TEST CASE SPECIFICATION

Assistive Vision

Version 3.0

Written by:
Venkat Yenduri, Pierre Tawfik,
Sukrut Nadigotti, Sharefa Alshaary

Revision History

Date	Description	Authors	Comments
03/17/2025	Version 1	Venkat Yenduri, Pierre Tawfik, Sukrut Nadigotti, Sharefa Alshaary	Initial submission of the test case specification document.
03/24/2025	Version 2	Venkat Yenduri, Pierre Tawfik, Sukrut Nadigotti, Sharefa Alshaary	Secondary submission of the test case specification document.
03/31/2025	Version 3	Venkat Yenduri, Pierre Tawfik, Sukrut Nadigotti, Sharefa Alshaary	Final submission of the test case specification document based on feedback.

Table of Contents

1. Introduction

1.1 Introduction

This document outlines the testing plan for the Assistive Vision project. It serves as a comprehensive guide that details the overall approach to testing, the methods to be used, and the criteria for assessing the application's performance and functionality. The document is organized to provide clear insight into the testing process by explaining how every component of the system, including real time object detection, voice command processing, audio feedback, and haptic feedback, will be rigorously evaluated. Its main objective is to confirm that the application not only meets the established requirements but also delivers a robust and accessible user experience.

1.2 Purpose

The testing plan is designed to define and document the strategies for verifying that the Assistive Vision application performs as expected under a variety of conditions. By presenting a systematic approach to testing, this plan aims to identify and rectify any issues before the application is released. The focus is on ensuring that the core functionalities such as the integration of CoreML for object detection and the handling of voice commands are reliable and efficient. The ultimate purpose is to provide assurance to stakeholders that the system is well prepared for deployment and meets both technical specifications and user centric requirements.

1.3 Objectives

The objectives of this testing plan are multifaceted. The first objective is to validate that all primary features operate correctly and efficiently while confirming that the system meets performance benchmarks in real time processing and responsiveness. The second objective is to verify that the accessibility features function seamlessly so that visually impaired users have an intuitive experience. The third objective is to conduct comprehensive integration testing to ensure that various modules, such as video capture, machine learning, and user interface components, work together harmoniously. Additionally, the testing process is intended to uncover any potential defects in order to facilitate prompt corrections and continuous improvements in the system.

1.4 Scope

This testing plan covers functional, performance, and usability assessments of the Assistive Vision application. Functional testing will examine every aspect of the application's behavior including video input, object detection accuracy, and the execution of voice commands. Performance testing will involve measuring critical parameters such as frame rate, latency, and

resource consumption to ensure optimal operation across different iOS devices. Usability testing will focus on evaluating whether the interface and feedback mechanisms are both intuitive and effective for the target user group. The plan excludes any non critical future enhancements and external systems that are not directly managed by the Assistive Vision project.

1.5 Intended Audience

The document is intended for quality assurance teams, developers, project stakeholders, and accessibility experts. Quality assurance professionals will use the testing plan as a detailed guide for executing test cases and verifying system integrity. Developers will refer to the plan to better understand the testing requirements and to integrate necessary improvements. Project stakeholders will gain assurance from the methodically tested product, and accessibility experts will evaluate whether the application meets the high standards required for users with visual impairments. Overall, this plan is designed to clearly communicate the testing strategy to all parties involved, ensuring that every aspect of the application is thoroughly examined before release.

2. Test Objectives

2.1 Functional Requirements

The foremost objective is to ensure that all functional aspects of the Assistive Vision application meet the design specifications. This involves confirming that each feature, from real time object detection and voice command processing to user interface interactions and audio feedback, operates as intended. Every function is rigorously examined to verify that it delivers a reliable and accessible experience for visually impaired users while meeting the intended user needs.

2.2 System Performance

An equally important objective is to validate that the system performs optimally under various conditions. This includes assessing key performance indicators such as frame rate, responsiveness, resource consumption, and latency on different iOS devices. The testing process simulates diverse real world scenarios to guarantee that the application maintains consistent performance and efficiency, even under high load or prolonged usage.

2.3 Security and Reliability

Finally, the testing plan aims to ensure that the application is secure and reliable. This objective focuses on identifying any potential vulnerabilities and verifying that the system can sustain continuous operation without failures. Emphasis is placed on robust error handling, data

integrity, and overall system stability, providing stakeholders with the assurance that the application is both safe and dependable for everyday use.

3. Test Strategy

3.1 Testing Approach

The Assistive Vision project primarily employs a white box testing approach, focusing on validating the internal structure, logic, and integration of key components such as CoreML models, video capture, and audio feedback modules. This ensures that data flows correctly between components, model inferences are accurate, and system performance meets expected benchmarks. Unit tests and integration tests will be used to verify algorithm correctness, model execution, and efficient resource utilization.

3.2 Testing Levels

The testing process is organized into multiple levels. Unit testing is conducted to verify the functionality of individual components in isolation, ensuring that each function performs correctly on its own. Integration testing follows, concentrating on the interaction between different modules, such as how the video capture component works in tandem with machine learning predictions and audio output. System testing evaluates the application as a whole, checking for performance benchmarks, stability, and responsiveness under various conditions. Finally, acceptance testing confirms that the application meets the requirements and expectations of its intended users, particularly addressing the needs of visually impaired individuals.

3.3 Special Techniques and Tools

A range of specialized techniques and tools is employed to support the testing process. Unit testing is performed using the XCTest framework available in Xcode, which allows for automated testing of individual functions and components. For integration and system testing, simulated inputs and controlled environments replicate real world conditions, ensuring that the application can handle diverse scenarios. Performance profiling tools, such as Instruments, are used to monitor key parameters like frame rate, resource consumption, and latency across different iOS devices. These techniques and tools collectively ensure that the Assistive Vision project is thoroughly tested for functionality, performance, security, and reliability.

4. Test Scope

4.1 Scope

The scope of testing for the Assistive Vision project encompasses a comprehensive evaluation of functional performance, system integration, user experience, and overall reliability.

Functional testing will examine every aspect of the application, from video capture and object detection to voice command processing and feedback mechanisms, ensuring that each component meets the defined specifications. Performance testing will assess the system's responsiveness, frame rate, latency, and resource consumption across various iOS devices and real world conditions, thereby confirming its ability to operate efficiently under different usage scenarios.

4.2 Usability Testing

Usability testing is also an integral part of the scope, focusing on the accessibility of the user interface and the effectiveness of both audio and haptic feedback, with special attention to the needs of visually impaired users. In addition, integration testing will verify that the individual modules interact seamlessly, while acceptance testing will involve key stakeholders to validate that the application meets user expectations and project requirements. The testing process does not cover non critical future enhancements or external systems that are not directly managed by the Assistive Vision project.

5. Test Deliverables

5.1 Test Plans

Test Plans serve as a comprehensive guide that outlines the overall strategy, scope, objectives, and schedule for the testing activities of the Assistive Vision project. They detail the methodologies, tools, and resources to be used and define the criteria for success. This deliverable establishes the framework that directs all subsequent testing efforts, ensuring that the process is systematic, well documented, and closely aligned with the project's goals.

5.2 Test Cases

Test Cases are detailed descriptions of individual scenarios designed to validate the application’s functionality. Each case specifies the conditions, inputs, and expected outcomes for a particular feature or use case. Derived from the project requirements, these cases provide a repeatable measure for verifying that every component of the application behaves as intended under various conditions.

5.3 Test Scripts

Test Scripts consist of step-by-step instructions used to execute the test cases. Whether developed for manual or automated testing, these scripts simulate real world scenarios to efficiently and accurately assess the system’s behavior. The use of test scripts enhances the consistency of the testing process while minimizing the potential for human error.

5.4 Test Data

Test Data includes the sets of input values, configuration settings, and environmental variables that are used during testing. This deliverable encompasses both typical scenarios and edge cases, ensuring that every aspect of the system is thoroughly examined, including its error handling and performance under stress. High-quality test data is essential for obtaining meaningful and accurate results throughout the testing cycle.

5.5 Test Reports

Test Reports document the outcomes of the testing process by providing a structured summary of the tests executed, recording any defects or anomalies, and evaluating the system's adherence to the acceptance criteria. These reports facilitate clear communication among developers, testers, and stakeholders, offering insights into the overall quality and readiness of the application for release.

6. Schedule

All the test cases were implemented early and throughout the development process to ensure that the system would be able to function to the best of its ability. Below is a list of the test cases next to the team member name who tested and confirmed that the functionality works.

Test Case ID	Name	Deadline Date
U1	Venkat Yenduri	03-21-25
U2	Sukrut Nadigotti	03-22-25
U3	Pierre Tawfik	03-23-25
U4	Sharefa Alshaary	03-24-25
U5	Venkat Yenduri	03-25-25
U6	Sukrut Nadigotti	03-26-25
U7	Pierre Tawfik	03-27-25
U8	Sharefa Alshaary	03-28-25
U9	Venkat Yenduri	03-29-25
U10	Sukrut Nadigotti	03-30-25
U11	Pierre Tawfik	03-31-25
U12	Sharefa Alshaary	04-01-25

U13	Venkat Yenduri	04-02-25
U14	Sukrut Nadigotti	04-03-25
U15	Pierre Tawfik	04-04-25
U16	Sharefa Alshaary	04-05-25
U17	Venkat Yenduri	04-06-25
U18	Sukrut Nadigotti	03-21-25
U19	Pierre Tawfik	03-22-25
U20	Sharefa Alshaary	03-23-25
U21	Venkat Yenduri	03-24-25
U22	Sukrut Nadigotti	03-25-25
U23	Pierre Tawfik	03-26-25
I1	Sharefa Alshaary	03-27-25
I2	Venkat Yenduri	03-28-25
I3	Sukrut Nadigotti	03-29-25
I4	Pierre Tawfik	03-30-25
I5	Sharefa Alshaary	03-31-25
S1	Venkat Yenduri	04-01-25
S2	Sukrut Nadigotti	04-02-25
S3	Pierre Tawfik	04-03-25
U24	Sharefa Alshaary	04-04-25
U25	Venkat Yenduri	04-05-25
U26	Sukrut Nadigotti	04-06-25
U27	Pierre Tawfik	03-21-25
S4	Sharefa Alshaary	03-22-25
S5	Venkat Yenduri	03-23-25
U30	Sukrut Nadigotti	03-24-25
U31	Pierre Tawfik	03-25-25
U32	Sharefa Alshaary	03-26-25
U33	Venkat Yenduri	03-27-25
U34	Sukrut Nadigotti	03-28-25

U35	Pierre Tawfik	03-29-25
U36	Sharefa Alshaary	03-30-25
I6	Venkat Yenduri	03-31-25
I7	Sukrut Nadigotti	04-01-25
U39	Pierre Tawfik	04-02-25
U40	Sharefa Alshaary	04-03-25
I8	Venkat Yenduri	04-04-25
U42	Sukrut Nadigotti	04-05-25
U43	Pierre Tawfik	04-06-25
U44	Sharefa Alshaary	03-21-25
I9	Venkat Yenduri	03-22-25
I10	Sukrut Nadigotti	03-23-25
S6	Pierre Tawfik	03-24-25
U48	Sharefa Alshaary	03-25-25
U49	Venkat Yenduri	03-26-25
U50	Sukrut Nadigotti	03-27-25
S7	Pierre Tawfik	03-28-25
S8	Sharefa Alshaary	03-29-25
S9	Venkat Yenduri	03-30-25
S10	Sukrut Nadigotti	03-31-25
S11	Pierre Tawfik	04-01-25
S12	Sharefa Alshaary	04-02-25
S13	Venkat Yenduri	04-03-25

7. Test Environment

7.1 Hardware Environment

The Assistive Vision project requires testing on a variety of iOS devices to ensure compatibility across different hardware configurations. Testing should include iPhone models ranging from iPhone 11 to iPhone 15 Pro, covering different processing power levels, camera capabilities, and hardware-based machine learning optimizations. Devices must feature both

standard wide-angle cameras and depth-sensing LiDAR cameras (available in Pro models) to assess object detection accuracy in various conditions. Performance requirements dictate that each device should have a minimum of an A13 Bionic chip (or later) and at least 4GB of RAM to support real-time video capture, CoreML-based object detection, and audio processing. Additionally, the devices should include stereo speakers, haptic feedback engines, and VoiceOver support to evaluate accessibility features effectively.

7.2 Software Environment

The test environment must support multiple iOS versions and include essential development tools to validate compatibility and performance. The primary target operating system for testing is iOS 17.3, with additional verification on iOS 16.x and 15.x to ensure backward compatibility with older devices. Development and testing will be conducted using Xcode 15.2 or later, which provides necessary debugging and profiling tools. The XCTest framework will be used for unit testing, while Instruments (part of Xcode) will assist with performance profiling and memory analysis. Automated UI testing will be performed using Appium (v2.0.0) to verify interaction consistency across different iOS versions, and TestFlight will facilitate beta testing. The application relies on CoreML 3.0 for on-device object detection, the Vision Framework for image processing and object tracking, and AVFoundation for real-time audio playback and speech synthesis.

7.3 Network Environment

Although the core application functions locally, some features require network connectivity, such as remote logging and over-the-air updates. To ensure reliability in different conditions, the test environment must simulate various network scenarios. Testing should include high-speed Wi-Fi (5GHz, 1Gbps+) for optimal performance, 4G LTE and 5G to assess mobile network reliability, and low-bandwidth conditions (3G, 500kbps) to evaluate performance under limited connectivity. Additionally, an offline mode test must be conducted to confirm that core functionality remains unaffected without an internet connection. If applicable, backend services should be tested, including cloud-based remote logging APIs (AWS/GCP) for debugging and analytics and OTA update mechanisms via TestFlight to validate seamless software updates.

7.4 Real-World Considerations

To ensure a comprehensive evaluation, the test environment must closely replicate real-world conditions. Object detection accuracy should be tested in varied lighting conditions, including bright daylight, dim indoor lighting, and complete darkness. Audio-based feedback and speech recognition must be assessed in environments with different ambient noise levels, such as quiet rooms, busy streets, and public spaces, to verify clarity and responsiveness. Testing should also account for user interaction variability, including different grip styles, movement speeds,

and device orientations, ensuring the application remains effective under diverse usage patterns. Additionally, test cases should include a broad range of object shapes, sizes, and textures to evaluate robustness in real-world scenarios.

8. Test Entry and Exit Criteria

8.1 Entry Criteria

The testing phase for the Assistive Vision project begins only after all preliminary development and integration tests have been successfully completed. Testing is initiated when the application meets specific entry criteria. First, all individual modules must pass unit tests, ensuring they function as expected in isolation. The application must also meet predefined functional requirements as outlined in the software requirements specification (SRS). A fully operational test environment, including necessary hardware, software, and network configurations, must be set up before testing begins. Additionally, all test plans, test cases, and expected outcomes must be finalized and approved. Any critical defects identified during earlier development phases must be resolved through a structured bug triage process. Finally, the required test datasets must be prepared, validated, and readily accessible for execution.

8.2 Exit Criteria

Testing concludes when all defined test cases have been executed, and the application meets the required performance and functional specifications. Exit criteria include the successful execution of all planned test cases, ensuring comprehensive coverage of functional, performance, security, and usability aspects. All high-priority and critical defects must be fixed, verified, and closed, while lower-priority issues are documented for future iterations. The application must meet the performance thresholds defined in the test plan, with no major security vulnerabilities remaining unresolved. Final test reports are reviewed and approved by key stakeholders, including the development and QA teams. Once all exit conditions are met, formal approval is obtained to transition the project to the release phase.

Testing may be suspended if critical system failures occur, blocking further execution. If a major defect prevents meaningful progress, testing will pause until the issue is resolved. Once fixes are implemented, retesting will be performed to verify defect resolution, followed by regression testing to ensure no unintended side effects. Testing will only resume when the system is stable enough for continued evaluation.

9. Test Pass and Fail Criteria

9.1 Pass Criteria

For a test to be considered successful, the Assistive Vision application must accurately process all defined input types and generate the intended outputs. Every function, whether it is real time object detection, voice command processing, or user interface interaction, is expected to operate with precision and in full accordance with stakeholder requirements. The system should demonstrate consistent performance, with acceptable response times and resource utilization, while gracefully handling any error conditions. The application must meet all established performance benchmarks and deliver a user experience that is both robust and reliable.

9.2 Fail Criteria

Conversely, a test is deemed to have failed if any component of the application produces unexpected results or deviates from its intended behavior. Failure is evident when the application is unable to correctly manage specific inputs or outputs, if error conditions are not handled appropriately, or if the system experiences crashes or significant delays in response times. Any shortcomings that compromise core functionality or degrade the overall user experience will be classified as a failure, preventing the project from advancing to the release phase.

10. Test Suspension and Resumption Criteria

10.1 Suspension Criteria

Testing will be suspended immediately if any critical failure is detected that significantly impacts the functionality of the Assistive Vision application. This includes instances where a test case produces unexpected outcomes, system crashes occur, or severe performance issues are identified that prevent reliable test execution. When such failures arise, all further testing activities will be halted to prevent additional complications and to allow for a thorough investigation of the underlying issues.

10.2 Resumption Criteria

Testing will resume only after the root cause of the failure has been identified and effectively addressed. This may involve updating the system software or revising the test cases to ensure accurate reflection of the intended functionality. Once corrective actions have been verified and a stable build of the application is confirmed, testing can restart. Resumption is contingent upon obtaining approval from the quality assurance team, ensuring that the system now meets the predefined requirements and is ready for continued evaluation.

11. Test Design and Execution

11.1 Testing Methodology

The Assistive Vision project employs a combined testing approach that leverages both automated and manual testing methods. Automated testing is used to ensure rapid, repeatable verification of core functionalities such as real time object detection, voice command processing, and UI interactions. In parallel, manual testing is applied to assess aspects that require subjective judgment, including usability, accessibility, and end-to-end user experience. This mixed methodology ensures comprehensive test coverage across all functional and nonfunctional requirements.

11.2 Automated Testing

Automated tests form the backbone of the technical verification process. Unit tests and integration tests are implemented using the XCTest framework in Xcode, allowing for the continuous execution of test scripts with each code change. These tests verify that individual components, such as CoreML model integration and video capture functionality, operate correctly and efficiently. Additionally, performance testing is conducted using tools like Instruments to monitor key parameters such as frame rate, latency, and resource consumption. This automated suite provides rapid feedback and helps to identify regression issues early in the development cycle.

11.3 Manual Testing

Manual testing focuses on the qualitative aspects of the application that cannot be fully captured through automation. Testers simulate real world usage scenarios to evaluate the clarity and accuracy of audio feedback, the responsiveness of haptic feedback, and the overall ease of use for visually impaired users. This testing is particularly important for validating the voice command processing under diverse environmental conditions and ensuring that the user interface meets accessibility standards. Manual testing sessions are scheduled at critical milestones, including system integration and user acceptance phases, to provide comprehensive insights into the user experience.

11.4 Test Execution Plan

The test execution plan outlines the schedule, resources, and responsibilities for the testing phase. Automated tests are integrated into the continuous integration process and executed regularly throughout development. Manual tests are conducted at predefined milestones, focusing on high-risk areas and critical functionalities. Test cases are prioritized based on their impact on system stability and user experience. Detailed reporting procedures are

in place to document test outcomes, track defects, and verify that corrective actions have been implemented. This coordinated approach ensures that all testing activities are aligned with project objectives, and that any issues are promptly addressed before the final release.

12. Test Data

12.1 Test Data Management

Effective management of test data is essential to ensure comprehensive coverage of the application's functionality. In the Assistive Vision project, test data will be generated through a combination of manual collection, automated scripts, and synthetic data generation techniques. Manually curated datasets will include real-world inputs captured using iOS devices under various conditions, ensuring practical relevance. Automated scripts will generate test cases that simulate diverse scenarios, including controlled edge cases and stress testing conditions. Additionally, synthetic data will be created using machine learning models to expand the dataset for rare or difficult-to-capture scenarios.

For testing the real-time model, a set of test images representing key objects will be used. These images will be stored as PNG files in a designated folder on GitHub, ensuring accessibility and consistency in evaluation.

Test data will be maintained in a dedicated repository with version control to track updates, modifications, and additions. Each dataset will be clearly documented, specifying its source, format, intended usage, and generation method, ensuring consistency and traceability throughout the testing process. Access to the repository will be restricted to authorized team members, with periodic audits conducted to confirm data relevance and accuracy.

12.2 Defect Management

The defect management process is structured to promptly capture, document, and resolve any issues discovered during testing. When a defect is identified, it will be recorded in a dedicated defect tracking system, documenting key details such as steps to reproduce the issue, expected vs. actual outcomes, severity level, and supporting evidence (e.g., logs, screenshots, or video recordings).

Defects will be categorized based on impact and priority, ensuring that critical issues are addressed first. The defect tracking system will facilitate status updates and monitor the progress of issue resolution. Regular review meetings will be held where the development and quality

assurance teams collaborate to assess defect status, validate fixes, and ensure thorough regression testing before closing issues. This structured approach ensures that the final product is stable, reliable, and aligned with stakeholder expectations.

13. Risk Analysis

13.1 Potential Risks

The Assistive Vision project faces risks that could impact both the project timeline and test execution feasibility. These risks fall into two main categories: project risks and execution risks.

Project risks include incomplete or inaccurate documentation, which can make it difficult for testers to validate key functionalities effectively. Resource constraints, such as limited access to specific iOS devices, network conditions, or testing tools, may delay test execution or reduce coverage. Testing environment instability, caused by misconfigured environments, software version mismatches, or unavailable backend services, could disrupt testing schedules. Additionally, delays in fixing critical defects may block testing progress, leading to missed deadlines.

Execution risks involve challenges in testing specific functionalities, particularly hardware-dependent features like depth sensing with LiDAR, which may be difficult to verify on non-Pro iPhone models. Unreliable test data or inaccurate simulations can result in improper validation of detection algorithms and voice processing models. Performance bottlenecks, such as unoptimized code causing unexpected slowdowns, high CPU/GPU usage, or excessive memory consumption, may affect test reliability. Lastly, tester uncertainty regarding edge cases, such as detecting small or fast-moving objects, may make validation difficult due to unclear expected outcomes.

13.2 Mitigation Strategies

To ensure risks are proactively managed, the Assistive Vision project will implement a series of mitigation strategies addressing both project and execution risks.

Project risk mitigation begins with ensuring complete and accurate documentation. Regular documentation reviews and validation sessions will be conducted before test execution to confirm that test plans, system requirements, and expected outcomes are well-defined. To expand device and resource availability, a device-sharing system will be established within the

team, and cloud-based testing solutions such as TestFlight and BrowserStack will be considered for testing on unavailable hardware. Stabilizing the test environment will involve pre-configuring and validating all necessary hardware, software versions, and network conditions before execution to ensure consistency. Additionally, a structured triage system will categorize defects by severity, with high-priority issues assigned strict resolution deadlines to prevent delays.

Execution risk mitigation strategies include alternative testing methods for hardware limitations, such as using simulated environments and mock testing frameworks when direct testing of iOS features like LiDAR-based depth sensing is not possible. To enhance test data reliability, synthetic datasets will be continuously refined to match real-world conditions, with manual validation conducted alongside automated tests to confirm accuracy. Performance monitoring and optimization will be achieved through tools like Xcode Instruments and CPU/GPU profiling to detect and resolve performance issues early in the testing process.

14. Roles and Responsibilities

The testing team for the Assistive Vision project consists of internal developers, external testers, and Graduate Teaching Assistants (GTAs) who contribute to both manual and automated testing efforts. Testing will be conducted using white-box testing (for code-level verification). Automated tests will run continuously during development, while manual testing will be scheduled at key milestones before prototype releases. The primary testing tools include XCTest for unit and UI tests.

14.1 Test Manager

The Test Manager oversees the entire testing process, ensuring that all activities align with project objectives, timelines, and quality standards. This role is responsible for test planning, resource coordination, scheduling test execution, and managing communication between internal and external testers. The Test Manager monitors progress, conducts risk assessments, and ensures that white-box testing approaches are executed effectively. Additionally, this role verifies that automated test cases are regularly updated and executed alongside manual testing phases.

14.2 White-Box Test Engineer

The White-Box Test Engineer is responsible for code-level verification, ensuring the internal logic and system structure are robust and optimized. This role develops and executes unit tests, integration tests, and automated scripts to validate code quality. The White-Box Test Engineer works closely with developers to identify potential logic errors, edge case failures, and performance inefficiencies. Testing tools such as XCTest, Appium, and custom test harnesses

will be used to ensure proper test coverage across various iOS models and operating system versions.

14.4 Defect Manager

The Defect Manager coordinates the process of defect identification, tracking, and resolution, ensuring that all reported issues are documented, prioritized, and communicated efficiently. This role maintains the defect tracking system, assigning severity levels and tracking progress to ensure timely resolution. The Defect Manager also organizes regular review sessions where testers, developers, and stakeholders discuss high-priority bugs, verification of fixes, and regression testing requirements before each release.

15. Functional Test Cases

1. U_VideoCapture initialization

ID	U1
Test Case	U_VideoCapture initialization
Test Type	Unit
Item	Video capture feature
Pre-Conditions	None
Test Steps	1. Open the part of the app that uses video capture. 2. Check the video capture settings.
Expected Results	The video capture element appears, and it shows that it is not set up and not running.
Priority	Medium
Pass/Fail Criteria	The test passes if the video capture shows it is not configured and is not running.
Executed By	QA Tester
Pass/Fail Status	Passed

2. U_VideoCapture setup

ID	U2
Test Case	U_VideoCapture setup
Test Type	Unit
Item	Video capture feature
Pre-Conditions	None
Test Steps	1. Press the “set up” button or run the setup process. 2. Check the outcome after setup.
Expected Results	The video capture shows that it is set up, and the process confirms success.
Priority	Medium
Pass/Fail Criteria	The test passes if the setup confirms success and marks the element as configured.
Executed By	QA Tester
Pass/Fail Status	Passed

3. U_VideoCapture start and stop

ID	U3
Test Case	U_VideoCapture start and stop
Test Type	Unit
Item	Video capture feature
Pre-Conditions	Video capture is set up
Test Steps	1. Verify the video capture is not running. 2. Press the “start” control. 3. Press the “stop” control.

Expected Results	The video capture starts when “start” is pressed and stops when “stop” is pressed.
Priority	Medium
Pass/Fail Criteria	The test passes if the running status changes correctly after start and stop actions.
Executed By	QA Tester
Pass/Fail Status	Passed

4. U_VideoCapture delegate

ID	**U4**
Test Case	U_VideoCapture delegate
Test Type	Unit
Item	Video capture feature with a delegate
Pre-Conditions	A delegate (person responsible for noticing events) is set
Test Steps	1. Assign the delegate to the video capture feature. 2. Simulate a video frame capture.
Expected Results	The delegate notices that a video frame has been captured.
Priority	Medium
Pass/Fail Criteria	The test passes if the delegate confirms it received a frame.
Executed By	QA Tester
Pass/Fail Status	Passed

5. U_Detector initialization

ID	U5
Test Case	U_Detector initialization
Test Type	Unit
Item	Object detection feature
Pre-Conditions	None
Test Steps	1. Open the object detector. 2. Check its starting state.
Expected Results	The detector shows no objects found and its confidence level is zero.
Priority	Medium
Pass/Fail Criteria	The test passes if the detector is empty and its confidence is 0.0.
Executed By	QA Tester
Pass/Fail Status	Passed

6. U_Detector detection

ID	U6
Test Case	U_Detector detection
Test Type	Unit
Item	Object detection feature
Pre-Conditions	None
Test Steps	1. Set the detector so that it expects two items (for example, “door” and “person”). 2. Ask it to detect on a sample image.
Expected	The detector shows that it has found “door”

Results	and “person”.
Priority	Medium
Pass/Fail Criteria	The test passes if the detector returns the expected objects.
Executed By	QA Tester
Pass/Fail Status	Passed

7. U_DoorsDetector initialization

ID	**U7**
Test Case	U_DoorsDetector initialization
Test Type	Unit
Item	Door detection feature
Pre-Conditions	None
Test Steps	1. Open the door detection feature. 2. Check its starting status.
Expected Results	The door detector shows no doors found and a confidence level of zero.
Priority	Medium
Pass/Fail Criteria	The test passes if the door detector’s default state is empty.
Executed By	QA Tester
Pass/Fail Status	Passed

8. U_DoorsDetector detection

ID	**U8**
Test Case	U_DoorsDetector detection

Test Type	Unit
Item	Door detection feature
Pre-Conditions	None
Test Steps	1. Set the door detector to expect “door” and “entrance”. 2. Ask it to scan a sample image.
Expected Results	The door detector shows “door” and “entrance”.
Priority	Medium
Pass/Fail Criteria	The test passes if the detector shows the correct door-related items.
Executed By	QA Tester
Pass/Fail Status	Passed

9. U_SpeechRecognizer initialization

ID	**U9**
Test Case	U_SpeechRecognizer initialization
Test Type	Unit
Item	Voice recognition feature
Pre-Conditions	None
Test Steps	1. Open the voice recognition feature. 2. Check its initial settings.
Expected Results	It should indicate that it has permission and that no speech has been recognized yet.
Priority	Medium
Pass/Fail Criteria	The test passes if the recognizer starts in the correct default state.
Executed By	QA Tester

Pass/Fail Status	Passed

10. U_SpeechRecognizer authorization

ID	U10
Test Case	U_SpeechRecognizer authorization
Test Type	Unit
Item	Voice recognition feature
Pre-Conditions	None
Test Steps	1. Ask the voice recognition feature to check its permission. 2. Wait for the answer.
Expected Results	It should say that permission is granted.
Priority	Medium
Pass/Fail Criteria	The test passes if the answer shows permission is given.
Executed By	QA Tester
Pass/Fail Status	Passed

11. U_SpeechRecognizer recognition

ID	U11
Test Case	U_SpeechRecognizer recognition
Test Type	Unit
Item	Voice recognition feature
Pre-Conditions	A sample phrase is entered into the recognizer

Test Steps	1. Enter the phrase “open the door” in the feature. 2. Start the recognition process.
Expected Results	The system should display that it recognized the phrase “open the door”.
Priority	Medium
Pass/Fail Criteria	The test passes if the output matches the entered phrase exactly.
Executed By	QA Tester
Pass/Fail Status	Passed

12. U_ViewController initialization

ID	**U12**
Test Case	U_ViewController initialization
Test Type	Unit
Item	Main screen (view controller)
Pre-Conditions	None
Test Steps	1. Open the main screen of the app. 2. Check that the screen and its parts load correctly.
Expected Results	The main screen and key parts (like buttons and sliders) are visible.
Priority	Medium
Pass/Fail Criteria	The test passes if the screen and every element appear as expected.
Executed By	QA Tester
Pass/Fail Status	Passed

13. U_ViewController UI elements

ID	U13
Test Case	U_ViewController UI elements
Test Type	Unit
Item	Main screen components
Pre-Conditions	The main screen is open
Test Steps	1. Check the screen for these eight items: one slider, three buttons, three text labels, and a loading indicator.
Expected Results	All eight items should be present on the screen.
Priority	Medium
Pass/Fail Criteria	The test passes if all eight elements are shown.
Executed By	QA Tester
Pass/Fail Status	Passed

14. U_ViewController slider interaction

ID	U14
Test Case	U_ViewController slider interaction
Test Type	Unit
Item	Slider on the main screen
Pre-Conditions	The main screen is open
Test Steps	1. Confirm the slider has not been moved. 2. Slide the control.
Expected	The system should show that the slider

Results	value has changed.
Priority	Medium
Pass/Fail Criteria	The test passes if the slider movement is detected.
Executed By	QA Tester
Pass/Fail Status	Passed

15. U_ViewController button interactions

ID	**U15**
Test Case	U_ViewController button interactions
Test Type	Unit
Item	Buttons on the main screen
Pre-Conditions	The main screen is open
Test Steps	1. Ensure the settings, help, and voice buttons are not active. 2. Tap each button one by one.
Expected Results	Each button should show that it has been pressed.
Priority	Medium
Pass/Fail Criteria	The test passes if all three taps are registered.
Executed By	QA Tester
Pass/Fail Status	Passed

16. U_ViewController label updates

ID	**U16**

Test Case	U_ViewController label updates
Test Type	Unit
Item	Text labels on the main screen
Pre-Conditions	The main screen is open
Test Steps	1. Change the detection label to “Door detected”. 2. Change the confidence label to “95% confidence”. 3. Change the instructions label to “Please move closer”.
Expected Results	Each label displays the new text as described.
Priority	Medium
Pass/Fail Criteria	The test passes if each label shows the updated message.
Executed By	QA Tester
Pass/Fail Status	Passed

17. U_Settings persistence

ID	**U17**
Test Case	U_Settings persistence
Test Type	Unit
Item	Settings storage
Pre-Conditions	None
Test Steps	1. Save a test value under a specific key. 2. Read the value back from storage.
Expected Results	The value read from storage should match the value saved.
Priority	Medium
Pass/Fail	The test passes if the stored value is

Criteria	returned correctly.
Executed By	QA Tester
Pass/Fail Status	Passed

18. U_SharedData singleton

ID	U18
Test Case	U_SharedData singleton
Test Type	Unit
Item	Shared settings (SharedData)
Pre-Conditions	None
Test Steps	1. Access the shared settings twice. 2. Compare the two results.
Expected Results	Both accesses should return the same settings instance.
Priority	Medium
Pass/Fail Criteria	The test passes if both results are identical.
Executed By	QA Tester
Pass/Fail Status	Passed

19. U_Environment configuration

ID	U19
Test Case	U_Environment configuration
Test Type	Unit
Item	Environment settings (for example, an API key)

Pre-Conditions	None
Test Steps	1. Check that the API key is available in the settings.
Expected Results	The API key should not be empty.
Priority	Medium
Pass/Fail Criteria	The test passes if the API key exists.
Executed By	QA Tester
Pass/Fail Status	Passed

20. U_Voice Command Parser for Language Switch

ID	**U20**
Test Case	U_Voice Command Parser for Language Switch
Test Type	Unit
Item	Voice command parser
Pre-Conditions	None
Test Steps	1. Enter the command “Change language to Spanish”.
Expected Results	The system shows that it will change the language to Spanish.
Priority	Medium
Pass/Fail Criteria	The test passes if the parser recognizes the command correctly.
Executed By	QA Tester
Pass/Fail Status	Passed

21. U_Voice Command Parser for Settings Modification

ID	U21
Test Case	U_Voice Command Parser for Settings Modification
Test Type	Unit
Item	Voice command parser
Pre-Conditions	None
Test Steps	1. Enter the command “Set volume to high”.
Expected Results	The system shows that it will change the volume setting to high.
Priority	Medium
Pass/Fail Criteria	The test passes if the parser identifies the settings change correctly.
Executed By	QA Tester
Pass/Fail Status	Passed

22. U_Siri Shortcut Button Function Call

ID	U22
Test Case	U_Siri Shortcut Button Function Call
Test Type	Unit
Item	Siri shortcut button
Pre-Conditions	None
Test Steps	1. Attach a simple action to the button. 2. Press the button.
Expected	The assigned action should run

Results	immediately.
Priority	Medium
Pass/Fail Criteria	The test passes if the button press runs the action.
Executed By	QA Tester
Pass/Fail Status	Passed

23. U_Haptic Feedback Function Trigger

ID	U23
Test Case	U_Haptic Feedback Function Trigger
Test Type	Unit
Item	Vibration feature (haptic feedback)
Pre-Conditions	None
Test Steps	1. Activate the haptic feedback function using a test distance (e.g., 0.5).
Expected Results	The system confirms that vibration occurs by returning “true”.
Priority	Medium
Pass/Fail Criteria	The test passes if the result is “true”.
Executed By	QA Tester
Pass/Fail Status	Passed

24. I_Language Switch via Voice Command

ID	I1
Test Case	I_Language Switch via Voice Command

Test Type	Integration
Item	Voice command feature
Pre-Conditions	None
Test Steps	1. Use the voice command to say “Change language to Spanish”.
Expected Results	The system changes the language setting to Spanish.
Priority	Medium
Pass/Fail Criteria	The test passes if the language setting becomes “es-ES”.
Executed By	QA Tester
Pass/Fail Status	Passed

25. I_Modify Settings with Voice

ID	**I2**
Test Case	I_Modify Settings with Voice
Test Type	Integration
Item	Voice command settings feature
Pre-Conditions	None
Test Steps	1. Use the voice command to say “Set volume to high”.
Expected Results	The system updates the volume setting to high.
Priority	Medium
Pass/Fail Criteria	The test passes if the volume level is updated correctly.
Executed By	QA Tester
Pass/Fail Status	Passed

26. I_Ask Gemini a Question

ID	I3
Test Case	I_Ask Gemini a Question
Test Type	Integration
Item	Question-answer feature (Gemini system)
Pre-Conditions	None
Test Steps	1. Ask “Hey Gemini, what's the weather?” using the app’s function.
Expected Results	The system returns an answer and shows that the question was processed.
Priority	Medium
Pass/Fail Criteria	The test passes if a valid answer is returned.
Executed By	QA Tester
Pass/Fail Status	Passed

27. I_Open the App with Siri

ID	I4
Test Case	I_Open the App with Siri
Test Type	Integration
Item	App opening via Siri
Pre-Conditions	None
Test Steps	1. Tell Siri “Open Assistive Vision”.
Expected Results	The app opens and a confirmation sound plays.

Priority	Medium
Pass/Fail Criteria	The test passes if the app opens and the sound is heard.
Executed By	QA Tester
Pass/Fail Status	Passed

28. I_Audio Confirmation When App Opens

ID	I5
Test Case	I_Audio Confirmation When App Opens
Test Type	Integration
Item	App launch audio confirmation
Pre-Conditions	None
Test Steps	1. Launch the app using the proper method.
Expected Results	A confirmation sound should play when the app opens.
Priority	Medium
Pass/Fail Criteria	The test passes if the audio confirmation is heard.
Executed By	QA Tester
Pass/Fail Status	Passed

29. S_Continuous Mode Overall Operation

ID	S1
Test Case	S_Continuous Mode Overall Operation
Test Type	System

Item	Continuous operation mode
Pre-Conditions	None
Test Steps	1. Enable continuous mode in the app. 2. Wait a short period to let the system work.
Expected Results	The screen shows that continuous mode is active, the camera feed is processed, and sound plus detection messages are provided.
Priority	Medium
Pass/Fail Criteria	The test passes if all features of continuous mode are active.
Executed By	QA Tester
Pass/Fail Status	Passed

30. S_On-Demand Mode Overall Operation

ID	**S2**
Test Case	S_On-Demand Mode Overall Operation
Test Type	System
Item	On-demand scanning mode
Pre-Conditions	None
Test Steps	1. Activate on-demand mode. 2. Press the “scan” button.
Expected Results	The system indicates on-demand mode is active, then takes one picture, processes it, and provides feedback while keeping the live feed off.
Priority	Medium
Pass/Fail Criteria	The test passes if a single scan occurs with proper feedback.

Executed By	QA Tester
Pass/Fail Status	Passed

31. S_Haptic Feedback for Approaching Hazard

ID	S3
Test Case	S_Haptic Feedback for Approaching Hazard
Test Type	System
Item	Hazard alert feature
Pre-Conditions	None
Test Steps	1. Simulate an object coming very close (for example, 0.3 units away).
Expected Results	The system vibrates (haptic feedback) and plays an audio alert.
Priority	High
Pass/Fail Criteria	The test passes if both vibration and sound alerts are produced.
Executed By	QA Tester
Pass/Fail Status	Passed

32. U_FAQ Content Localization

ID	U24
Test Case	U_FAQ Content Localization
Test Type	Unit
Item	FAQ screen text
Pre-Conditions	The FAQ screen is open

Test Steps	1. Set the FAQ screen to show a sentence in English. 2. Change the language to Spanish and update the text accordingly.
Expected Results	The FAQ text changes to the correct language each time.
Priority	Medium
Pass/Fail Criteria	The test passes if the displayed text is correct for each language.
Executed By	QA Tester
Pass/Fail Status	Passed

33. U_Speech Message Functionality

ID	**U25**
Test Case	U_Speech Message Functionality
Test Type	Unit
Item	Voice output (text-to-speech)
Pre-Conditions	None
Test Steps	1. Use the speech function to speak a test message (“Test message”).
Expected Results	The system indicates that it is speaking, and the message is recorded.
Priority	Medium
Pass/Fail Criteria	The test passes if text-to-speech is triggered and shows the correct message.
Executed By	QA Tester
Pass/Fail Status	Passed

34. U_Read Button Functionality

ID	U26
Test Case	U_Read Button Functionality
Test Type	Unit
Item	Read button feature
Pre-Conditions	None
Test Steps	1. Press the read button on the FAQ screen.
Expected Results	The system starts to speak the displayed text.
Priority	Medium
Pass/Fail Criteria	The test passes if the speech output is activated.
Executed By	QA Tester
Pass/Fail Status	Passed

35. U_Auto Read on Appear

ID	U27
Test Case	U_Auto Read on Appear
Test Type	Unit
Item	Auto-read feature on FAQ screen
Pre-Conditions	shouldReadText is set to true
Test Steps	1. Open the FAQ screen and let the view appear.
Expected Results	The system automatically starts to speak the text.
Priority	Medium

Pass/Fail Criteria	The test passes if the auto-read function is triggered.
Executed By	QA Tester
Pass/Fail Status	Passed

36. S_FAQ View Controller Initialization

ID	S4
Test Case	S_FAQ View Controller Initialization
Test Type	System
Item	FAQ screen
Pre-Conditions	None
Test Steps	1. Open the FAQ screen.
Expected Results	The FAQ screen displays its text area and has the speech output ready.
Priority	Medium
Pass/Fail Criteria	The test passes if the FAQ screen loads properly.
Executed By	QA Tester
Pass/Fail Status	Passed

37. S_Settings View Controller Initialization

ID	S5
Test Case	S_Settings View Controller Initialization
Test Type	System
Item	Settings screen

Pre-Conditions	None
Test Steps	1. Open the settings screen.
Expected Results	All the settings options (text field, sliders, toggles, etc.) are visible.
Priority	Medium
Pass/Fail Criteria	The test passes if all expected settings options are shown.
Executed By	QA Tester
Pass/Fail Status	Passed

38. U_Initial Values Test

ID	**U30**
Test Case	U_Initial Values Test
Test Type	Unit
Item	Settings screen values
Pre-Conditions	None
Test Steps	1. Pre-load storage with test values (for example, iou = 0.7, confidence = 0.5). 2. Open the settings screen.
Expected Results	The settings screen shows the values saved in storage.
Priority	Medium
Pass/Fail Criteria	The test passes if the settings options display the correct saved values.
Executed By	QA Tester
Pass/Fail Status	Passed

39. U_Slider Updates Test

ID	U31
Test Case	U_Slider Updates Test
Test Type	Unit
Item	Sliders on the settings screen
Pre-Conditions	The settings screen is open
Test Steps	1. Change the slider values (for example, adjust sensitivity). 2. Confirm that the new values are saved in storage.
Expected Results	The storage shows the updated slider values.
Priority	Medium
Pass/Fail Criteria	The test passes if the new slider values are correctly saved.
Executed By	QA Tester
Pass/Fail Status	Passed

40. U_Language Change Test

ID	U32
Test Case	U_Language Change Test
Test Type	Unit
Item	Language option in the settings screen
Pre-Conditions	The settings screen is open
Test Steps	1. Change the language option to Spanish and save.

	2. Change it back to English and save again.
Expected Results	The storage shows the language changes correctly (first “es-ES” then “en-US”).
Priority	Medium
Pass/Fail Criteria	The test passes if the language option is updated in storage.
Executed By	QA Tester
Pass/Fail Status	Passed

41. U_Haptics Change Test

ID	**U33**
Test Case	U_Haptics Change Test
Test Type	Unit
Item	Haptics toggle on the settings screen
Pre-Conditions	None
Test Steps	1. Turn the haptics toggle off and save. 2. Turn it on and save again.
Expected Results	The storage shows the correct haptics setting (first false, then true).
Priority	Medium
Pass/Fail Criteria	The test passes if the haptics toggle updates correctly in storage.
Executed By	QA Tester
Pass/Fail Status	Passed

42. U_Scanning Mode Change Test

ID	U34
Test Case	U_Scanning Mode Change Test
Test Type	Unit
Item	Scanning mode option in the settings screen
Pre-Conditions	None
Test Steps	1. Select continuous mode and save. 2. Select on-demand mode and save.
Expected Results	The storage shows the correct scanning mode setting (true for continuous then false for on-demand).
Priority	Medium
Pass/Fail Criteria	The test passes if the scanning mode option is updated correctly in storage.
Executed By	QA Tester
Pass/Fail Status	Passed

43. U_Model Change Test

ID	U35
Test Case	U_Model Change Test
Test Type	Unit
Item	Model selection in the settings screen
Pre-Conditions	None
Test Steps	1. Change the model option to the second option. 2. Save the change.

Expected Results	The storage should show that the model index is updated to 1.
Priority	Medium
Pass/Fail Criteria	The test passes if the model option is saved correctly.
Executed By	QA Tester
Pass/Fail Status	Passed

44. U_Speech Message Test

ID	**U36**
Test Case	U_Speech Message Test
Test Type	Unit
Item	Speech output (text-to-speech) in settings screen
Pre-Conditions	None
Test Steps	1. Use the “speakText” function to speak “Test message”.
Expected Results	The system indicates it is speaking and the message “Test message” is recorded.
Priority	Medium
Pass/Fail Criteria	The test passes if text-to-speech is activated and the correct message is shown.
Executed By	QA Tester
Pass/Fail Status	Passed

45. I_Settings Persistence Test

ID	I6
Test Case	I_Settings Persistence Test
Test Type	Integration
Item	Settings screen and storage
Pre-Conditions	None
Test Steps	1. Open the settings screen and change some values. 2. Close the screen and open it again to check the values.
Expected Results	The settings screen shows the same values as before.
Priority	Medium
Pass/Fail Criteria	The test passes if the settings remain unchanged between sessions.
Executed By	QA Tester
Pass/Fail Status	Passed

46. I_Settings Notification Updates Test

ID	I7
Test Case	I_Settings Notification Updates Test
Test Type	Integration
Item	Slider update events on the settings screen
Pre-Conditions	A notification observer is set up
Test Steps	1. Change a slider value on the settings screen.

	2. Check that a notification is received.
Expected Results	A notification is received confirming that the slider value changed.
Priority	Medium
Pass/Fail Criteria	The test passes if the expected notification is observed.
Executed By	QA Tester
Pass/Fail Status	Passed

47. U_Siri Shortcut Availability Test

ID	**U39**
Test Case	U_Siri Shortcut Availability Test
Test Type	Unit
Item	Siri shortcut feature
Pre-Conditions	None
Test Steps	1. Open the Siri shortcut feature.
Expected Results	The system shows that there is a valid Siri shortcut set up.
Priority	Medium
Pass/Fail Criteria	The test passes if a valid Siri shortcut is present.
Executed By	QA Tester
Pass/Fail Status	Passed

48. U_Siri Shortcut Response Test

ID	U40
Test Case	U_Siri Shortcut Response Test
Test Type	Unit
Item	Siri shortcut feature
Pre-Conditions	None
Test Steps	1. Check the response code after triggering the Siri shortcut.
Expected Results	The response should indicate success.
Priority	Medium
Pass/Fail Criteria	The test passes if the response shows success.
Executed By	QA Tester
Pass/Fail Status	Passed

49. I_Siri Shortcut Handling Integration Test

ID	I8
Test Case	I_Siri Shortcut Handling Integration Test
Test Type	Integration
Item	Siri shortcut handling feature
Pre-Conditions	None
Test Steps	1. Trigger the Siri shortcut handler. 2. Check the result.
Expected Results	The system confirms that the Siri shortcut was handled successfully.

Priority	Medium
Pass/Fail Criteria	The test passes if the handler responds with success.
Executed By	QA Tester
Pass/Fail Status	Passed

50. U_Continuous Mode Toggle

ID	U42
Test Case	U_Continuous Mode Toggle
Test Type	Unit
Item	Continuous mode option on the main screen
Pre-Conditions	None
Test Steps	1. Press the continuous mode button to turn it on. 2. Press it again to turn it off.
Expected Results	The system shows that continuous mode turns on and then off, with the proper sound and text messages.
Priority	Medium
Pass/Fail Criteria	The test passes if the continuous mode status changes correctly.
Executed By	QA Tester
Pass/Fail Status	Passed

51. U_On-Demand Mode Toggle

ID	U43
Test Case	U_On-Demand Mode Toggle

Test Type	Unit
Item	On-demand mode option on the main screen
Pre-Conditions	None
Test Steps	1. Press the on-demand mode button once to activate it. 2. Press it again to deactivate it.
Expected Results	The system toggles the on-demand mode between active and inactive.
Priority	Medium
Pass/Fail Criteria	The test passes if the on-demand mode switches correctly.
Executed By	QA Tester
Pass/Fail Status	Passed

52. U_Scanning Mode Persistence

ID	**U44**
Test Case	U_Scanning Mode Persistence
Test Type	Unit
Item	Scanning mode setting on the main screen
Pre-Conditions	None
Test Steps	1. Turn on continuous mode and save the setting. 2. Close and reopen the main screen to check the setting.
Expected Results	The main screen should show that continuous mode is still active.
Priority	Medium
Pass/Fail Criteria	The test passes if the setting is retained.

Executed By	QA Tester
Pass/Fail Status	Passed

53. I_Scanning Mode Audio Feedback

ID	I9
Test Case	I_Scanning Mode Audio Feedback
Test Type	Integration
Item	Continuous mode function on the main screen
Pre-Conditions	None
Test Steps	1. Press the button to enable continuous mode.
Expected Results	The system plays an audio message that mentions “Continuous mode”.
Priority	Medium
Pass/Fail Criteria	The test passes if the audio feedback is heard.
Executed By	QA Tester
Pass/Fail Status	Passed

54. I_Scanning Mode Haptic Feedback

ID	I10
Test Case	I_Scanning Mode Haptic Feedback
Test Type	Integration
Item	Continuous mode function on the main screen

Pre-Conditions	None
Test Steps	1. Press the button to enable continuous mode.
Expected Results	The system produces a vibration (haptic feedback).
Priority	Medium
Pass/Fail Criteria	The test passes if haptic feedback is activated.
Executed By	QA Tester
Pass/Fail Status	Passed

55. S_Tutorial Screen Initialization

ID	**S6**
Test Case	S_Tutorial Screen Initialization
Test Type	System
Item	Tutorial screen
Pre-Conditions	None
Test Steps	1. Open the tutorial screen of the app.
Expected Results	The tutorial screen shows all its elements (text area, chat table, text field, send button, and loading indicator).
Priority	Medium
Pass/Fail Criteria	The test passes if every main element of the tutorial screen is visible.
Executed By	QA Tester
Pass/Fail Status	Passed

56. U_Tutorial Language Localization

ID	U48
Test Case	U_Tutorial Language Localization
Test Type	Unit
Item	Tutorial screen text
Pre-Conditions	None
Test Steps	1. Set the screen language to English and fill in greeting and placeholder text. 2. Change the language to Spanish and update the texts.
Expected Results	The greeting and placeholder texts change to the new language.
Priority	Medium
Pass/Fail Criteria	The test passes if the displayed texts are updated correctly for each language.
Executed By	QA Tester
Pass/Fail Status	Passed

57. U_Initial Message Test

ID	U49
Test Case	U_Initial Message Test
Test Type	Unit
Item	Tutorial screen chat log
Pre-Conditions	None
Test Steps	1. Open the tutorial screen.
Expected	The chat log should contain one initial

Results	greeting message.
Priority	Medium
Pass/Fail Criteria	The test passes if the initial greeting is correct.
Executed By	QA Tester
Pass/Fail Status	Passed

58. U_Send Button Test

ID	**U50**
Test Case	U_Send Button Test
Test Type	Unit
Item	Tutorial screen chat log
Pre-Conditions	None
Test Steps	1. Enter “Test question” into the message field. 2. Press the send button.
Expected Results	The chat log should add a new message reading “Test question”.
Priority	Medium
Pass/Fail Criteria	The test passes if the new message appears in the chat log.
Executed By	QA Tester
Pass/Fail Status	Passed

59. S_Camera system

ID	**S7**

Test Case	S_Camera system
Test Type	System
Item	Camera capture feature
Pre-Conditions	None
Test Steps	1. Enable the camera setup process. 2. Start the camera. 3. Simulate a frame capture. 4. Stop the camera.
Expected Results	The system shows that the camera has been set up; it starts, captures a frame, and stops without issues.
Priority	Medium
Pass/Fail Criteria	The test passes if the full camera process works as described.
Executed By	QA Tester
Pass/Fail Status	Passed

60. S_Detection system

ID	**S8**
Test Case	S_Detection system
Test Type	System
Item	Object detection features
Pre-Conditions	None
Test Steps	1. Set the general detector to expect “person” and “chair”. 2. Set the door detector to expect “door” and “entrance”. 3. Ask both detectors to scan a sample image.
Expected Results	The general detector shows “person” and “chair”, and the door detector shows “door”

	and “entrance”.
Priority	Medium
Pass/Fail Criteria	The test passes if both detectors return the correct items.
Executed By	QA Tester
Pass/Fail Status	Passed

61. S_Voice command system

ID	S9
Test Case	S_Voice command system
Test Type	System
Item	Voice command recognition
Pre-Conditions	None
Test Steps	1. Enter “open the door” as the spoken phrase. 2. Start the recognition function.
Expected Results	The system displays “open the door” as the recognized phrase and processes the command.
Priority	Medium
Pass/Fail Criteria	The test passes if the phrase is recognized exactly and the command is identified.
Executed By	QA Tester
Pass/Fail Status	Passed

62. S_Settings system

ID	S10
Test Case	S_Settings system
Test Type	System
Item	App settings
Pre-Conditions	None
Test Steps	1. Change settings like detection threshold, voice commands, and language in the app. 2. Confirm that the app reads and applies these settings correctly.
Expected Results	The app shows the updated settings correctly.
Priority	Medium
Pass/Fail Criteria	The test passes if the app applies the settings from storage.
Executed By	QA Tester
Pass/Fail Status	Passed

63. S_Error handling system

ID	S11
Test Case	S_Error handling system
Test Type	System
Item	Error management for camera, detection, and voice recognition
Pre-Conditions	None

Test Steps	1. Simulate an error in the camera. 2. Simulate an error in the object detector. 3. Simulate an error in the voice recognition feature. 4. Try running the voice recognition function.
Expected Results	Each part should show an error message, and the voice recognition function should indicate failure with the correct error message.
Priority	Medium
Pass/Fail Criteria	The test passes if errors are handled and the correct messages appear.
Executed By	QA Tester
Pass/Fail Status	Passed

64. S_Performance system

ID	**S12**
Test Case	S_Performance system
Test Type	System
Item	Camera setup and detection performance
Pre-Conditions	None
Test Steps	1. Start the camera setup and detection process. 2. Measure the time taken to complete the process.
Expected Results	The total time taken should be less than 5 seconds.
Priority	Medium
Pass/Fail Criteria	The test passes if the process meets the time requirement.

Executed By	QA Tester
Pass/Fail Status	Passed

65. S_Memory management system

ID	S13
Test Case	S_Memory management system
Test Type	System
Item	View controller memory usage
Pre-Conditions	None
Test Steps	1. Open a view on a temporary basis. 2. Close the view and check if the system has released it.
Expected Results	The view should be fully released from memory.
Priority	Medium
Pass/Fail Criteria	The test passes if the temporary view is deallocated.
Executed By	QA Tester
Pass/Fail Status	Passed

Traceability Matrix

Requirement ID	Requirement Name	Use Case ID	Test Case ID
FR1	Launch Application	UC-01	U12, U13, I5
FR2	Camera Initialization	UC-12	U1, U2, U3, U4, S7
FR3	Real Time Object Detection (Continuous Mode)	UC-02	U5, U6, S1, S8
FR4	Object Detection (On Demand Mode)	UC-03	U5, U6, S2, S8
FR5	Provide Audio Feedback	UC-13	U25, U26, U27, U36, I5, I9
FR6	Voice Command Interface	UC-14	U9, U10, U11, U20, U21, S9
FR7	Customizable Detection Settings	UC-04	U30, U31, U34, U35, I6, I7
FR8	Audio and Haptic Feedback Configuration	UC-08	U23, U33, I10, S3
FR9	Continuous versus On Demand Detection Modes	UC-09	U42, U43, U44, I9, I10
FR10	Detection History	UC-07	U16
FR11	Siri Integration and Shortcuts	UC-16	U22, U39, U40, I4, I8
FR12	User Help and Tutorial	UC-17	U24, U48, U49, U50, S4, S6
FR13	Battery Optimization Mode	UC-06	S12
FR14	Offline Operation	UC-18	S10
FR15	Hazard Alert Customization	UC-15	U23, S3
FR16	Scene Exploration Mode	UC-05	U7, U8, S8
FR17	Local Usage Analytics	UC-10	U17, U18
FR18	Haptic Feedback for Close Objects	UC-11	U23, S3
NFR1	Performance	UC-02	S12
NFR2	Reliability	UC-03	S11
NFR3	Availability	UC-01	S10
NFR4	Security	UC-04	U19
NFR5	Maintainability	UC-17	S13
NFR6	Portability	UC-19	U32, I1