Project Title: System Verification and Validation Plan for SFWRENG 4G06 Capstone Design Project

Team #18, InfiniView-AI
Anhao Jiao
Kehao Huang
Qianlin Chen
Qi Shu
Xunzhou Ye

March 27, 2024

Revision History

Date	Developer(s)	Change
November 3, 2023	KH, QS, AJ, QC, XY	Initial Draft
March 6, 2024	KH, QS, AJ, QC, XY	Rev 0.1
March 27, 2024	KH, QS, AJ, QC, XY	Rev 1

Contents

List of Tables

1 Symbols, Abbreviations, and Acronyms

1.1 Symbols

Table 1 lists all the symbols, abbreviations, and acronyms used throughout this document.

Symbol	Description
Tai Chi	A classical Chinese martial art system practiced for health promotion and rehabilitation.
Instructor	A person who teaches a Tai Chi class through an online conference system.
Practitioner	A person who learns Tai Chi through an online conference system.
Pipeline	A sequence of processing elements connected in series, which are responsible for generating annotations.
UI	User Interface.
TA	Teaching Assistant.
SRS	The Software Requirements Specification document.
POC	proof of concept.
SFU	Selective Forwarding Unit, a component in real-time communication systems like WebRTC that routes and selectively forwards audio and video streams from one participant to others.
V&V Plan	The verification and validation plan document(this document).
Type: Functional	Functional test: An input/output blackbox type test.
Type: Dynamic	Dynamic test: Test type that requires code/program to be executed.
Type: Manual	Manual test: Test type that requires the user to do the test without using automated tools.
Type: Automated	Automated test: Test type containing code/program that can be executed.
Stress test	Testing the limits (often the amount of data) of a system.
Performance test	Testing the performance(includes different metrics) of a system.

Table 1: List of symbols, abbreviations, and acronyms

1.2 Symbolic Constants

Symbolic Constant	Value
MAX_RESPONSE_TIME	2 seconds
TEST_ITERATIONS	100
TEST_ACCOUNTS_NUMBERS	5
MIN_NUM_USER_SUPPORTED	5
TEST_DISCONNECT_DURATION	30 seconds
MAX_DELAY	500 ms
MAX_TEMP	65 °C
MIN_RES	720p
TARGET_FACTOR	85 %

Table 2: List of Symbolic Constants

2 General Information

This section introduces some general information about the present document, including a summary, objectives of the document, and related documentations that were mentioned or referred to in the present document.

2.1 Summary

This document describes the testing, validation, and verification procedures that will be implemented for the "Real-time Digital Annotation Solution for Online Exercise Lessons" project. The general functionality of this project is to set up a live stream call streaming online TaiChi exercising class, while practitioners can configure the annotation they like to be added on to the instructor's video stream. The test cases specified in this document were conceived prior to the majority of the implementation and are intended to be used by the project team for future reference during project development, testing, and maintenance.

2.2 Objectives

The objectives of the validation and verification document for this project isare to serve as a comprehensive guideline for validating project requirements, ensuring the correctness of code, and establishing metrics to enhance user satisfaction. As all developers working on this project also have other life priorities, we don't have the resources to test out all aspects of the project. Some parts of the project will be considered out of scope. For instance, we will not be testing the machine learning model utilized in the project, and we assume the machine learning model has already been verified by its implementation team. And Our tests specified in this document will prioritize the functionalities and connections between components.
The objectives of the validation and verification document for this project are to provide a comprehensive guide for validating project requirements, ensuring the correctness of code, and establishing metrics to enhance user satisfaction. Given the scope of the project and the time constraints, certain aspects will be prioritized to ensure effective use of resources. For example, this document will not cover the testing of the machine learning model utilized in the project, as it is presumed to have undergone verification by its dedicated implementation team. The focus of the testing will be on critical functionalities and the interconnectivity between components, to ensure a robust and seamless user experience.

2.3 Relevant Documentation

Relevant documentations include:

SRS The SRS document is a related document as most of the requirements specified in this document come from the SRS document.

Development Plan The Development Plan document is a related document as it specifies the technologies that will be used in the project, including testing tools.

Hazard Analysis The Hazard Analysis document is a related document as some of the requirements come from the Hazard Analysis document.

Module Guide The Module Guide is essential as it will outline the high-level architecture of the system, providing a roadmap for development and facilitating understanding of the system's structure and modularity.

Module Interface Specification The Module Interface Specification is pivotal for detailing the interactions between various system components, ensuring compatibility and defining the communication protocols necessary for module integration.

3 Plan

This section introduces the verification and validation team with their associated responsibilities, and provides verification plans for SRS, system design, V&V plan, and system implementation. In addition, it introduces several automated testing and verification tools that are planned to be utilized by the testing team during the verification and validation process.

3.1 Verification and Validation Team

The verification and validation process is a collective responsibility within the team, with each member actively participating.

Kehao Huang, Qi Shu Responsible for Integration of Machine Learning Models and Application Back-end Testing. This involves unit testing to validate each individual component of the machine learning models for correctness, efficiency, and reliability. Integration testing will follow, which checks the interaction

between the machine learning models and the back-end systems to ensure data flows correctly and that the models perform as expected within the application environment. They might also use validation testing to confirm that the models meet the initial requirements and perform accurately with real-world data.

Qianlin Chen, Xunzhou Ye, Anhao Jiao Responsible for Full Stack Development Testing. For the front-end, they will use black box testing, which tests the user interface without knowledge of the underlying code, to ensure the user experience is as designed and is bug-free. They will conduct white box testing on the back-end to verify the internal workings of the application's logic. System testing will also be part of their responsibility to check the complete and integrated software product functions in line with the requirements. Finally, they'll perform end-to-end testing to simulate real user scenarios and make sure the application behaves as expected in production.

3.2 SRS Verification Plan

~~Our SRS (Software Requirements Specification) will be subject to verification through a flexible and collaborative feedback process involving our classmates and Teaching Assistants (TAs). Given that the individuals involved in reviewing our VnV (Verification and Validation) plan and our TAs possess in-depth knowledge of our SRS, we have designed a verification plan that encourages them to explore our application with a degree of improvisation. The primary goal is to ensure comprehensive coverage of every functionality specified in our SRS. Key aspects of the verification approach include flexible feedback, exploratory assessment, and functional coverage. Reviewers will be encouraged to explore the application using their own methods and approaches, allowing for creative and innovative testing techniques. The review process will actively seek out and identify possible behavioural bugs and anomalies within the application. This approach leverages the diverse perspectives and problem-solving abilities of our reviewers, helping us ensure that our SRS is thorough, accurate, and aligned with the project's objectives, ultimately contributing to the success of our project.~~

The Software Requirements Specification (SRS) is crucial for defining the project scope and guiding its implementation. To ensure the SRS meets the project's standards and objectives, a rigorous verification plan involving iterative feedback from peers and Teaching Assistants (TAs) will be employed. This process begins with an initial review where the SRS will be scrutinized for clarity, completeness, and alignment with the project's goals. Feedback will be systematically gathered and categorized to identify areas needing refinement. Additionally, exploratory assess-

ment techniques will be utilized, enabling reviewers to interact with the application's prototype or review design mockups, thereby ensuring requirements are realistically translated into application features.

Subsequent to the exploratory phase, functional testing based on the SRS will be performed to validate the behavior of the application. This testing will be geared toward verifying that each function of the application performs according to the specified requirements. The feedback gathered will be carefully integrated into the SRS, with the development team revising any sections as necessary to improve clarity and accuracy. The final step of the verification plan involves a comprehensive review to confirm that all feedback has been effectively addressed, solidifying the SRS as a true reflection of the intended system. By leveraging the diverse insights from our reviewers, the SRS will be refined to a state that facilitates a smooth transition into the development phase, setting a strong foundation for the project's success.

3.3 Design Verification Plan

The design verification plan outlines the strategies and procedures that will be taken by the team to verify the correctness and reliability of the design of our Tai Chi video conferencing application. This plan will be treated as a guideline during the testing phase to ensure that the design meets the intended requirements and are able to mitigate potential hazards that the team has discovered. The following procedures will be taken by the testing team during the verification process:

Document Review The system's design documentation and related material will be reviewed by each member of the testing team after the first draft is produced. During the document review process, the testing team will

1. ensure that the design of the system aligns with all the functional and non-functional requirements, and
2. assess if the document accurately describes the intended functionality and behaviour of the system. Any design elements that deviate from specified requirements should be recorded and reported for further discussion.

Prototype Testing After the scheduled POC demo, there should be a completed prototype for each key component of the system. The prototypes will be assembled to conduct system testing. During the prototype testing process, the testing team will verify the usability of the user interface, system performance under different loads, and handling of failure conditions.

3.4 Verification and Validation Plan Verification Plan

The verification and validation plan verification plan provides a guideline for the testing team to ensure the quality, accuracy and completeness of the verification and validation plan document itself. The following procedures will be taken by the testing team during the verification process:

Document Review The V&V plan document will be reviewed by each member in the testing team after the initial draft is finished. The testing team will ensure that the document addresses all the key aspects of the verification and validation process, aligns with the project objectives and goals, and maintains completeness and consistency.

Cross-Reference with other Documents After the initial draft of the V&V plan document is completed, the document will be cross-referenced with other projects including problem statement, SRS and hazard analysis to verify consistency and alignment with the project's overall objective.

Stakeholder Review The initial draft of the V&V plan document will be presented to relevant stakeholders to gather feedback. The feedback will be reviewed and discussed by the testing team, and changes to the V&V plan document will be made accordingly to ensure stakeholders' satisfaction.

3.5 Implementation Verification Plan

~~For ensuring~~To ensure the accuracy and effectiveness of our code implementation, we will employ a multi-faceted verification approach.

System Testing The primary means of validating the functionality and efficiency of our software will be through system testing. This process will confirm that our software consistently executes all designated tasks as per the given specifications. Furthermore, Non-Functional Requirements (NFRs) validation will also be incorporated as part of this system testing. The tests will simulate real-world conditions and environments to ensure our application delivers optimal performance.

Unit Testing At a backend level, unit tests will be in place to evaluate the individual components or modules within our system. These tests, aided by appropriate stubs and drivers, will focus on the quality of each module, ensuring they operate as intended. For every function within a module, we will formulate distinct test cases to guarantee precision and reliability.

Static Verification Beyond dynamic testing, our implementation will be subjected to static verification methods. We propose a peer review system where fellow developers and the TA will conduct a thorough examination of our source code. This manual inspection serves a dual purpose:

1. Highlight any deviations from the accepted coding standards and conventions.
2. Unearth potential logical inconsistencies or oversights in our coding that automated tests might miss.

By integrating these multifaceted testing and verification techniques, we aim to ensure our code is both robust and aligned with ~~the~~ both functional and non-functional requirements. This plan will not only verify that the implementation meets the designated specifications but will also foster code quality, readability, and maintainability.

3.6 Automated Testing and Verification Tools

To ensure the robustness and stability of our real-time video conference application, we will utilize a range of automated testing and verification tools tailored to each specific component of our system.

Module/Component	**Testing & Verification Tools**
Front-End (Client-Side)	
UI	React-testing-library
WebRTC Client	KITE, testRTC
Real-Time Communication Engine	RCTPeerConnection API, WebRTC Internals
Back-End (Server-Side)	
Signaling Server	Socket.io tester, Wireshark
SFU	Jitsi-meet-torture
API Server	Jest, Cypress.io

Table 3: List of testing tools

The tinter tool for both Front-End and Back-End will be ESlint in order to make sure the coding standard is followed. Table 3 listed the relevant testing tools. For the client-side components:

UI We will use React-testing-library for unit tests in the user interface, as the react-testing-library is the default and most used library for react testing.

WebRTC Client We will use KITE (Karoshi Interoperability Testing Engine) for compatibility and interoperability tests, while testRTC helps in monitoring, functional and load testing of the WebRTC services.

Real-Time Communication Engine The RTCPeerConnection API will be used to inspect the properties and performance of a WebRTC peer connection. The WebRTC ~~I~~internals tool offers deeper insights into the client-side WebRTC status.

On the server-side:

Signaling Server Socket.io tester will help in validating WebSocket communication, while Wireshark will allow for network protocol analysis ensuring data packets are transmitted correctly.

SFU The Jitsi-meet-torture will be employed for stress and performance testing. API Server: Jest will be employed for unit testing our server-side logic, and Cypress.io will be used for more complex integration tests.

Through this diverse toolkit, we aim to maintain the high reliability and functionality standards of our real-time video conference application.

3.7 Software Validation Plan

In the development phase, sync-up and review sessions will be conducted with the stakeholders on a regular basis. Intermediate results or evidence of progression will be shared and discussed with supervising stakeholders. This is to ensure that the requirement documents together with the development, under the assumption that the development is verified to satisfy the requirements, are aligned with the stakeholders' expectations and needs.

Near the end of the capstone timeline as the product is being gradually finalized within the refined scope of the capstone project, a software design evaluation and validation study will be conducted with volunteers from each of the intended user groups. Specifically, at least one Tai Chi instructor and at least one Tai Chi practitioner or potential Tai Chi student should be invited to evaluate the software product. The process will be broken down into two main components. The first one ~~being~~is an observation study. Study participants will be asked to perform specific tasks as identified and described as use case scenarios in SRS. For the instructor, tasks include:

- Launch the application and set up appropriate peripherals, such as webcams~~,~~ and microphones, to create an environment ready for online instructional Tai Chi demonstrations.
- Navigate through the application interfaces and initiate a video conferencing session.

For the practitioner, tasks include:

- Navigate through the application interfaces and join an existing video conferencing session. Attempt to follow along the instructor's movements.

Developments will observe and note how well the participants can complete the given tasks, making connections to the fit criteria~~s~~ of the non-functional requirements. The second component will be a user satisfaction survey (refer to Appendix 6.1) conducted after the observation study. Participants will be asked to take a survey to rate their experience in using the application, ~~in the~~on a scale of 1 (unsatisfied) to 5 (satisfied). The targeted satisfactory factor is set to TARGET_FACTOR. The survey also includes questions about difficulties the user has encountered, and potential future improvements can be ~~done~~made to the application.

For future developments beyond the capstone timeline, the development team will continue to conduct review sessions and presentations with stakeholder representatives, making sure that the development stays on track. For each major revision or software release, product evaluations will be carried out with targeted users. The feedback and user data gathered from the evaluation process will be used to guide the development and refinement of the product.

4 System Test Description

This sections provides system tests for both functional requirements and non-functional requirements.

4.1 Tests for Functional Requirements

FR-T1 **Type** Functional, Dynamic, Automated

Initial State Client application is running on the user's device, but the user didn't do any operations yet.

Input/Condition User clicks on the applicable identity(instructor/practitioner) button

Output/Result The live stream video window pops out on the user's screen.

How Test Will Be Performed Test will be done automatically by the Cypress testing framework in the CI pipeline. The test code will simulate a user's click on the screen and click on the applicable identity button, then assert the live stream video window showed up on the user's screen within the next ~~2 seconds~~MAX_RESPONSE_TIME.

FR-T2 **Type** Functional, Dynamic, Manual

Initial State Application running on user's computer, and the user has clicked on "the instructor identity button" to indicate they are a TaiChi instructor. A window asking for permission to use the camera on the instructor's device popped out.

Input/Condition User allow/deny the webcam permission

Output/Result The webcam on the instructor's device is turned on

How Test Will Be Performed Test will be done manually by developers running the prototype. The developer will make sure the on-device camera should be turned on after allowing the webcam permission, or the on-device camera should not be turned on after denying the webcam permission. The test should be done at least ~~100~~TEST_ITERATIONS times to ensure the webcam on the instructor's device is turned on properly.

FR-T3 **Type** Functional, Dynamic, Automatic

Initial State ~~b~~Both client applications and the server are running.

Input/Condition The user clicks on the applicable identity button to indicate they are an instructor or a practitioner.

Output/Result A log message indicates connection between the user's device and the server has been established.

How Test Will Be Performed Test will be done automatically by the Cypress testing framework in the CI pipeline. The test code for the client application component will try to connect to a server and establish connection, then it will assert the log indicates a successful connection showed up in the console.

FR-T4 **Type** Functional, Dynamic, Automated

Initial State The live stream Window for practitioners.

Input/Condition The user's device has established a connection with the server as a practitioner device.

Output/Result A request from the client device to the server for accessing the list of available annotation configurations.

How Test Will Be Performed Test will be done automatically by the Cypress testing framework. The test code will first establish a connection with a server as a practitioner device. Once the connection is established, the server will send a list of available annotations to the client device, and the client device should render the list into a selectable list of types of annotation on the user's screen. The test for the client application will assert the list of available annotations from the server is received and rendered as a selectable list.

FR-T5 **Type** Functional, Dynamic, Automated

Initial State The selectable list of the type of annotations is rendered on the user's screen.

Input/Condition Practitioner's selection on the list of types of annotations.

Output/Result A request(that reflects user's annotation selection) from the client device to the server for updating the annotation configuration, with a log indicating the request is sent.

How Test Will Be Performed Test will be done automatically by the Cypress testing framework. The unit test code will first establish connection with a server as a practitioner device. The testing framework will then simulate a user's click to configure annotation selections on the rendered selectable list. After that, the testing framework will assert a request that reflects the annotation configuration ~~is~~ received by the server. As well, assert a log is shown on the client device console indicating the request has been sent.

FR-T6 **Type** Functional, Dynamic, Automated

Initial State The system is running and actively connected to practitioners.

Input/Condition Practitioners initiate updates to annotation configurations.

Output/Result The system receives and processes the updated annotation configurations.

How Test Will Be Performed Automated tests will be conducted to verify that the system successfully listens to updates on annotation configurations from practitioners. In a controlled test environment, practitioners

will initiate updates to annotation configurations. The test will then verify that the system correctly receives and processes the updated annotation configurations, ensuring that it can effectively listen to and respond to practitioner-initiated updates.

FR-T7 **Type** Functional, Dynamic, Automated

Initial State The server is running and actively receiving annotation configuration updates.

Input/Condition In a controlled test environment, the practitioner-client initiates the update of an annotation configuration. The update is sent to the server for processing.

Output/Result The expected result is that the server correctly processes the received annotation configuration from the practitioner-client.

How Test Will Be Performed In the test environment, a simulated annotation configuration update is initiated by the practitioner-client. The system will process this update, and the test will verify that the processing is accurate, ensuring that the server correctly interprets and acts upon the received annotation configuration.

FR-T8 **Type** Functional, Dynamic, Automated

Initial State The server has received and processed the annotation configuration.

Input/Condition The server uses the received annotation configuration to configure machine learning pipelines.

Output/Result The machine learning pipelines are arranged and configured based on the annotation configuration.

How Test Will Be Performed Automated tests will be performed to ensure that the system can configure machine learning pipelines based on annotation configuration. Sample annotation configurations will be used to assess the correct arrangement and setup of machine learning components within the system.

FR-T9 **Type** Functional, Dynamic, Automated

Initial State The machine learning pipelines are configured and active.

Input/Condition The instructor's video stream is processed with the annotation configuration.

Output/Result The instructor's video stream is rendered with accurate annotations.

How Test Will Be Performed Dynamic tests will be conducted to validate the rendering process of the instructor's video stream with accurate annotations. Real or simulated video streams, combined with annotation configurations, will be used to confirm that the system correctly renders the video with the expected annotations.

FR-T10 **Type** Functional, Dynamic, Automated

Initial State The server is actively connected to practitioner-client.

Input/Condition The annotated video stream is generated and ready for transmission.

Output/Result The annotated video stream is transmitted to each practitioner-client through their established connections.

How Test Will Be Performed Automated tests will be employed to assess the transmission of annotated video streams from the server to practitioner clients. The test will involve simulating the generation of annotated video streams and verifying their successful transmission to practitioner-clients through their established connections.

FR-T11 **Type** Functional, Dynamic, Automated

Initial State The signaling server is running.

Input/Condition Signaling requests for WebRTC connections are initiated.

Output/Result The signaling server consistently responds to requests and establishes WebRTC connections.

How Test Will Be Performed Automated tests will be used to evaluate the availability and reliability of the signaling server. These tests will involve monitoring and verifying the responsiveness of the signaling server when requests for WebRTC connections are initiated, ensuring its consistent availability.

FR-T12 **Type** Functional, Dynamic, Automated

Initial State The client application is running, but the user ~~didn't do~~hasn't done any operations yet

Input/Condition The user joining the video stream session.

Output/Result A button to identify if a user is an instructor or a practitioner is rendered.

How Test Will Be Performed Test will be done automatically by unit testing framework in the CI pipeline. The test code will simulate a user's clicks to join a video stream session, then assert the button to identify if a user is an instructor or a practitioner showed up on the user's screen within the next ~~1 second~~MAX_RESPONSE_TIME.

4.2 Tests for Nonfunctional Requirements

NFR-T1 **Type** Functional, Dynamic, Manual

Initial State The system is in a typical operational state with all components and services running, including the user interface.

Input/Condition User interactions such as button clicks and menu selections.

Output/Result The user is able to interact with the client application and understand the response from and results yielded by the system.

How Test Will Be Performed Test will be done manually by developers in coordination with representatives of the intended user groups of different ages and various experiences in practicing Tai Chi. The users will be asked to complete a series of navigations within the user interface and interactions with the system, such as creating, joining, and leaving a video conference session. The user experience will be evaluated by the developers in terms of the completeness of the tasks, the time it takes to complete the tasks, as well as the user's experience in completing the tasks gathered through a short and informal interview. The test is considered successful if ~~the~~ 4 out of 5 developers agree that the overall user experience meets their expectations. Otherwise, the test is considered a failure.

NFR-T2 **Type** Structural, Dynamic, Manual

Initial State The system is in a typical operational state with all components and services running, including the user interface.

Input/Condition User interactions such as button clicks and menu selections.

Output/Result System response to user interactions.

How Test Will Be Performed Test will be done manually by developers running the system. Testers will perform various user interactions, such as button clicks and menu selections, across different areas of the system UI. A test scenario should be designed for each use case to cover all possible interactions within that use case. Use performance monitoring tools to record the response time for each interaction. Repeat each test scenario at least ~~5~~TEST_ACCOUNTS_NUMBERS times for each use case under different system loads to account for variability. Compare the recorded response times against the 1-second requirement. If the system consistently responds within this timeframe for all test scenarios, the test is considered successful. Any deviations beyond ~~1 second~~MAX_RESPONSE_TIME will be considered a failure.

NFR-T3 **Type** Structural, Dynamic, Manual

Initial State The system is in a stable operational state with necessary services and components active. No users are currently logged in.

Input/Condition ~~5~~TEST_ACCOUNTS_NUMBERS test accounts (or more) for users. Predefined user actions/scripts (relevant to normal system interactions during peak periods).

Output/Result The system remains stable and responsive. All user transactions are processed successfully. System performance metrics (such as response time, throughput, and resource utilization) stay within acceptable limits.

How Test Will Be Performed Set up an automated load testing tool and create test scripts mimicking typical user actions during peak periods. This might include logging in, accessing services, executing standard operations, and logging out. Initiate the test by having ~~5~~MIN_NUM_USER_SUPPORTED users (controlled by the load testing software) log into the system simultaneously, executing the predefined scripts that mimic peak load activities. Gradually increase the system load by adding more users (if applicable) to ensure it can handle more than the minimal load without performance degradation. Monitor system performance indicators, focusing on metrics like transaction success rate, response times, and CPU/memory usage. Continuously record system metrics throughout the testing duration. After completing the test, analyze the data to assess whether the system could sustain ~~the~~ a minimum of ~~5~~MIN_NUM_USER_SUPPORTED users during simulated peak conditions without performance degradation.

NFR-T4 **Type** Structural, Dynamic, Manual

Initial State The system is operational, with all services and components running. User accounts for tests are set up, and no tasks are being performed.

Input/Condition Erroneous user input data. Improper user actions.

Output/Result The system detects and rejects invalid inputs or actions, providing clear and helpful error messages to the user. Transactions or actions are either rolled back safely or do not proceed until valid input is provided.

How Test Will Be Performed Identify common user actions and points in the system where user errors could occur. Design a series of test cases that deliberately introduce various user errors. These might include inputting invalid data, using incorrect sequences of commands, or attempting unauthorized actions. Prepare the test environment, ensuring that the system is running and that tools for monitoring system behaviour (like log watchers, debuggers, etc.) are in place. Monitor how the system handles these errors, noting any deviations from expected behaviour, such as system crashes, unclear or missing error messages, incorrect handling, or data corruption. Check system performance indicators to ensure that the errors did not affect the system's stability or consume excessive resources. Document the outcomes of each test case, including the errors introduced, the system's response, and whether the result was as expected.

NFR-T5 ~~**Type** Functional, Dynamic, Manual~~

~~**Initial State** The system is in a stable, operational state with the core functionalities running. No extensions are installed at the outset of the test.~~

~~**Input/Condition** One or more sample extensions ready for installation~~

~~**Output/Result** Extensions are successfully installed on the system. The system recognizes and integrates new extensions' functionalities. System stability and core functionality remain uncompromised after the addition of the extensions.~~

~~**How Test Will Be Performed** Identify or develop sample extensions for the test. These should be varied to represent different types of functionality users might add to the system. Ensure the system is running in a controlled test environment that replicates the end-user environment. Manually add the extensions to the system, following the documented procedures. Observe the installation process for any discrepancies, errors, or issues~~

~~that arise at the same time. After installation, verify that each extension is functioning within the system as intended. Check interoperability with existing features to ensure that there are no conflicts or issues. Test the system's core and extended functionalities to ensure all operate correctly.~~

NFR-T6 **Type** Structural, Dynamic, Automated

Initial State The system, with the SFU component, is in a stable, operational state. Initially, there is a base number of video streams being processed, which is within the SFU's current capacity.

Input/Condition A load testing tool or custom script capable of simulating multiple, simultaneous video streams.

Output/Result The SFU successfully processes an increasing number of video streams.

How Test Will Be Performed Configure a load testing tool to simulate video streams that mimic real-world usage. This could involve setting up dummy user connections each initiating video streams. Run the system with the base number of video streams. Gradually increase the number of simulated video streams being sent through the SFU, starting from the base number and increasing in pre-defined increments. Identify the point(s) at which video quality begins to degrade beyond acceptable levels, or system resources reach critical usage. Collect and analyze data to determine the maximum number of video streams the SFU can handle while maintaining acceptable quality of service.

NFR-T7 **Type** Structural, Dynamic, Manual

Initial State This system is in a typical operational state with all components and services running, the user is in a live session.

Input/Condition Network Interruption/Network Resumption

Output/Result The system attempts to resume the previous session.

How Test Will Be Performed Test will be done manually by developers running the system. Testers will join a live session from either the instructor-client application or the practitioner-client application. Tester will intentionally disrupt the internet connection on the device that the system is running on, wait for ~~30 seconds~~TEST_DISCONNECT_DURATION, and resume the internet connection on the device to simulate an application crash or internet interruption. The tester shall restart the application and

then manually inspect the system UI to see if the user is able to automatically rejoin the previous live session after the network resumption. The test will be repeated ~~for~~ ~~5~~TEST_ACCOUNTS_NUMBERS times on both the instructor-client application and the practitioner-client application under different system loads. The test is considered successful if the user automatically joins the previous live session after application restart when the network resumes. The test will be considered a failure if, after restarting the application, the user is unable to automatically enter a live session or joins the wrong live session.

NFR-T8 **Type** Structural, Dynamic, Manual

Initial State This system is in a typical operational state with all components and services running, the user is in a live session.

Input/Condition Network instability

Output/Result A pop-up window for network instability warning.

How Test Will Be Performed Test will be done manually by developers running the system. Testers will join a live session from either the instructor-client application or the practitioner-client application. The tester will produce network fluctuation/instability on the internet that the system has access to. The tester shall monitor the system UI to inspect for the appearance of network instability warning. The test will be repeated ~~for~~ ~~5~~TEST_ACCOUNTS_NUMBERS times on both the instructor-client application and the practitioner client application under different system loads. The test is considered successful if a pop-up window for network instability warning is displayed after the network fluctuation/instability is created. Otherwise, if no pop up window is displayed to provide a network instability warning, the test will be considered a failure.

NFR-T9 **Type** Structural, Dynamic, Manual

Initial State This system is in a typical operational state with all components and services running.

Input/Condition Turn down the primary signaling server.

Output/Result The redundant server takes over until the primary server resumes.

How Test Will Be Performed Test will be done manually by developers running the system. Tester will intentionally turn down the running pri-

mary signaling server by introducing a controlled fault. Tester shall monitor the system's behaviour during the failover process by observing the annotation delay, video stream quality and system response to user interactions. The test is considered successful if the redundant server takes over within an acceptable time frame, the system continues to operate without any service interruptions, and the system is able to transition back to using the primary server smoothly when restored.

NFR-T10 **Type** Structural, Dynamic, Manual

Initial State The system is in a typical operational state with all components and services running. The user is in a live session with a media-capturing device plugged in.

Input/Condition Disconnect the media-capturing device from the user's computing device.

Output/Result A pop-up window which warns the user that the client application has lost connection to the media-capturing device and prompts the user to reconnect the device to resume the live session.

How Test Will Be Performed The test will be done manually by developers running the system. A tester will make sure a functional media-capturing device is securely connected to the computer. The tester will then run the instructor-client application on the computer, create a live session, and make sure that the application receives and is able to send media streams from the capturing device. The tester will proceed to unplug the media capturing device from the computer. The tester shall monitor the client application and look for any pop up warnings. This test shall be repeated for ~~text~~TEST_ACCOUNTS_NUMBERS times. The test is considered successful if a pop up warning about media-capturing devices being disconnected is displayed every time the media capturing device is disconnected. Otherwise, if no pop up warning is displayed, the test is considered a failure.

NFR-T11 **Type** Structural, Dynamic, Manual

Initial State The system is in a typical operational state with all components and services running.

Input/Condition Connect a media-capturing device to the user's computing device.

Output/Result A pop up window which warns the user that the video capturing device does not meet the required specifications if the client application detects that the video input stream has a resolution lower than MIN_RES.

How Test Will Be Performed The test will be done manually by developers running the system. A tester shall prepare media-capturing devices of various specifications. The tester shall make sure that at least one of them captures video at a resolution below MIN_RES, at least one at MIN_RES, and at least one higher than MIN_RES. The tester shall connect each media-capturing device to the client application. The test is considered successful if a pop up warning about video input not meeting the minimum required resolution is displayed whenever a capturing device with a resolution below MIN_RES. Otherwise, the test is considered a failure.

NFR-T12 **Type** Structural, Dynamic, Manual

Initial State The system is in a typical operational state with all components and services running.

Input/Condition ŧThe user starts a live session and begins streaming.

Output/Result Accurate instructional annotations are rendered and overlaid on the instructor's video stream.

How Test Will Be Performed The test will be done manually by developers running the system. The tester will run the instructor-client application on a computer, create a live session, and make sure that the application receives and is able to send media streams from the capturing device. The tester shall then monitor the annotations produced and rendered by the annotating machine learning pipeline, export samples of the results into local media files, and ask other developers in the team to review and evaluate the annotated videos. The test is considered successful if 4 out of 5 developers are satisfied with the result. Otherwise, the test is considered a failure.

NFR-T13 **Type** Structural, Dynamic, Manual

Initial State The system is in a typical operational state with all components and services running.

Input/Condition Connect more than one media-capturing device to the user's computing device.

Output/Result A ~~dialog~~dialogue listing all the detected capturing devices is displayed to prompt the user ~~for selecting~~to select a device to use.

How Test Will Be Performed The test will be done manually by developers running the system. A tester shall prepare media-capturing devices of various types. The tester shall make sure that all prepared capturing devices are functional and establish secure connections to the computer running the client application. The test is considered successful if all media-capturing devices are listed and selectable in a ~~dialog~~dialogue in the client application. Upon selecting any of the listed devices, the client application would switch to the selected device and fetch media streams from the device. Failing ~~in completing~~to complete any of the described interactions would be considered a failure of this test case.

NFR-T14 **Type** Structural, Dynamic, Manual

Initial State This system is in a typical operational state with all components and services running, with a live session created.

Input/Condition The live video and audio stream from instructor-client application.

Output/Result Audio stream and video stream with annotations on practitioner-client application.

How Test Will Be Performed Test will be done manually by developers running the system. A tester will run the instructor-client application, create a live session, and start streaming using media-capturing devices for 30 minutes. User performance monitoring tools to measure and record the delay between the video stream frames and annotation frames. The test will be repeated ~~for~~ at least ~~5~~TEST_ACCOUNTS_NUMBERS times under different system loads. The test is considered successful if the recorded delay remains under MAX_DELAY through each live session. Any deviations beyond MAX_DELAY will be considered a failure.

NFR-T15 **Type** Structural, Static, Manual

Initial State The system is in a typical operational state with all components and services running.

Input/Condition The user starts the client application for instructors.

Output/Result A message with detailed instructions to set up a media capturing device and a list of cautions is displayed. The user is informed that

for generating accurate annotations, their full body must be within the field of view from the media-capturing device.

How Test Will Be Performed The test will be done manually by developers running the system. A tester launches the client application for instructors and looks for any instructional messages or cautions. The test is considered successful if the tester is able to properly set up the media-capturing device by strictly following the instructions displayed, such that their entire body is visible from the perspective of the capturing device. Otherwise, if any problem or confusion occurs during the setup process, the test is considered a failure.

NFR-T16 **Type** Structural, Dynamic, Manual

Initial State The software system is fully developed, stable, and ready for testing. The different test environments for the latest versions of Windows, Linux, and macOS are set up, each with default settings.

Input/Condition The latest stable versions of Windows, Linux, and macOS. Test cases are designed to cover all the main functionalities of the system.

Output/Result The system functions correctly on all mentioned operating systems without crashes, unexpected behaviour, or significant performance issues.

How Test Will Be Performed Set up test environments with the latest versions of Windows, Linux, and macOS, ensuring they meet the system requirements for the software. Prepare a suite of test cases that cover the full range of the system's functionality. These should include common user tasks and interactions with the system. Install or deploy the system in each test environment. Execute the suite of test cases in each operating system environment, documenting any discrepancies in functionality, performance, or user interface issues specific to each platform. Compare results across the different operating systems to ensure that features and functionalities are consistent. Validate that performance is consistent across different operating systems and that there are no platform-specific lags or speed issues.

NFR-T17 **Type** Structural, Dynamic, Manual

Initial State The system is fully developed, with all features operational, and is hosted in a stable environment accessible via the web. Test environments with the latest versions of Chrome, Firefox, Safari, and Edge are established, each configured with default browser settings.

Input/Condition Latest stable versions of Chrome, Firefox, Safari, and Edge. A series of test cases are designed to fully evaluate the functionalities and features of the system.

Output/Result The system operates as intended on all tested web browsers without significant functionality issues, crashes, or severe performance degradation. Features and visual elements are consistent across all browsers.

How Test Will Be Performed Prepare detailed test cases covering all aspects of system functionality, including user interface, input fields, navigation, performance, and security aspects. Ensure the availability of the latest versions of each browser and that each test environment reflects the end-user's setting as closely as possible. In each browser, manually execute all test cases, performing all the actions an end-user would do. This includes testing visual elements, interactive components, and backend functionality (like form submission, data processing, etc.) for consistency. Compare results for each browser to check for consistency in feature behaviour and data processing. Validate response times and performance metrics to ensure there are no browser-specific lags or speed issues.

NFR-T18 **Type** Structural, Dynamic, Manual

Initial State The system is fully developed and operational. The testing environments represent a range of standard personal computer and laptop configurations (covering various manufacturers, system specifications, and age of devices) equipped with cameras and, where applicable, microphones. The devices are running compatible operating systems with the necessary drivers installed.

Input/Condition Range of standard computers or laptops with various specifications, but all within the commonly accepted 'standard' range for current users. Devices have functional cameras and optional microphones. Test script detailing system operation tasks.

Output/Result The system operates without significant delays, errors, or crashes.

How Test Will Be Performed Identify a range of standard personal computers and laptops that reflect the typical user base's hardware. This range should include variations in processor speed, memory, age, and manufacturer. Ensure that each test machine is equipped with a functional camera and, if necessary for the system's functionality, a microphone. Install or access the system on each device, ensuring proper installation

of peripheral drivers and any necessary system software. Prepare a test script that encompasses everyday tasks users would perform on the system, ensuring it thoroughly tests interaction with camera and microphone functions. Execute the test script on each device, interacting with the system as an ordinary user would. This process should include tasks that specifically utilize the camera and microphone to test the system's handling of these resources. Observe and document system performance, noting any lag, stuttering, quality issues with audio/video, crashes, or errors. Review the quality of the audio and video captured or utilized by the system, checking for any delays, synchronization issues, or loss of quality.

NFR-T19 **Type** Structural, Static, Manual

Initial State The system is fully developed, with comprehensive documentation on its architecture, codebase, dependencies, and update procedures. The development environment is available for testing maintenance procedures, including tools for version control, testing, and deployment.

Input/Condition Simulated or real maintenance tasks.

Output/Result Successful completion of maintenance tasks without introducing significant issues or disruptions to the system's operation.

How Test Will Be Performed Begin with a comprehensive review of the system documentation, focusing on the clarity, completeness, and accuracy of information related to system architecture, dependencies, and procedures for updates and maintenance tasks. Analyze the system's codebase for adherence to coding standards, modularity, use of dependencies, commenting, and documentation within the code. Identify any areas of technical debt, outdated dependencies, or overly complex structures that could complicate maintenance efforts. Simulate various maintenance scenarios, such as applying an update, fixing a bug, or adding a new feature. This process should cover a range of complexity and potential impact on different parts of the system. Monitor the ease with which these tasks can be completed, the risk of introducing errors, and the effectiveness of the documentation and procedures. Test the system's ability to roll back updates by simulating a scenario where an update introduces a significant issue. Evaluate the effectiveness and ease of reversing changes without losing data or causing system downtime.

NFR-T20 ~~**Type** Structural, Dynamic, Manual~~

Initial State The system is fully operational, and a maintenance/update plan is ready, detailing the procedures to be followed. A stable pre-update version of the system is running, and a controlled environment is available to simulate the maintenance process without affecting live users.

Input/Condition Tools and resources necessary for the maintenance task (e.g., update packages, scripts, database migrations, etc.).

Output/Result The total time taken for the procedure is recorded, from initiation to completion, including the system being fully operational post-update.

How Test Will Be Performed Prepare the test environment to match the live system as closely as possible in terms of data, configurations, and system dependencies. Ready the maintenance/update package along with detailed procedures and any scripts or commands necessary for the process. Begin the timed maintenance process, initiating the update procedure as documented. A timer or time log should be used to record the process's duration accurately. Perform the steps required for the update, which may include system shutdown, backup, application of update packages, database migrations, system configuration adjustments, and restart procedures. Document each step, noting the time taken and any complications or deviations from the expected process. Once the update steps are completed, perform a comprehensive system check to verify that all functionalities are operational and that the update has been applied successfully. Analyze the outcomes of the update, including any improvements or regressions in system performance. Review the time log to calculate the total duration of the maintenance/update process. This calculation should encompass all stages, from the initial shutdown to the restoration of full functionality.If the process took longer than the specified 4-hour window, identify the stages that consumed more time than expected and analyze the reasons for this delay.

NFR-T21 **Type** Structural, Dynamic, Automated

Initial State The system is fully operational, and user accounts containing mock personal information are available for testing. Security measures, such as encryption protocols and compliance measures, are in place.

Input/Condition Automated security testing tools or penetration testing tools for vulnerability scanning. Test scripts are designed to simulate typical user interactions, including data input and retrieval.

~~Output/Result~~ ~~Confirmation that stored data, especially personal information, is encrypted or securely hashed within the database or storage system.~~

~~How Test Will Be Performed~~ ~~Use network monitoring tools to capture the data being transmitted during the test sessions. This activity might include creating user accounts, logging in, or transmitting personal information. Analyze the captured data to verify that it's encrypted and that sensitive information isn't exposed in plaintext or easily decipherable formats. Then access the storage system (database, file storage) and inspect how user data, particularly personal information, is stored. Verify that sensitive data is encrypted or hashed, making it unreadable without proper authorization or decryption keys. Implement automated security tools to scan for vulnerabilities that could be exploited to gain unauthorized access to sensitive data. These might include outdated software, misconfigurations, or known vulnerabilities within the system components. Document all findings from the security tests, including any potential vulnerabilities identified, successful and failed breach attempts, and the security status of data transmission and storage.~~

NFR-T22 **Type** Structural, Static, Manual

Initial State The system is in a stable state, ready for testing. The user interface for account creation or any other information input is accessible, and documentation related to data handling is available.

Input/Condition Guidelines or criteria specifying what constitutes "essential" versus "nonessential" information for the system's purpose. Access to the system's user interfaces (UI) where data submission forms are present.

Output/Result Comparison of requested data against the criteria for essential information.

How Test Will Be Performed Examine all user interfaces (UIs) that prompt data entry forms used during regular operations. Based on the system's purpose and functionality, classify the information that the system needs to operate effectively. Define what is considered "essential" information that the system cannot function without. Identify any information that may be deemed "non-essential," meaning that its collection is not justified based on the system's core functions. Compare the data collected from the entry with the classifications. Highlight any data fields that appear to be non-essential for the system's operation.

NFR-T23 **Type** Structural, Dynamic, Manual

Initial State This system is in a typical operational state with all components and services running.

Input/Condition Attempted unauthorized modifications to the system including access to system files and modifications to system configurations.

Output/Result A determination of whether the system are able to prevent access or modifications from unauthorized users.

How Test Will Be Performed Test will be done manually by developers running the system. Tester will simulate an unauthorized user of the system and make the following attempts:

- Access system settings.
- Modify Critical system configuration files.
- Gain access to media streams in the system.

The test will be performed on both the instructor-client application and practitioner-client application. The test is considered successful if all attempts by the unauthorized user are prevented by the system. The test will be considered a failure if any successful unauthorized access or modification to the system ~~were~~is made during the testing process.

NFR-T24 **Type** Structural, Dynamic, Manual

Initial State The system is in a typical operational state with all components and services running.

Input/Condition The user starts a live session and begins streaming.

Output/Result A prompt for granting access to use the media-capturing devices is displayed.

How Test Will Be Performed Test will be done manually by developers running the system. A tester will make sure a functional media-capturing device is securely connected to the computer. The tester will then run the instructor-client application on the computer. Upon creating a live session, the tester shall look for a prompt and they are able to consent to giving the client application access to the media-capturing device. If the video stream starts, or the media stream from the capturing device is displayed on the screen before the tester's consent, the test is considered a failure. Otherwise, the test is considered successful.

NFR-T25 **Type** Structural, Dynamic, Manual

Initial State The system is in a typical operational state with all components and services running.

Input/Condition The user starts a live session and begins streaming.

Output/Result An indicator that the media-capturing device is in use and that the user is being recorded is visible in the client application.

How Test Will Be Performed Test will be done manually by developers running the system. A tester will make sure a functional media-capturing device is securely connected to the computer. The tester will then run the instructor-client application on the computer, create a live session, and make sure that the application receives and is able to send media streams from the capturing device for 30 minutes. The tester shall look for an indicator that the media-capturing device is in use. If such an indicator cannot be found, or if the indicator disappears at any moment of time during the whole testing process, the test is considered a failure. Otherwise, the test is considered successful.

NFR-T26 **Type** Structural, Dynamic, Manual

Initial State The system is in a typical operational state with all components and services running.

Input/Condition The user ends a live session and stops streaming.

Output/Result The indicator that the media-capturing device is in use disappears. The media stream from the capturing device is no longer displayed on the screen.

How Test Will Be Performed Test will be done manually by developers running the system. A tester will make sure a functional media-capturing device is securely connected to the computer. The tester will then run the instructor-client application on the computer, create a live session, and make sure that the application receives and is able to send media streams from the capturing device. The tester shall then manually end the live stream session. The tester shall look for the indicator that the media capturing device is in use. If such an indicator is no longer visible and the media stream from the capturing device is not displayed on screen, the test is considered successful. Otherwise, the test is considered a failure.

NFR-T27 **Type** Structural, Static, Manual

Initial State This system is in a typical operational state with all components and services running.

Input/Condition User inputs

Output/Result System response to user inputs

How Test Will Be Performed Test will be done manually by developers running the system. Testers will perform various user interactions that cover all possible use cases of the system and inspect if any offensive cultural symbols/language are displayed on the UI. The test is considered successful if all testers agree that there is no offensive cultural symbol/language displayed through the testing process.

NFR-T28 **Type** Structural, Static, Manual

Initial State The system is in a typical operational state with all components and services running.

Input/Condition A set of ISO/IEC 12207 standards.

Output/Result A determination of whether the system and its associated processes, documentation, and activities comply with the ISO/IEC 12207 standards.

How Test Will Be Performed Test will be performed by: 1. Testers carefully review all project documentation, including software development processes, design documentations, coding standards and testing plans to ensure they all align with the ISO/IEC 12207 standards. 2. Testers examine the processes and activities involved in the software development life cycle to determine if they adhere to the ISO/IEC 12207 standards. The test is considered successful if all testers agree that the system and its associated processes, documentation, and activities comply with the ISO/IEC 12207 standards.

NFR-T29 **Type** Structural, Dynamic, Manual

Initial State This system is in a typical operational state with all components and services running.

Input/Condition Various user interactions.

Output/Result The computers continue to function within acceptable performance parameters throughout the test.

How Test Will Be Performed Test will be done manually by developers running the system. Testers will perform various user interactions, such as button clicks and menu selections, and joining/creating live sessions. The input user interactions shall cover all possible use cases of the system.

Repeat the test on different computing devices. The tester shall monitor the CPU temperature of the computer that the system is running on. The test is considered successful if the CPU temperature of the computer that the system is running on remains under MAX_TEMP through the test. Otherwise, the test is considered a failure.

NFR-T30 **Type** Structural, Static, Manual

Initial State This system is in a typical operational state with all components and services running.

Input/Condition User inputs

Output/Result System response from user inputs

How Test Will Be Performed Test will be done manually by developers running the system. Testers will perform various user interactions, such as button clicks and menu selections, and joining/creating live sessions. Testers will use the system as Tai Chi instructors and Tai Chi practitioners and evaluate the usability and user experience of the system. The test is considered successful if all testers agree that the system does not affect their physical or mental health. Otherwise, if any tester states that the system harms their physical or mental health, the test is considered a failure.

4.3 Traceability Between Test Cases and Requirements

Please refer to Table 4, Table 5, and Table 6.

	FR1	FR2	FR3	FR4	FR5	FR6	FR7	FR8	FR9	FR10	FR11	FR12	FR13	FR14	FR15	FR16
FR-T1	X	X														
FR-T2			X													X
FR-T3				X	X											
FR-T4						X									X	
FR-T5								X								
FR-T6							X									
FR-T7									X							
FR-T8										X						
FR-T9											X					
FR-T10												X				
FR-T11													X			
FR-T12														X		

Table 4: Traceability matrix showing the connections between test cases and functional requirements

	LF1	UH1	UH2	PR1	PR2	PR3	PR4	PR5	PR6	PR7	PR8	PR9	PR10	PR11	PR12	PR13
NFR-T1	X	X	X													
NFR-T2				X	X											
NFR-T3					X	X										
NFR-T4							X									
NFR-T5								X								
NFR-T6									X							
NFR-T7										X						
NFR-T8											X					
NFR-T9												X				
NFR-T10													X			
NFR-T11														X		
NFR-T12															X	
NFR-T13																X

Table 5: Traceability matrix showing the connections between test cases and non-functional requirements

	PR14	PR15	OE1	OE2	OE3	MS1	MS2	SR1	SR2	SR3	SR4	SR5	SR6	CR1	LR1	HS1	HS2
NFR-T14	X																
NFR-T15		X															
NFR-T16			X														
NFR-T17				X													
NFR-T18					X												
NFR-T19						X											
NFR-T20							X										
NFR-T21								X									
NFR-T22									X								
NFR-T23										X							
NFR-T24											X						
NFR-T25												X					
NFR-T26													X				
NFR-T27														X			
NFR-T28															X		
NFR-T29																X	
NFR-T30																	X

Table 6: Traceability matrix showing the connections between test cases and non-functional requirements continued

5 Unit Test Description

This section provides a detailed description of the guidelines for conducting unit testing of the system.

5.1 Unit Testing Scope

The primary focus of the unit tests in this project will be on assessing the functionality of individual components. To ensure extensive coverage, test coverage metrics will be employed to quantify and oversee testing activities. Integration of unit tests into the Continuous Integration (CI) pipeline will guarantee that new modifications do not disrupt existing functional components.

The following modules are considered out of scope for unit testing:

ML Model Accuracy This exclusion is due to the decision to employ an external model.

Client Code Accessing User's Device Hardware This aspect will be managed by the WebRTC framework, thus exempting it from unit testing.

Network Connections between Components Testing of network connections is beyond the scope of unit testing.

More specifically, the majority of non-functional requirements exclude verification via unit testing. Non-functional testing methodologies tend to qualify as performance testing and preclude unit testing validation.

All the parts that are not out of scope for testing should be unit-tested.

6 Appendix

6.1 Usability Survey Questions

Rating scales are defined as:

1 Well below expectations

2 Below expectations

3 Met expectations

4 Above expectations

5 Beyond expectations

Questions for the user experience survey:

1. On a scale of 1 to 5 ~~(beyond expectation)~~, how would you rate your overall experience with our Tai Chi video conferencing application?
2. On a scale of 1 to 5 ~~(beyond expectation)~~, how would you rate the experience in reading the in-app instructions and navigating through the application?
3. On a scale of 1 to 5 ~~(beyond expectation)~~, how would you rate the video conference quality, considering the clarity, latency, and the accuracy of annotation overlays?
4. On a scale of 1 to 5 ~~(beyond expectation)~~, how would you rate your overall Tai Chi learning experience with the aid of the application?
5. Are there any specific issues or challenges you've encountered while using the application?
6. How satisfied are you with the speed and responsiveness of the application during your Tai Chi sessions?
7. Do you find that the annotations and real-time guidance provided by the application enhance your Tai Chi learning experience?
8. Have you experienced any disruptions or crashes during your sessions? If yes, please describe.

9. Have you ever received any notifications or warnings related to the security of your sessions, e.g., when media-capturing devices are accessed?

10. Have you received sufficient guidance on using the application effectively for your Tai Chi practice?

11. Are there any specific content or learning materials you would like to see added to the application?

Appendix — Reflection

Anhao I am responsible for coming up with the test plan for functional requirements. During the test plan development, I learned about various of testing methods, such as Dynamic testing - Black-box testing - Functional testing, Unit testing and Integration testing. I had past experience with others but integration testing. As integration testing involves evaluating the interaction and interoperability of different system components, I need to learn more about how to conduct integration testing on our system.

To learn more about integration testing, I will start by understanding its fundamental concepts, including assessing interactions between system components. Identify integration points, define test scenarios that mimic real-world usage, and choose appropriate testing techniques. Develop detailed test cases and set up a test environment mirroring the production setup. Execute tests, monitor interactions, and assess system behaviour. Analyze results, document findings, and address integration issues, retesting scenarios after resolution. After knowing the details of integration tests, then I can apply it effectively to ensure component interactions in the system work harmoniously.

Xunzhou My primary focus while composing this verification and validation plan is to ensure that the product we develop meets the performance and usability requirements. The overall user experience of a software product is significantly influenced by its performance and usability. Planning out test cases under various use case scenarios provides us with an opportunity to view the product from a different perspective and critically analyze all the ways in which the system might fail.

The other part of the document for which I am responsible is the software validation plan. The key lesson I've learned from crafting this section is the crucial distinction between software validation and software verification. Verification is the process of confirming whether the software is being developed correctly and whether it adheres to its specified design and development requirements. In contrast, validation is the process of evaluating the final software product to ensure that it fulfills its intended purpose and meets the needs of the end-users. Keeping this distinction in mind, I have been able to create a focused, feasible, and comprehensive plan for validating our product.

Reflecting on the overall work completed in this document, the two portions I've worked on are indeed closely related to each other. User satisfaction serves as a solid indicator of valid software. A software application that performs well and is easy to use is more likely to satisfy and retain users.

Kehao I was primarily focused on the documentation aspect, which involved developing the design verification plan, the V&V plan verification plan and system tests for non-functional requirements. These tasks exposed me to an opportunity to gain insights into the in-depth details of quality assurance for a software system.

While working on the system tests for non-functional requirements, I delved into the performance and security aspects of our system. This was an enriching learning experience as I had to envision various usage scenarios and systematically design test cases to evaluate the system's performance and ~~behavior~~behaviour under different conditions. It was during this process that I realized the critical role that the system's performance and security play in shaping the overall user experience. Meanwhile, the development of the design verification plan and the V&V plan verification plan taught me valuable lessons about the importance of thoroughly verifying that our software adheres to its intended design and development requirements.

Reflecting on my work, I've realized the interconnectedness of these documentation tasks. The design verification plan ensures that our system is built according to the design, which, in turn, has a direct impact on the system tests for non-functional requirements. Through the process of developing these documents, I gained a deeper understanding of the importance of planning, verification, and validation in delivering a software product ~~aht~~that not only meets design specifications but also satisfied the ~~use-users~~users by performing effectively and reliably.

To further enhance the related skills, I plan to take proactive steps in the future. First, I will seek opportunities to engage in more real-world projects and interact closely with the quality assurance team to acquire hands-on experience. Also, I will explore relevant articles, papers and courses about software testing and quality assurance in order to gain theoretical knowledge and stay updated with the industry best practices.

Qi I am tasked with developing the test plan for the functional requirements, leveraging my proficiency in various functional testing tools. I have observed that certain ambiguities within our requirements become apparent during the formulation of the verification plan. This realization brings to mind the concept of Test Driven Development, wherein creating unit tests prior to coding offers a precise assessment of the actual implementation. In our context, this practice enhances the clarity of our functional requirements, serving as a guiding principle for our development process.

Throughout the formulation of this Verification and Validation (V&V) plan, I have come to understand the necessity of not only ensuring the correctness of the code through testing but also devising strategies to validate whether our requirements have been met. Additionally, establishing corresponding metrics is vital for enhancing user

experiences post-software development.

While I lack familiarity with certain tools specific to WebRTC testing, I view this as an opportunity for learning and skill development. Regarding the tools we have chosen, there is a wealth of online tutorials and forums available to acquire the requisite knowledge and skills. Furthermore, the utilization of generative AI proves to be a valuable resource when troubleshooting challenges.

Qianlin In the process of our recent project documentation, my primary role was devising testing methods for certain non-functional requirements. Given the intrinsic characteristics of non-functional requirements, many of our tests leaned heavily towards manual procedures and user input. This pivot towards manual processes underscored the importance of physical testing sessions, ensuring our application ticked all the boxes in terms of requirements. However, the challenge wasn't just about conducting tests, but doing so efficiently and effectively. This means considering aspects like identifying the right testing protocols for post-development app testing. Moreover, I had a deeper understanding of non-functional requirements testing through the documentation procedure. While my foundational knowledge serves me well, there's a vast ocean of methodologies and best practices that await exploration. Further, the troves of open-source projects on platforms like GitHub offer another layer of knowledge, especially when it comes to deciphering testing reports. Furthermore, I'll dedicate time to analyze open-source projects for practical insights. Given my role, I've decided to prioritize test planning by enrolling in a specialized online tutorial, ensuring our testing processes are both thorough and efficient.