Adaptive HCI:

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

Test Specification Document

Version No.  1.0

**Project Document Revision History**

| **VersionNumber** | **Date** | **Revision Author** | **Description of Revision** |
| --- | --- | --- | --- |
| 1.0 | 03/16/2025 | Team 5 | Original Published Version |
|  |  |  |  |

[**1.0 Introduction 4**](#_heading=h.7byh8hx9ivk4)

[1.1 Goals and objectives 4](#_heading=h.tk3j9vd2lt9x)

[1.2 Statement of scope 4](#_heading=h.oyggpk4876yg)

[1.3 Major constraints 5](#_heading=h.ode0cuqg0zo3)

[**2.0 Test Plan 5**](#_heading=h.bioj8g6nqmtm)

[2.1 Software (SCIís) to be tested 5](#_heading=h.jgxqxrugy923)

[2.2 Testing strategy 6](#_heading=h.y58hsrdj608k)

[2.2.1 Unit testing 6](#_heading=h.oelfyda4hjuo)

[2.2.2 Integration testing 6](#_heading=h.khevvefzz6di)

[2.2.3 Validation testing 7](#_heading=h.g3scf1rv51ma)

[2.2.4 High-order testing 7](#_heading=h.71r4j1ylsg59)

[2.3 Testing resources and staffing 8](#_heading=h.xhjctt4r0m5c)

[2.4 Test work products 8](#_heading=h.7qggqr4qq35y)

[2.5 Test record keeping 8](#_heading=h.ma2oyaaide6)

[2.6 Test metrics 9](#_heading=h.gxvy8zb0msqb)

[2.7 Testing tools and environment 10](#_heading=h.emc2ackusnrt)

[2.8 Test schedule 10](#_heading=h.pthezulgrqps)

[2.8.1 - Week 1 (March 17 - March 23) → Unit & Initial Integration Testing 10](#_heading=h.v7y9xsyt8zcd)

[2.8.2 - Week 2 (March 24 - March 30) → Full System & Performance Testing 10](#_heading=h.7ccj2shfx0lz)

[2.8.3 - Week 3 (March 31 - April 6) → Validation & Stress Testing 11](#_heading=h.rm7prpxp0nn2)

[2.8.4 - Week 4 (April 7 - April 11) → Final Optimization & Report Preparation 11](#_heading=h.hdn1f7hoarkm)

[**3.0 Test Procedure 11**](#_heading=h.fex4pr3b0e3b)

[3.1 Software (SCIís) to be tested 11](#_heading=h.2xseq6vvvq4m)

[3.2 Testing procedure 11](#_heading=h.yghj1yixn187)

[3.2.1 Unit test cases 12](#_heading=h.1t07homk1tj8)

[3.2.1.2 Stubs and/or drivers for component i 12](#_heading=h.xld199fex9w4)

[3.2.1.3 Test cases component i 12](#_heading=h.8yehk6ikm04h)

[3.2.1.4 Purpose of tests for component i 12](#_heading=h.8rw85dmjgioj)

[3.2.1.5 Expected results for component i 12](#_heading=h.d5kfw861j5ge)

[3.2.2 Integration testing 12](#_heading=h.kduo8pnpvkdl)

[3.2.2.1 Testing procedure for integration 12](#_heading=h.34s6m9lg7xf6)

[3.2.2.2 Stubs and drivers required 13](#_heading=h.1gcxfhty8h1t)

[3.2.2.3 Test cases and their purpose 13](#_heading=h.gbnm6rjooiex)

[3.2.2.4 Expected results 13](#_heading=h.gnlrnfr23bj6)

[3.2.3 Validation testing 13](#_heading=h.xicee2svq0r)

[3.2.3.1 Testing procedure for validation 13](#_heading=h.j5y9ankr213y)

[3.2.3.3 Expected results 13](#_heading=h.85lcwuvxtzkw)

[3.2.3.4 Pass/fail criterion for all validation tests 13](#_heading=h.k0tnywdrnbo)

[3.2.4 High-order testing (a.k.a. System Testing) 13](#_heading=h.kogw2opp2ob2)

[3.2.4.1 Recovery testing 13](#_heading=h.rtchf0848ub3)

[3.2.4.2 Security testing 13](#_heading=h.vei4rve9nph8)

[3.2.4.3 Stress testing 13](#_heading=h.t3n10urkw8l2)

[3.2.4.4 Performance testing 13](#_heading=h.10p297vidqg7)

[3.2.4.5 Alpha/beta testing 13](#_heading=h.f4a6nnwbucf7)

[3.2.4.6 Pass/fail criterion for all validation tests 13](#_heading=h.8taj9dty3uh2)

[3.3 Testing resources and staffing 13](#_heading=h.y5j0ayajp0d0)

[3.4 Test work products 14](#_heading=h.e8kc8i1v14f3)

[3.5 Test record keeping and test log 14](#_heading=h.dmsykqgrdfw4)

# 1.0 Introduction

This document provides a comprehensive test plan and testing procedures for the Adaptive HCI system, ensuring its functionality, performance, and reliability. The document outlines the overall goals and objectives of testing, defines the scope of the testing process, and highlights any constraints that impact the testing methodology. By systematically verifying each component, we aim to enhance the stability, accuracy, and real-time performance of the Adaptive HCI system.

## 1.1 Goals and objectives

The primary goal of the testing process is to validate that the Adaptive HCI system performs as intended, ensuring that real-time air-writing tracking, stroke stabilization, and text inference operate with high accuracy and minimal latency. The testing process aims to:

* Verify the system’s ability to track hand gestures and air-written strokes using LED-equipped gloves.
* Ensure real-time stabilization and smoothing of hand movements to reduce tremors and improve handwriting accuracy.
* Test the accuracy of text inference models in converting air-written strokes into digital text.
* Evaluate system responsiveness and real-time performance on supported mobile devices.
* Identify and mitigate potential performance bottlenecks in image processing, object detection, and visualization.
* Assess the system’s ability to function effectively under varying environmental conditions (lighting, background interference, and motion variability).

By implementing a structured testing methodology, we aim to effectively validate our technology’s performance and robustness while maintaining an intuitive and functional user interface for demonstration purposes.

## 1.2 Statement of scope

The scope of the testing process includes all major functionalities of the Adaptive HCI system, ensuring that each component operates as expected. The key features to be tested include:

* **Real-Time Gesture Tracking:** Assess the system’s ability to capture LED-equipped glove movements and generate accurate 3D motion paths.
* **Path Smoothing and Tremor Reduction:** Test the effectiveness of Kalman filtering and interpolation techniques in stabilizing air-written strokes.
* **Text Inference Accuracy:** Evaluate the system’s ability to correctly interpret air-written strokes and convert them into digital text.
* **System Performance and Responsiveness:** Measure the frame processing speed, real-time visualization efficiency, and overall system latency.
* **User Interface and Experience:** Verify that the mobile application’s interface is intuitive, responsive, and provides real-time feedback.
* **Environmental Robustness:** Ensure system functionality under different lighting conditions and background environments.
* **Multi-User Support:** Validate the system’s ability to differentiate multiple users using On-Off Keying (OOK) LED signals.

Features or functionalities that will **not** be tested include:

* Cloud-based processing, as all computations occur locally on the device.
* Multi-device compatibility beyond the target devices (Motorola G Play and Google Pixel 8A).
* Extended multi-user support beyond three simultaneous users.

## 1.3 Major constraints

Several constraints impact the testing and development of the Adaptive HCI system:

* **Hardware Limitations:** The system is optimized for smartphones with rolling shutter cameras and may experience performance variations on different hardware.
* **Real-Time Processing Requirements:** Achieving a smooth real-time experience necessitates efficient processing techniques, including hardware-accelerated inference and optimized frame processing.
* **Lighting and Environmental Factors:** Variability in ambient lighting and background interference can affect LED detection accuracy, necessitating robust preprocessing techniques.
* **Data Privacy and Security:** All processing occurs locally on the device to ensure user privacy, meaning no cloud-based storage or external data transmission is involved.
* **Limited Testing Time Frame:** The project operates within an academic semester, requiring efficient milestone tracking and rapid iteration during testing.

# 2.0 Test Plan

This section describes the overall testing strategy and the project management issues that are required to properly execute effective tests.

## 2.1 Software (SCIís) to be tested

The software components of the **Adaptive HCI** system that will undergo testing include:

* **Mobile Application (Xamera)** – Captures real-time video input, tracks LED-equipped hand movements, and provides real-time visualization.
* **Image Processing Module** – Preprocesses video frames by adjusting brightness, reducing noise, and detecting LED positions for contour tracking.
* **Object Detection Module (YOLO-Based Tracking)** – Detects and tracks the LED marker on gloves and differentiates multiple users using On-Off Keying (OOK).
* **Path Smoothing Module** – Applies Kalman filtering and interpolation techniques to stabilize air-writing strokes.
* **Text Inference Module** – Converts air-written strokes into digital text using deep learning models.
* **3D Visualization Module** – Renders stabilized strokes and predicted text in a virtual 3D space, supporting AR/VR visualization.
* **User Interface (UI)** – Ensures real-time interaction, user customization options, and responsive feedback.

**Exclusions:**

* **Cloud-Based Processing** – The system operates entirely on the device, with no cloud dependency.
* **Multi-Device Compatibility Beyond Target Hardware** – Testing will be conducted on Motorola G Play and Google Pixel 8A; performance on other devices will not be validated.
* **Multi-User Tracking Beyond Three Users** – The system is optimized for tracking up to three users using OOK differentiation; additional users will not be tested.

## 2.2 Testing strategy

The testing process will follow a structured approach, ensuring each module functions correctly in isolation before being integrated into the full system. The strategy includes **unit testing**, **integration testing**, **validation testing**, and **high-order testing** to ensure correctness, performance, and usability.

### 2.2.1 Unit testing

**Objective:** Verify the correctness of individual software components before integration.  
**Scope:**

* Each module will be tested independently to ensure its core functionality works as expected.
* Unit tests will focus on:
  + **Frame Processing Functions:** Ensure frames are captured, preprocessed, and passed correctly to object detection.
  + **Object Detection Accuracy:** Validate YOLO model’s ability to detect LED markers in different conditions.
  + **Path Smoothing Algorithms:** Check that Kalman filtering effectively reduces motion jitter and improves stroke clarity.
  + **Text Inference Module:** Ensure correct mapping of gesture data to text output.
  + **UI Components:** Verify UI elements respond to user interactions correctly.

**Selection Criteria:**

* Components critical to real-time performance and accuracy.
* Functions with the highest complexity or likelihood of failure.

### 2.2.2 Integration testing

**Objective:** Ensure smooth data flow between different modules and validate interaction correctness.

**Integration Order:**

1. **Mobile Application → Image Processing Module** – Validate that the mobile app correctly captures and transmits frames to the image processing module.
2. **Image Processing → Object Detection** – Ensure detected LED positions match actual movement paths.
3. **Object Detection → Path Smoothing** – Confirm the processed LED data is accurately smoothed without excessive lag.
4. **Path Smoothing → Text Inference** – Verify that stabilized stroke paths are correctly mapped to text predictions.
5. **Text Inference → 3D Visualization** – Ensure inferred text and air-writing strokes are correctly rendered in the UI and AR/VR space.
6. **Full System Testing** – Conduct end-to-end testing to evaluate system reliability under different user interactions.

**Key Considerations:**

* Latency between modules should remain within acceptable real-time constraints (<50ms per frame as defined by our Client).
* Data loss during transmission between modules must be minimized.
* Error handling and exception recovery should be validated during integration.

### 2.2.3 Validation testing

**Objective:** Ensure the entire system meets functional and non-functional requirements under realistic conditions.

**Validation Criteria:**

* **Real-time Gesture Tracking:** Verify that the system accurately tracks LED motion without significant delays.
* **Text Conversion Accuracy:** Validate that air-written strokes are correctly translated into text.
* **User Experience (UX):** Ensure the system responds smoothly to user inputs and displays real-time feedback.
* **Environmental Testing:** Assess performance under different lighting conditions and user movement speeds.
* **Device Performance:** Measure CPU/GPU utilization and battery consumption to ensure sustainability during prolonged use.

**Testing Order:**

1. Conduct validation in a controlled environment with optimal conditions.
2. Introduce variations (lighting, background interference, hand tremors) to simulate real-world conditions.
3. Conduct user testing with volunteers to gather feedback on usability and responsiveness.

### 2.2.4 High-order testing

**Objective:** Evaluate system robustness, performance limits, and long-term stability.

**Types of High-Order Tests:**

* **Stress Testing:** Simulate extended usage over long durations to identify potential memory leaks, overheating, or processing slowdowns.
* **Performance Benchmarking:** Measure FPS, processing time per frame, and latency to ensure the system meets real-time constraints.
* **Security Testing:** Ensure data privacy by verifying that all processing occurs locally without external data leaks.

**Responsibility:**

* The development team will conduct internal stress and performance tests.
* Security assessments will be performed to validate compliance with privacy standards.
* Performance benchmarking, including frame rate analysis and latency testing, will be handled by the development team using Android Profiler and OpenGL frame logging. AR/VR visualization tests will be conducted in both controlled and real-world conditions to assess smoothness and accuracy.

## 2.3 Testing resources and staffing

**Testing Resources:**

* **Test Devices:** Motorola G Play, Google Pixel 8A
* **LED-Equipped Gloves:** Hardware used for real-time gesture tracking
* **Camera Setup:** Smartphone rolling shutter camera for video processing
* **VR/AR Headsets:** Optional testing for immersive visualization
* **Development and Testing Tools:** OpenCV, TensorFlow Lite, PyTorch (TorchScript), Android Debug Bridge (ADB)
* **Software Simulators:** Android Studio Emulator for mobile app testing

**Staffing:**

* **Software Engineers:** Responsible for unit testing, integration testing, and debugging
* **QA Testers:** Execute test cases, report defects, and validate expected outcomes
* **Machine Learning Engineers:** Evaluate model performance and improve text inference/object detection accuracy
* **Project Supervisors:** Oversee testing milestones and ensure alignment with project goals

## 2.4 Test work products

The following artifacts will be generated as part of the testing process:

* **Test Cases Document:** Covers unit, integration, validation, and high-order testing scenarios
* **Bug Reports:** Logs detailing software defects, conditions under which they occur, and steps to reproduce them
* **Performance Reports:** Measures frame rates, processing times, and memory usage
* **Accuracy Metrics:** Evaluates object detection, gesture smoothing, and text inference performance
* **Test Summary Reports:** Consolidates test results and key findings for each testing phase

## 2.5 Test record keeping

All test results, bug reports, and performance evaluations will be stored using the following mechanisms:

* **Google Drive/GitHub Repository:** Storing test documentation and scripts
* **Issue Tracking System (GitHub Issues):** Logging and tracking bugs with priority levels
* **Performance Log Files:** Storing real-time logs for debugging latency and frame rate issues

Each test session will be documented, including:

* Test date and time
* Device and software version used
* Test scenario executed
* Observed results and deviations from expected behavior

## 2.6 Test metrics

To ensure the system meets performance and accuracy goals, the following metrics will be used:

**Performance Metrics:**

* **Frame Processing Time:** Should remain under **50ms per frame** for real-time execution
* **Latency:** The end-to-end latency from gesture input to on-screen visualization should remain under **50ms**, ensuring a minimum frame rate of **20 FPS**. Each frame's processing, including video capture, object detection, line rendering, and post-processing, must complete within **50ms** to maintain real-time responsiveness.
* **VR/AR Rendering Performance: The system should maintain a minimum of 30 FPS for smooth visualization.**
  + **Benchmarking Method:** FPS will be logged using Android Profiler and OpenGL frame rendering tests, ensuring that AR/VR environments remain responsive.
  + **Performance Goal:** The system should maintain at least 30 FPS for usability. If FPS drops below 25 FPS, visual latency and motion stutter will be considered excessive.

**Accuracy Metrics:**

* **Gesture Detection Accuracy:** At least **90% accuracy** in tracking LED-equipped glove movements.
  + **Benchmarking Method:** Accuracy will be evaluated using a **ground truth comparison**—manual annotation of LED positions versus system-detected positions across a dataset of air-writing gestures.
  + **Performance Goal:** The system should successfully track LED motion **in at least 90% of frames** under controlled conditions and maintain **above 85% accuracy in varied lighting conditions**.
* **Text Inference Accuracy:** At least **90% accuracy** in converting air-written strokes into text.
  + **Benchmarking Method:** Accuracy will be calculated using a **Levenshtein Distance (edit distance) metric**, comparing system-generated text to expected input.
  + **Performance Goal:** The system should recognize **at least 9 out of 10 characters correctly** in a standard test dataset, accounting for stroke variation.
* **Path Smoothing Effectiveness:** Reduce tremor variance by **≥70%**
  + **Benchmarking Method:** Tremor reduction will be measured by comparing the variance in the raw vs. smoothed gesture path using mean squared error (MSE) and signal-to-noise ratio (SNR).
  + **Performance Goal:**
    - Smoothed gesture paths should maintain at least 70% reduction in variance compared to raw data.
    - Path smoothing should not introduce artificial distortions and should closely match expected hand motion.
    - Smoothing computations should remain within 20ms per frame to ensure minimal delay in real-time rendering.

**User Experience Metrics:**

* **AR/VR Rendering Quality:** 3D visualization should have **no perceptible lag**
* **Smooth Mobile App User:** UI and window switching should have **no perceptible lag**

## 2.7 Testing tools and environment

**Testing Tools:**

* **Software Testing Tools:** Android Debug Bridge (ADB), Android Profiler, Firebase Test Lab
* **Object Detection and AI Testing:** TensorFlow Lite, OpenCV for real-time tracking validation
* **Performance Monitoring:** Systrace (Android), Perfetto for analyzing frame drops and CPU load
* **Bug Tracking:** GitHub Issues for managing software defects

**Testing Environment:**

* **Controlled Lab Setup:** Low-light and bright-light conditions for LED tracking tests
* **Real-World Scenarios:** Public spaces and different room environments for robustness testing
* **Mobile Device Testing:** Motorola G Play (low-end device), Google Pixel 8A (high-end device)

Special considerations:

* **Gesture Variability:** Testing across different hand speeds, movement styles, and backgrounds

## 2.8 Test schedule

### 2.8.1 - Week 1 (March 17 - March 23) → Unit & Initial Integration Testing

Unit Testing:

* Frame extraction & image processing validation
* Object detection accuracy (YOLO-based LED tracking)
* Path smoothing (Kalman filter) & text inference accuracy

Initial Integration Testing:

* Mobile app → Image processing module communication
* Image processing → Object detection module validation

### 2.8.2 - Week 2 (March 24 - March 30) → Full System & Performance Testing

Complete integration testing:

* Object detection → Path smoothing → Text inference validation
* UI responsiveness & real-time feedback testing
* Multi-user tracking with OOK signal differentiation

Performance Testing:

* Latency assessment (target <50ms per frame - 20FPS)

### 2.8.3 - Week 3 (March 31 - April 6) → Validation & Stress Testing

Real-world scenario testing:

* Different lighting conditions (low light, bright light, moving backgrounds)
* User testing for gesture tracking accuracy & usability

System robustness testing:

* VR/AR visualization performance
* Stress testing (extended use, memory leaks, CPU load)
* Security & privacy validation (ensuring local-only processing)

### 2.8.4 - Week 4 (April 7 - April 11) → Final Optimization & Report Preparation

* Final bug fixes & performance optimizations
* Consolidate test reports & documentation
* Final user acceptance testing

# 3.0 Test Procedure

This section describes the software components to be tested and the step-by-step testing procedure, including responsible testers, tools, and documentation processes.

## 3.1 Software (SCIís) to be tested

#### **Xamera (Main App)**

* *Xamera is the primary application that handles:*
  + ***Rolling shutter camera operations***
  + ***Real-time video processing*** *(OpenCV)*
  + ***Machine learning inference*** *(TensorFlow Lite)*
  + ***Camera optimizations & settings***
* *Xamera is written in* ***Kotlin*** *and integrates* ***TFLite for machine learning inference*** *and* ***OpenCV for real-time video processing.***
* *It is the core environment where* ***gesture tracking, stabilization, and text inference*** *occur.*

#### **Xamera AR (Augmented Reality Component)**

* *Xamera AR is a separate* ***intent*** *that launches within Xamera, handling the* ***3D visualization of air-writing strokes.***
* *It uses* ***Google ARCore*** *for managing AR environments and* ***OpenGL*** *for rendering* ***3D paths and letter boxes.***
* *Since* ***Google ARCore has higher system requirements****,* ***only ARCore-supported devices can run Xamera AR.***
* *The two primary graphical components:*
  + ***3D Path Rendering*** *– Converts stabilized air-written strokes into virtual paths.*
  + ***3D Letter Box*** *– Displays the inferred text in AR space.*

## 3.2 Testing procedure

#### **Testers Assigned**

* ***Primary Tester:******Soham Naik***
* ***Alternative Tester:******Deniz Acikbas***

*Xamera (Main App) Testing*

1. *Soham will retrieve the latest updates from the Adaptive-HCI GitHub repository while the Xamera project is open in Android Studio.*
2. *In Android Studio, Soham will establish unit testing classes under "kotlin+java/com.developer27.xamera (test)".*
3. *Each unit testing class will evaluate one class related to the project. For example, "MainActivity.kt" will be assessed by "MainActivityUnitTesting.kt".*
4. *During the testing process, Kotlin and built-in Android libraries will be utilized.*
5. *After completing the testing for each class, results will be documented in Alan's Testing Excel file found either in Google Drive or GitHub.*

*Xamera AR Testing*

1. *Soham will pull the latest updates from the Xamera AR GitHub repository while the Xamera AR project is open in Android Studio.*
2. *Soham will create a folder to keep the testing classes.*
3. *Unit tests will be developed in Java and will utilize either built-in JUnit or Android code for testing.*
4. *Once testing for each class is finished, results will be logged in Alan's Testing Excel file located either in Google Drive or GitHub.*

### 3.2.1 Unit test cases

The procedure for unit testing is described for each software component (that will be unit tested) is presented. This section is repeated for all components i.

#### 3.2.1.2 Stubs and/or drivers for Xamera (Main App)

#### 3.2.1.3 Test cases component Xamera (Main App)

1. *Rolling Shutter Camera Frequency*
2. *Recognition of letters A to Z*
3. *Recognition of digits 0 to 9*
4. *Phone Number keying with airwriting*
5. *Airwriting a word*
6. *Accurate letter recognition*
7. *Accurate digit recognition*
8. *AR intent launch*
9. *Video processing*
10. *Zoom-in and zoom-out*
11. *Front - back camera switch*
12. *Line cleaning*

#### 3.2.1.4 Purpose of tests for Xamera (Main App)

1. *Arduino gloves are optimized for a 6 kHz frequency; therefore, Xamera's rolling shutter camera effect should match that frequency.*
2. *The alphabet consists of letters A to Z, and the model is trained accordingly.*
3. *The digits range from 0 to 9, and the model is trained accordingly.*
4. *Since Xamera works on smartphone, therefore it is convenient to call a number with airwriting.*
5. *The user might need to message someone using the airwriting feature.*
6. *Since Xamera is a machine learning based app, the letter recognition must be working accurately.*
7. *Since Xamera is a machine learning based app, the digit recognition must be working accurately.*
8. *Xamera is also an AR app which requires an AR component.*
9. *Video processing plays an important role when airwriting.*
10. *Zoom in and zoom out features are important to be able to use Xamera under various circumstances.*
11. *Not all devices have a back camera, so using the front camera option should be available.*
12. *To not create a mess of lines, it is important that the user should be able to clear the screen.*

#### 3.2.1.5 Expected results for Xamera (Main App)

1. *Make sure the rolling shutter camera actually reaches 6 kHz.*
2. *The user must be able to write letters from A to Z.*
3. *The user must be able to write digits from 0 to 9.*
4. *The user must be able to write telephone numbers by clicking on 'Start Writing' and must click on 'Stop Writing' when finished.*
5. *The user must be able to write words by clicking on 'Start Writing' and then clicking on 'Stop Writing' when finished.*
6. *The Digit Recognition ML Model must be able to recognize digits with an accuracy greater than 90%.*
7. *The Letter Recognition ML Model must be able to recognize letters with an accuracy greater than 90%.*
8. *Xamera must be able to launch the AR component within the app using an intent.*
9. *Video processing features (such as Kalman Filters) should operate as described in Dr. Zhang's paper.*
10. *Zoom in and zoom out functions must be working properly.*
11. *The user must be able to switch between the front and back cameras and perform all video processing functions.*
12. *The screen must be cleared when the user presses on “C” button.*

#### 3.2.1.2 Stubs and/or drivers for Xamera AR

#### 3.2.1.3 Test cases component Xamera AR

1. *3D Path Visualization*
2. *3D Box with Letter Visualization*

#### 3.2.1.4 Purpose of tests for Xamera AR

1. *We would like to simulate the path the user draws in an Augmented Reality environment.*
2. *3D Boxes are the best way to illustrate the recognition.*

#### 3.2.1.5 Expected results for Xamera AR

1. *When a surface is recognized, top any surface to draw a 3D path.*
2. *When a surface is recognized, top any surface to draw a 3D Letter Box.*

### 3.2.2 Integration testing

*Tester:* Alan Raj

*Alternative Tester:* Deniz Acikbas

#### 3.2.2.1 Testing procedure for integration

1. *Deniz will supply Alan with an Android device that has the Xamera Research Preview installed (along with the APK file).*
2. *Alan will set up an Excel sheet either on GitHub or Google Drive.*
3. *The Excel file will include "Closed-Box Testing" and "Unit Testing" pages.*
4. *Under the Closed-Box Testing section, Alan will create columns for "Test No", "Test Name", "Description", and "Result".*
5. *Alan will conduct single letter and digit testing using letters from A to Z and digits from 0 to 9.*
6. *Each letter must yield three outcomes: "Email Writing", "3D Path in Xamera AR", and "3D Letter Box in Xamera AR".*
7. *Each digit must also produce three outcomes: "Phone Call", "3D Path in Xamera AR", and "3D Letter Box in Xamera AR".*
8. *In the following step, Alan will repeat tasks 5 to 7 for testing phone numbers and words.*
9. *Closed box testing will be conducted three times for each letter and digit.*
10. *The task must be finished within the specified completion time.*

#### 3.2.2.2 Stubs and drivers required

1. **Synthetic Gesture Input Stub**
   * **Purpose:** Provides predefined gesture sequences for **testing air-writing recognition** without requiring a real user.
   * **Implementation:** A stored dataset of pre-created bitmaps of characters and digits.
   * **Used For:** Ensuring **text inference** work correctly in an integrated pipeline.
2. **Mock Object Detection API (YOLO)**
   * **Purpose:** Bypasses the need for actual object detection, providing **simulated bounding box** coordinates for LED tracking.
   * **Implementation:** Returns test data with (X, Y, confidence score) to verify that the **path smoothing and text inference modules** process data correctly.
3. **Simulated Email and Phone Call Handling**
   * **Purpose:** Mocks **email writing and phone call actions** triggered by recognized gestures.
   * **Implementation:** Instead of making real calls/emails, a **test logging system** records whether the correct letters/numbers were detected.
4. **Predefined Word and Number Input Stub**
   * **Purpose:** Provides a fixed set of words and numbers for **testing inference accuracy** under different conditions.
   * **Implementation:** Uses a **dataset of manually recorded** test inputs.

#### 3.2.2.3 Test cases and their purpose

| **Test No** | **Test Name** | **Description** | **Purpose** |
| --- | --- | --- | --- |
| 1 | Single Letter Recognition | Test recognition of individual letters (A-Z). | Ensure system correctly identifies letters in air-writing. |
| 2 | Single Digit Recognition | Test recognition of individual digits (0-9). | Ensure system correctly identifies digits in air-writing. |
| 3 | Letter to Email Function | Verify that recognized letters are correctly used for email composition. | Ensure text inference integrates correctly with email writing. |
| 4 | Letter to 3D Path in Xamera AR | Ensure letters create corresponding 3D paths in AR visualization. | Validate integration of air-writing with AR rendering. |
| 5 | Letter to 3D Letter Box in AR | Verify letters generate correct 3D letter box representations in AR. | Ensure proper display of letter-based objects in AR. |
| 6 | Digit to Phone Call Function | Test whether recognized digits correctly map to phone dialing actions. | Validate gesture-to-phone number mapping. |
| 7 | Digit to 3D Path in Xamera AR | Check if digits form correct 3D paths in AR visualization. | Ensure digit-based gesture tracking in AR. |
| 8 | Digit to 3D Letter Box in AR | Validate that recognized digits create appropriate 3D letter boxes. | Ensure AR rendering works for numeric gestures. |
| 9 | Phone Number Recognition | Test air-writing of full phone numbers and system's accuracy. | Validate complete digit sequence recognition. |
| 10 | Word Recognition | Check recognition of full words in air-writing mode. | Ensure accurate multi-character gesture recognition. |
| 11 | Performance Under Time Constraint | Run all tests within the predefined completion time. | Assess system responsiveness and efficiency. |

#### 3.2.2.4 Expected results

| **Test No** | **Expected Result** |
| --- | --- |
| 1 | Each letter (A-Z) is correctly recognized and logged in the test sheet. |
| 2 | Each digit (0-9) is correctly recognized and logged in the test sheet. |
| 3 | Recognized letters correctly appear in email composition fields. |
| 4 | Recognized letters are accurately visualized as 3D paths in Xamera AR. |
| 5 | Recognized letters generate appropriate 3D letter boxes in AR. |
| 6 | Recognized digits correctly trigger simulated phone call entries. |
| 7 | Recognized digits are accurately visualized as 3D paths in Xamera AR. |
| 8 | Recognized digits generate appropriate 3D letter boxes in AR. |
| 9 | Complete phone numbers are correctly recognized and processed. |
| 10 | Full words are accurately recognized and interpreted in air-writing mode. |
| 11 | All tasks are completed within the specified time constraints. |

### 3.2.3 Validation testing

Validation testing ensures that the Adaptive HCI system meets its functional and non-functional requirements under real-world conditions. This phase confirms that the system operates correctly across different environments, devices, and use cases.

#### 3.2.3.1 Testing procedure for validation

The validation testing process follows a structured approach to confirm system correctness, usability, and performance under various conditions.

The following tests will be conducted:

#### **Real-Time Gesture Tracking Accuracy**

* **Procedure:**
  + Track hand gestures using LED-equipped gloves in controlled and real-world settings.
  + Measure accuracy by comparing detected movement paths to ground truth annotations.
* **Test Scenarios:**
  + Different hand speeds and movement styles.
  + Variable lighting conditions (low light, bright light, indirect lighting).
  + Static and dynamic backgrounds.
* **Expected Outcome:**
  + Gesture tracking accuracy ≥ 90% in controlled environments and ≥ 85% in real-world conditions.

#### **Text Conversion Accuracy**

* **Procedure:**
  + Perform air-writing tests where users write predefined words and numbers.
  + Compare system-generated text to expected output using Levenshtein Distance (edit distance).
* **Test Scenarios:**
  + Different stroke speeds and writing angles.
  + Hand tremors and non-uniform handwriting.
* **Expected Outcome:**
  + At least 9 out of 10 characters/digits correctly recognized.

#### **User Experience (UX) and Responsiveness**

* **Procedure:**
  + Evaluate system latency from input to output display.
  + Measure user satisfaction through feedback surveys.
* **Test Scenarios:**
  + Mobile app responsiveness during user interactions.
  + Delays in text recognition and UI updates.
* **Expected Outcome:**
  + Gesture-to-text conversion latency < 50ms.
  + User feedback indicating smooth and intuitive interaction.

#### **Environmental Testing**

* **Procedure:**
  + Test system robustness under different environmental conditions.
* **Test Scenarios:**
  + Bright and dim lighting.
  + Motion interference (background objects, multiple users).
* **Expected Outcome:**
  + System should maintain at least 85% accuracy in non-ideal conditions.

#### **Device Performance Testing**

* **Procedure:**
  + Monitor CPU/GPU utilization and battery consumption.
  + Benchmark system responsiveness using Android Profiler and Perfetto.
* **Test Scenarios:**
  + Extended operation over a 2-hour period.
  + Simultaneous execution of gesture tracking and text inference.
* **Expected Outcome:**
  + The system maintains at least 30 FPS and does not overheat the device.

#### 3.2.3.3 Expected results

* The system correctly detects hand movements, stabilizes strokes, and converts gestures to text with high accuracy.
* The application responds smoothly to user interactions with minimal latency.
* The system functions effectively in various lighting and motion conditions.
* Performance benchmarks meet predefined criteria without excessive battery drain or overheating.

#### 3.2.3.4 Pass/fail criterion for all validation tests

A test will be considered successful if:

* Gesture tracking accuracy is ≥ 90% in controlled conditions and ≥ 85% in real-world conditions.
* Text inference accuracy is ≥ 90% with proper character/digit recognition.
* Frame processing time is ≤ 50ms per frame, ensuring real-time performance.
* The system maintains an FPS of at least 30 for smooth rendering.
* The application remains stable for 2+ hours of continuous use without critical failures.

If any of these criteria are not met, the test will be marked a fail, and issues will be logged for debugging and optimization.

### 3.2.4 High-order testing (a.k.a. System Testing)

High-order testing evaluates the robustness, security, and long-term stability of the Adaptive HCI system under various stress conditions.

#### 3.2.4.1 Recovery testing

#### **Objective:**

Assess the system’s ability to recover from failures such as crashes, unexpected interruptions, and power loss.

#### **Test Procedure:**

* Simulate unexpected shutdowns and verify data integrity after restart.
* Manually terminate the app during operation and check if it resumes correctly.
* Test for error recovery in case of lost gesture tracking.

#### **Expected Outcome:**

* The system should resume from the last known state after a crash.
* No corrupted data or unexpected errors should occur after restart.

#### 3.2.4.2 Security testing

#### **Objective:**

Ensure that the system operates securely without external data leaks or unauthorized access.

#### **Test Procedure:**

* Verify that all data processing occurs locally on the device with no cloud dependency.
* Perform penetration testing to check for vulnerabilities in the app.
* Confirm that user gestures are not stored or transmitted externally.

#### **Expected Outcome:**

* The system must not send or store user data externally.
* No unauthorized access should be possible.

#### 3.2.4.3 Stress testing

#### **Objective:**

Determine how the system handles prolonged use and high workloads.

#### **Test Procedure:**

* Run the application continuously for 4+ hours and monitor memory usage.
* Introduce excessive hand movements and assess tracking stability.
* Test with multiple users writing simultaneously.

#### **Expected Outcome:**

* No memory leaks or excessive lag should occur.
* Tracking should remain accurate under high user activity.

#### 3.2.4.4 Performance testing

#### **Objective:**

Measure system efficiency in terms of speed, responsiveness, and resource utilization.

#### **Test Procedure:**

* Measure FPS, CPU/GPU usage, and latency.
* Run performance benchmarks on different hardware (Motorola G Play vs. Google Pixel 8A).

#### **Expected Outcome:**

* The system should process frames under 50ms with an FPS of 30+.
* The application should remain responsive under normal conditions.

#### 3.2.4.5 Alpha/beta testing

#### **Objective:**

Gather feedback from real users to evaluate usability and overall system performance.

#### **Test Procedure:**

* Conduct alpha testing internally with the development team.
* Beta test with external users to collect feedback on usability and reliability.

#### **Expected Outcome:**

* Users should report positive experiences with system responsiveness.
* No major usability issues should be identified.

#### 3.2.4.6 Pass/fail criterion for all validation tests

A test will be considered successful if:

* The system successfully recovers from crashes without data loss.
* No security vulnerabilities or external data leaks are detected.
* The application runs continuously for 4+ hours without performance degradation.
* FPS remains ≥ 30, and processing time per frame stays ≤ 50ms.

Any test that fails to meet these conditions will be logged for further debugging and optimization.

## 3.3 Testing resources and staffing

**Testing Resources**

The Adaptive HCI system will require various resources to conduct thorough testing, including hardware, software, and testing tools to ensure the system meets its performance, accuracy, and reliability requirements.

1. **Hardware Resources:**
   1. **Test Devices:**
      1. Motorola G Play – A budget smartphone used for testing performance on lower-end hardware.
      2. Google Pixel 8A – A high-end smartphone used for evaluating system performance under optimal conditions.
   2. **Input Devices:**
      1. LED-Equipped Gloves – Used for tracking real-time hand gestures and air-writing strokes.
      2. Smartphone Rolling Shutter Cameras– Used for capturing high-speed hand movements and processing real-time videos.
   3. **Visualization Equipment:**
      1. AR/VR headsets – Used for testing immersive 3D gesture visualization in augmented reality environments.
2. **Software Resources:**
   1. Mobile Application (Xamera) – The main test environment for real-time video processing, gesture tracking, and text inference.
   2. Xamera AR (Augmented Reality Module) – Responsible for 3D visualization and gesture rendering.
   3. Android Studio – Primary development and testing environment for testing and debugging.
   4. GitHub – Used for version control and issue tracking.
3. **Testing Tools:**
   1. **Development and Debugging:**
      1. Android Debug Bridge (ADB) – Used for debugging and logging software performance.
      2. Android Profiler – Analyzes memory usage, frame rate (FPS), and CPU/GPU performance.
   2. **Object Detection and Image Processing:**
      1. TensorFlow Lite – Validates the performance of deep learning models used for text inference.
      2. OpenCV – Assesses the accuracy of image processing techniques applied to gesture tracking.
   3. **Performance Monitoring and Benchmarking:**
      1. Perfetto – Used to analyze CPU and GPU load, frame rendering speed, and system performance.
      2. Systrace (Android) – Evaluates application responsiveness and identifies performance issues.
   4. **Bug Tracking and Issue Management:**
      1. GitHub Issues – Tracks defects, categorizes them by severity, and assigns them for resolution.
      2. Excel Logs – Documents test cases, execution results, and system performance benchmarks.

**Staffing**

Each member of the development team is assigned a specific role in the testing process to ensure validation and performance analysis.

| Role | Responsibilities | Assigned Personnel |
| --- | --- | --- |
| Software Engineers | Develop, debug, and optimize software components. | Deniz Acikbas, Soham Naik, Alan Raj, and Zaynab Mourtada |
| QA Testers | Responsible for unit testing, integration testing, and debugging. | Soham Naik, Deniz Acikbas |
| Integration Tester | Conduct integration testing between modules and validate system-wide interactions. | Alan Raj |
| Machine Learning Engineers | Evaluate gesture tracking and text inference models for accuracy and performance. | Deniz Acikbas, Soham Naik, Zaynab Mourtada |
| Project Supervisor | Oversee test planning, milestone tracking, and final validation of system performance. | Xiao Zhang |

## 3.4 Test work products

The following test work products will be generated during the testing process:

* **Test Cases Document** – List all test cases for unit, integration, validation, and high-order testing, including test inputs, expected outputs, and pass/fail criteria.
* **Bug Reports** – Logs software defects with details such as severity, steps to reproduce, observed behavior, and resolution status. Managed through GitHub Issues.
* **Performance Reports** – Evaluates FPS, latency, CPU/GPU usage, and memory consumption to identify system issues and areas for improvement.
* **Accuracy Metrics** – Assesses gesture tracking precision, text inference accuracy, and path smoothing effectiveness using statistical benchmarks.
* **Test Summary Reports** – Consolidates results from all testing phases, highlighting success rates, critical issues, and optimization recommendations.
* **Test Scripts and Logs** – Includes automated test scripts and execution logs stored in Google Drive/GitHub for debugging and future reference.
* **User Testing Feedback** – Collects usability and performance feedback from test users to improve system responsiveness and user experience.

All test work products will be stored in Google Drive and GitHub for documentation and tracking purposes.

## 3.5 Test record keeping and test log

All test results, logs, and reports will be systematically recorded to ensure thorough documentation and analysis for the testing process. The records will be maintained in Google Drive and GitHub, allowing easy access, version control, and collaboration.

**Test Record Keeping**

The following mechanism will be used for storing and evaluating test results:

* **Test Case Execution Logs** – Documents test scenarios, expected vs. actual results, and pass/fail status.
* **Bug Tracking System** – All defects will be logged in GitHub Issues, categorized by severity, and assigned for resolution.
* **Performance Logs** – Captures FPS, latency, CPU/GPU load, and memory usage for benchmarking and optimization.
* **System Behavior Logs** – Debugging logs recorded via Android Debug Bridge (ADB) to analyze software performance.

**Test Log Structure**

Each test session will be documented using the following structured format:

| Date & Time | Test Name | Module Tested | Tester | Device Used | Expected Result | Actual Result | Status (Pass/Fail) | Bug ID |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| MM/DD/YYYY | Object Detection Accuracy | YOLO-Based Tracking | Soham Naik | Google Pixel 8A | 95%+ Detection | 93% Detection | Fail | BUG-01 |
| MM/DD/YYYY | Text Inference | CNN Digit Recognizer | Zaynab Mourtada | Motorola G Play | 90%+ Accuracy | 89%+ Accuracy | Fail | BUG-02 |
| MM/DD/YYYY | Frame Rate Test | UI Performance | Alan Raj | Motorola G Play | 30+ FPS | 32 FPS | Pass | N/A |

Regular updates to the test log will ensure a chronological record of all testing activities, making it easier to track progress, identify recurring problems, and refine system performance.