To ensure that the app captures and displays gestures instantly, maintaining a smooth and responsive experience.
   * **Focus**: Measures latency between physical gestures and on-screen or VR display feedback.
2. **Continuous Frame Processing Tests**:
   * **Objective**: To validate that the app maintains smooth, uninterrupted 60Hz frame processing during continuous tracking.
   * **Focus**: Evaluates frame drop rates, processing consistency, and stability over extended use.
3. **Smoothing Algorithm Performance Tests**:
   * **Objective**: To confirm that the neural network-based smoothing algorithm operates in real time, preserving responsiveness.
   * **Focus**: Monitors the effect of smoothing on the visual display, ensuring that transitions appear smooth without delay.
4. **PWM Filtering Accuracy Tests**:
   * **Objective**: To validate PWM filtering accuracy, particularly in challenging lighting environments, ensuring correct tracking of the LED signal.
   * **Focus**: Assesses the app’s ability to isolate and maintain a lock on the correct LED source in high-light or multi-light conditions.
5. **3D Path Visualization Quality Tests**:
   * **Objective**: To ensure that 3D visualizations of gesture paths are smooth, continuous, and synchronized with user movements.
   * **Focus**: Checks for lag or disjointed movement in the rendered path, focusing on the overall quality and stability of real-time feedback.

## 7.2 Expected Software Response

The following outlines the expected results from each class of tests, specifying the criteria for a successful software response:

1. **Real-Time Responsiveness**:
   * **Expected Outcome**: The app should achieve latency under 50ms, ensuring that gestures appear on-screen or in VR without perceptible delay. User movements should be mirrored accurately and in real time.
2. **Continuous Frame Processing**:
   * **Expected Outcome**: The app should process frames at a continuous 60Hz rate with no dropped frames or noticeable lags over a period of at least 5 minutes. The feedback should be consistent and uninterrupted, providing a fluid user experience.
3. **Smoothing Algorithm Effectiveness**:
   * **Expected Outcome**: The app should display smoothed gestures immediately, with no lag introduced by the smoothing process. Transitions should appear seamless, and the gesture path should maintain high accuracy and fluidity.
4. **PWM Filtering Accuracy in Boundary Conditions**:
   * **Expected Outcome**: The app should accurately filter and track the LED glove source, even in challenging environments like bright ambient lighting or multiple light sources. The tracking should remain locked onto the correct signal, providing real-time visualization without interruption.
5. **3D Path Visualization Quality**:
   * **Expected Outcome**: The 3D path of gestures should appear smooth, continuous, and closely follow the user’s movements in real time. There should be no perceptible lag, jitter, or disjointed movements, maintaining a high-quality visual experience.

## 7.3 Performance bounds

The following performance requirements outline the minimum and target performance thresholds necessary to ensure a seamless user experience across gesture tracking, smoothing, and visualization functionalities. These bounds establish real-time responsiveness, system efficiency, and quality standards for the **Adaptive HCI** application.

### 7.3.1 Latency

* **Target Latency**: Less than 50 milliseconds from gesture input to visual display.
  + **Description**: This latency ensures near-instantaneous feedback, allowing users to see gestures closely mirror their hand movements in real-time.
  + **Acceptable Maximum**: Up to 100 milliseconds. If latency exceeds this limit, the system may degrade responsiveness and impact user experience, especially in VR applications.

### 7.3.2 Frame Rate

* **Target Frame Rate**: Continuous 60 frames per second (fps).
  + **Description**: A steady 60fps is necessary to maintain smooth and uninterrupted tracking and visualization, with minimal flicker or stutter.
  + **Acceptable Minimum**: 45fps. Frame rates below this threshold could result in noticeable lag and reduce the perceived smoothness of gestures.

### 7.3.3 Gesture Smoothing and Stabilization

* **Smoothing Processing Time**: Each smoothing calculation should be completed in under 10 milliseconds.
  + **Description**: This processing time ensures that the neural network model operates without causing noticeable delays, supporting a real-time response.
  + **Acceptable Deviation**: Minor delays up to 15 milliseconds may occur in complex lighting or gesture scenarios but should remain below this threshold to preserve real-time interaction quality.

### 7.3.4 Environmental Adaptation Response

* **Automatic Light Adjustment Time**: The system should adjust for changes in ambient lighting conditions within 500 milliseconds.
  + **Description**: The system must quickly adapt to challenging or changing lighting conditions to maintain accurate gesture tracking.
  + **Acceptable Range**: Between 500 and 750 milliseconds in extreme light fluctuations. If adaptation exceeds this range, the system might struggle to accurately isolate the LED source, impacting tracking quality.

### 7.3.5 Battery and Resource Efficiency

* **CPU Usage**: The app should utilize no more than 50% of available CPU resources on standard devices (e.g., mid-range Android smartphones).
  + **Description**: Ensuring efficient CPU usage prevents excessive battery drain and overheating, allowing users to interact with the application over extended periods.
* **Battery Consumption**: The app should consume no more than 15% battery per hour during continuous use in standard conditions.
  + **Description**: This consumption rate allows users to use the app for reasonable durations without excessive battery drain, accommodating mobile usage in research or experimental settings.

### 7.3.6 Data Storage and Transfer

* **Data Storage for Gesture Data**: Each recorded session should not exceed 5 MB in storage to accommodate multiple sessions on the device.
  + **Description**: Efficient storage of gesture data allows users to record multiple sessions without overwhelming device storage, preserving session quality for future analysis or model adaptation.
* **Optional Cloud Data Transfer Latency** (if cloud storage is enabled): Data should be uploaded within 2 seconds of session completion.
  + **Description**: For registered users opting for cloud storage, data transfer should be quick and seamless, allowing efficient storage of neural network adjustments and session data.

# 8.0 Appendices

This section provides supplementary information relevant to the **Adaptive HCI** project, including a traceability matrix, research-focused product strategy, analysis metrics, and additional resources.

## 8.1 System traceability matrix

The system traceability matrix provides a mapping of the software requirements to corresponding functional and technical specifications, ensuring that each requirement is addressed in the system’s design and implementation. This traceability matrix will support the research objective by confirming alignment between the intended goals (e.g., real-time smoothing, VR/AR integration) and the actual functionality developed for demonstration purposes.

| Requirement ID | Description | System Specification | Verification Method | Test Case ID |
| --- | --- | --- | --- | --- |
| REQ-01 | Real-time gesture tracking and smoothing | Functional Model, Section 4.1.1 (Mobile Application) | Black-box testing | TC-01 |
| REQ-02 | VR/AR integration for 3D gesture visualization | Software Interface, Section 4.2.1 (External Machine Interfaces) | Real-time testing in VR/AR environment | TC-02 |
| REQ-03 | Adaptive lighting adjustment to ensure LED tracking | Behavioral Model, Section 5.1.1 (Environment Adaptation) | Boundary condition testing in varying lighting environments | TC-03 |
| REQ-04 | User control for tracking start/stop and settings adjustments | Functional Model, Section 4.1.4 (Triggers) | UI/UX testing for responsiveness and functionality | TC-04 |
| REQ-05 | Storage of gesture data for analysis | Functional Model, Section 4.1.6 (Post Conditions) | Data integrity testing after session end | TC-05 |
| REQ-06 | Smoothing and stabilization using CNN-LSTM model | Functional Model, Section 4.1.5 (Image Processing and ML modules) | Performance testing for real-time responsiveness | TC-06 |
| REQ-07 | Environmental adaptation for boundary lighting conditions | Functional Model, Section 4.1.5 (Image Processing) | Adaptive response testing to adjust tracking in bright and low light | TC-07 |
| REQ-08 | Real-time 3D rendering for gesture visualization | Functional Model, Section 4.1.5 (3D Finger Tracking) | Real-time rendering verification in 3D mode | TC-08 |
| REQ-09 | Cloud storage option for user-specific neural network preferences | Software Interface, Section 4.2.2 (External System Interfaces) | Cloud integration testing (if implemented) | TC-09 |
| REQ-10 | Customizable user settings for gesture smoothing and light filtering | Human Interface, Section 4.2.3 (Settings Screen) | UI functionality testing for settings adjustment | TC-10 |
| REQ-11 | Error handling for hardware connectivity (Camera and Gloves) | Functional Model, Section 4.1.7 (Exceptions) | Fault injection testing for camera and glove connectivity issues | TC-11 |
| REQ-12 | Intuitive user interface and real-time feedback | Human Interface, Section 4.2.3 (Tracking Screen) | Usability testing for intuitive design and real-time responsiveness | TC-12 |

## 8.2 Product Strategies

Since the **Adaptive HCI** project is a proof of concept intended for research publication rather than commercial distribution, the product strategy focuses on showcasing technical feasibility and innovation in gesture tracking for accessibility.

* **Primary Goal**: To validate the effectiveness of real-time gesture tracking and smoothing technology, especially for users with hand instability, and to publish findings in a relevant journal.
* **Research Outcome**: The project will provide a basis for potential future studies on improving gesture-based interactions in VR/AR environments, especially for accessibility and medical applications.
* **Open-Source Consideration**: Following publication, the project may be released as open-source software, allowing the research community to build upon this foundation and explore additional applications in healthcare and human-computer interaction.

## 8.3 Analysis metrics to be used

The following metrics will be used to evaluate the performance and usability of the **Adaptive HCI** system. These metrics are aligned with the project’s research focus on responsiveness, accuracy, and user experience.

* **Latency (ms)**: Measures the time delay from gesture input to visual output, with a target of under 50 milliseconds.
* **Frame Rate Consistency (fps)**: Assesses stability and smoothness in real-time tracking, targeting a continuous 60fps rate.
* **Gesture Smoothing Accuracy (%)**: Evaluates the effectiveness of the neural network in reducing tremors, with accuracy measured against known gesture patterns.
* **Environmental Adaptation Efficiency (ms)**: Measures the system’s response time to significant lighting changes, targeting a sub-500 ms adjustment time.
* **User Satisfaction Rating**: Qualitative feedback collected through user testing, assessing the perceived responsiveness and ease of use, especially for users with hand instability.

## 8.4 Supplementary information

Additional resources and references to support the **Adaptive HCI** project will eventually include:

* **LED Glove Specifications**: Detailed technical specifications of the LED gloves, including PWM frequency settings and sensor compatibility.
* **Model Training Documentation**: Information on the trained CNN-LSTM model used for gesture smoothing, along with training parameters and sample data.
* **Publication and Research Resources**: Journal and conference recommendations for submitting the research paper, including formatting guidelines and submission timelines.

# 

# 

# 

# 

# **Review and Sign Off**

| **Name** | **Project Team Role** | **Signature** | **Date** |
| --- | --- | --- | --- |
| Soham Naik | Lead Image Processing & Computer Vision Engineer | Soham Naik | 11/12/2024 |
| Deniz Acikbas | Lead Mobile App & VR/AR Engineer | Deniz Acikbas | 11/12/2024 |
| Zaynab Mourtada | Lead Machine Learning Engineer | Zaynab Mourtada | 11/12/2024 |
| Alan Raj | Lead 3D Visualizer Engineer | Alan Raj | 11/12/2024 |
| Xiao Zhang | Client | Xiao Zhang | 11/12/2024 |