System Verification and Validation Plan for Software Engineering

Team #22, TeleHealth Insights
Mitchell Weingust
Parisha Nizam
Promish Kandel
Jasmine Sun-Hu

November 4, 2024

Revision History

Table 1: Revision History

Date	Vers.	Contributors	Notes
10/30/24	1.0	Mitchell Weingust	Added: • 3.1 Verification and Validation Team • 3.4 Verification and Validation Plan • 6.2 Usability Survey Questions
10/30/24	1.0	Promish Kandel	Added: • 3.6 Automated Testing and Verification Tools • 3.7 Software Validation Plan • 4.1 Tests for Functional Requirements • 4.2 Tests for Nonfunctional Requirements
11/01/24	1.1	Jasmine Sun-Hu	Added: • 3.2 SRS Validation Plan • 4.1 Data Processing and Display • 4.2 Look and Feel FR tests • 4.2 Maintainence and Support NFR tests • 4.2 Cultural NFR tests
11/02/24	1.2	Mitchell Weingust	Added: • 4.1 Authentication FR tests • 4.2 Usability and Humanity NFR tests • 4.2 Operational and Environmental NFR tests
11/04/24	1.3	Mitchell Weingust	Added: • Reflection
11/04/24	1.4	Jasmine Sun-Hu	Added: • 6.1 Symbolic Parameters • Reflection
11/04/24	1.5	Promish Kandel	Added: • section 1 Symbols, Abbreviations and Acronyms • Reflection

[The intention of the VnV plan is to increase confidence in the software. However, this does not mean listing every verification and validation technique that has ever been devised. The VnV plan should also be a **feasible** plan. Execution of the plan should be possible with the time and team available. If the full plan cannot be completed during the time available, it can either be modified to "fake it", or a better solution is to add a section describing what work has been completed and what work is still planned for the future. —SS]

[The VnV plan is typically started after the requirements stage, but before the design stage. This means that the sections related to unit testing cannot initially be completed. The sections will be filled in after the design stage is complete. the final version of the VnV plan

should have all sections filled in. —SS]

Contents

List of Tables

List of Figures

[Remove this section if it isn't needed —SS]

1 Symbols, Abbreviations, and Acronyms

symbol	description
SRS	Software Requirements Specification
VnV	Verification and Validation
PII	Personally Identifiable Information

[symbols, abbreviations, or acronyms — you can simply reference the SRS (Author, 2019) tables, if appropriate —SS]

[Remove this section if it isn't needed —SS]

This document ... [provide an introductory blurb and roadmap of the Verification and Validation plan —SS]

2 General Information

2.1 Summary

[Say what software is being tested. Give its name and a brief overview of its general functions. —SS]

2.2 Objectives

[State what is intended to be accomplished. The objective will be around the qualities that are most important for your project. You might have something like: "build confidence in the software correctness," "demonstrate adequate usability." etc. You won't list all of the qualities, just those that are most important. —SS]

[You should also list the objectives that are out of scope. You don't have the resources to do everything, so what will you be leaving out. For instance, if you are not going to verify the quality of usability, state this. It is also worthwhile to justify why the objectives are left out. —SS]

[The objectives are important because they highlight that you are aware of limitations in your resources for verification and validation. You can't do everything, so what are you going to prioritize? As an example, if your system depends on an external library, you can explicitly state that you will assume that external library has already been verified by its implementation team. —SS]

2.3 Challenge Level and Extras

[State the challenge level (advanced, general, basic) for your project. Your challenge level should exactly match what is included in your problem statement. This should be the challenge level agreed on between you and the course instructor. You can use a pull request to update your challenge level (in TeamComposition.csv or Repos.csv) if your plan changes as a result of the VnV planning exercise. —SS]

[Summarize the extras (if any) that were tackled by this project. Extras can include usability testing, code walkthroughs, user documentation, formal proof, GenderMag personas, Design Thinking, etc. Extras should have already been approved by the course instructor as included in your problem statement. You can use a pull request to update your extras (in TeamComposition.csv or Repos.csv) if your plan changes as a result of the VnV planning exercise. —SS]

2.4 Relevant Documentation

[Reference relevant documentation. This will definitely include your SRS and your other project documents (design documents, like MG, MIS, etc). You can include these even before they are written, since by the time the project is done, they will be written. You can create BibTeX entries for your documents and within those entries include a hyperlink to the documents. —SS]

Author (2019)

[Don't just list the other documents. You should explain why they are relevant and how they relate to your VnV efforts. —SS]

3 Plan

This section includes plans for how the team's verification and validation processes will be implemented. This covers the team and its roles, the SRS verification plan, the design verification plan, the verification and validation plan verification plan, the implementation verification plan, automated testing and verification tools, and the software validation plan.

3.1 Verification and Validation Team

Name	Roles and Responsibilities
Mitchell Weingust	• Audio Analysis Model verification • System Architecture and Design Validation • SRS Verification
Parisha Nizam	• Frontend Interface Verification • Backend Database Verification • VnV Verification
Promish Kandel	• Frontend Interface Verification • Video Analysis Model verification • VnV Verification
Jasmine Sun-Hu	• Backend Database Verification • System Architecture and Design Validation • SRS Verification
Dr. Irene Yuan	• Providing feedback (including Hands-On) during project development
Dr. Yao Du	• Providing written feedback on user experiences and testing
Chris Schankula	• Providing feedback during project development • Revision recommendations

Table 2: Verification and Validation Team Table

3.2 SRS Verification Plan

[List any approaches you intend to use for SRS verification. This may include ad hoc feedback from reviewers, like your classmates (like your primary reviewer), or you may plan for something more rigorous/systematic. —SS]

[If you have a supervisor for the project, you shouldn't just say they will read over the SRS. You should explain your structured approach to the review. Will you have a meeting? What will you present? What questions will you ask? Will you give them instructions for a task-based inspection? Will you use your issue tracker? —SS]

[Maybe create an SRS checklist? —SS]

The following approaches will be used to verify the SRS:

- **Ad hoc Peer Review**: Informal reviews of the SRS will be conducted after every major revision to the SRS with classmates who will serve as peer reviewers. This will provide new perspectives that can help identify ambiguities, missing requirements and/or areas of improvement.
  - Peer reviewers will submit feedback using GitHub issue tracker for organized task assignment and tracking.
- **Supervisor Review**: Before every major SRS revision submission, the team will send a copy of the SRS to the project supervisor a week before meeting, along with a checklist highlighing priority areas for their feedback. The team will also prepare questions about the requirements related to interfaces and usability which is the supervisor's area of expertise. During the review meeting, the team will first review the supervisor's initial feedback with the supervisor, and then ask the prepared questions.
  - The meeting notes will be documented using GitHub issue tracker.
- **Internal Team Walkthrough**: The team will hold a collaborative session before every major SRS revision submission where the team will discuss each section one by one to verify that all requirements are understood by all members, and that each section is consistent with the project goals/objectives. A checklist will act as a guide to highlight the key concepts the review should focus on, and will help ensure no critical areas are missed in review. The checklist can be found below.
  - Any corrections or modifications that need to be made will be noted in the team meeting's GitHub issue tracker.

The following checklist will be used for the internal team walkthrough:

- ☐ Are all major functions required for the website (interface, backend, recording, analysis, storage) covered?
- ☐ Does each function have clear input and output specifications?
- ☐ Are all requirements written with consistent terminology?

- ☐ Do all requirements avoid conflict with each other?
- ☐ Does each requirement avoid ambiguous language?
- ☐ Are all requirements verifiable and testable?
- ☐ Is each requirement written in a way all teawm members and stakeholders can understand?
- ☐ Are any assumptions about user behavior or system behaviour explicity stated?

3.3 Design Verification Plan

[Plans for design verification —SS]

[The review will include reviews by your classmates —SS]

[Create a checklists? —SS]

3.4 Verification and Validation Plan Verification Plan

As the verification and validation plan is an artifact, it must be verified too. The team's verification of the VnV plan follows:

- Peer reviews by classmates, including other teams' peer reviews, to identify areas of improvement and general feedback
- Documentation review by the project's supervisor, Dr. Irene Ye Yuan, to ensure that the team's planned verification and validation plan is realistic and feasible
- Teammate documentation reviews, to provide critical feedback and ensure that all intended goals and outcomes are met
- Mutation testing to ensure that changes to aspects of the source plan can be detected by test cases.

The below checklist will be used, in addition, to ensure the team's VnV plan is correct and complete.

- ☐ Does the VnV Plan verify all functional requirements are met?
- ☐ Does the VnV Plan verify all non-functional requirements are met?
- ☐ Have all peer-review issues been addressed and closed?
- ☐ Have all members of the Verification and Validation Team contributed to the review and approved the document?
- ☐ Are all aspects of the system boundary being verified, validated, and tested?
- ☐ Do the system tests cover all requirements mentioned in the SRS?
- ☐ Did the test cases detect mutations and give desired outputs?
- ☐ Did the test cases' expected output match the actual output?

- ☐ Is there a process for documenting and resolving defects?

[The verification and validation plan is an artifact that should also be verified. Techniques for this include review and mutation testing. —SS]

[The review will include reviews by your classmates —SS]

[Create a checklists? —SS]

3.5 Implementation Verification Plan

[You should at least point to the tests listed in this document and the unit testing plan. —SS]

[In this section you would also give any details of any plans for static verification of the implementation. Potential techniques include code walkthroughs, code inspection, static analyzers, etc. —SS]

[The final class presentation in CAS 741 could be used as a code walkthrough. There is also a possibility of using the final presentation (in CAS741) for a partial usability survey. —SS]

3.6 Automated Testing and Verification Tools

[What tools are you using for automated testing. Likely a unit testing framework and maybe a profiling tool, like ValGrind. Other possible tools include a static analyzer, make, continuous integration tools, test coverage tools, etc. Explain your plans for summarizing code coverage metrics. Linters are another important class of tools. For the programming language you select, you should look at the available linters. There may also be tools that verify that coding standards have been respected, like flake9 for Python. —SS]

[If you have already done this in the development plan, you can point to that document. —SS]

[The details of this section will likely evolve as you get closer to the implementation. —SS]

The following are the automated testing and verification tools to be used during the validation and verification process for the software being tested in this VnV plan:

- Unit Tests: Jest, Pytest
- Linters: Flake9, Prettier, ESLint
- Continuous Intergration: GitHub Actions

For our code coverage, we will use Istanbul and Coverage.py. Istanbul is a code coverage tool that works with JavaScript testing frameworks like Jest. It helps developers see how much of their code is tested by creating reports that show untested lines and functions. Coverage.py is a code coverage tool for Python, which we use for our machine learning model. It measures how much of the code runs during tests and generates reports in different formats, helping developers find untested parts of their Python applications and improve test coverage.

3.7 Software Validation Plan

[If there is any external data that can be used for validation, you should point to it here. If there are no plans for validation, you should state that here. —SS]

[You might want to use review sessions with the stakeholder to check that the requirements document captures the right requirements. Maybe task based inspection? —SS]

[For those capstone teams with an external supervisor, the Rev 0 demo should be used as an opportunity to validate the requirements. You should plan on demonstrating your project to your supervisor shortly after the scheduled Rev 0 demo. The feedback from your supervisor will be very useful for improving your project. —SS]

[For teams without an external supervisor, user testing can serve the same purpose as a Rev 0 demo for the supervisor. —SS]

[This section might reference back to the SRS verification section. —SS]

Our plan for validating the software includes review sessions with stakeholders and extensive user testing. Dr.Yao Du, one of our key stakeholders, will provide the software to clinicians and patients for real-world testing in clinical settings. This will allow us to gather valuable feedback on the software's functionality and usability in actual healthcare environments. In addition to this field testing, we will conduct structured user testing sessions where participants will simulate the experience of being a patient. During these sessions, users will navigate through the software, interacting with its features, and afterward, they will share their insights on what worked well and what didn't. Overall, using both field testing and targeted user testing, the information gathered will help us refine the software and ensure it meets the needs of its intended users.

4 System Tests

This section includes systems tests for functional requirements and nonfunctional requirements mentioned in the SRS. For traceability, the team created a traceability matrix to correlate the system tests to the requirement(s) they cover.

4.1 Tests for Functional Requirements

[Subsets of the tests may be in related, so this section is divided into different areas. If there are no identifiable subsets for the tests, this level of document structure can be removed. —SS]

[Include a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good here. —SS]

4.1.1 Area of Testing1

4.1.2 Authentication

[It would be nice to have a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good here. If a section covers tests for input constraints, you should reference the data constraints table in the SRS. —SS]

The test cases below focus on ensuring users can safely and securely login, create and access their accounts without worrying about others accessing their information.

- FR-ST-A1

  Control: Manual

  Initial State: User has a Parent account already created and stored in the database

  Input: Selection of Parent account role for login

  Output: The expected result is the Parent account role is selected and User is brought to the Parent login screen

  Test Case Derivation: The expected output is justified based on FR-A1 in the SRS document

  How the test will be performed:

  1. Select 'Login' to go to login screen
  2. Select 'Parent' when prompted to select between Parent and Clinician roles
  3. User is brought to the Parent login screen

- FR-ST-A2

Control: Manual

Initial State: User has a Clinician account already created and stored in the database

Input: Selection of Clinician account role for login

Output: The expected result is the Clinician account role is selected and User is brought to the Clinician login screen

Test Case Derivation: The expected output is justified based on FR-A1 in the SRS document

How the test will be performed:

1. Select 'Login' to go to login screen
2. Select 'Clinician' when prompted to select between Parent and Clinician roles
3. User is brought to the Clinician login screen

- FR-ST-A3

Control: Manual

Initial State: User does not have a Parent account stored in the database

Input: Selection of 'Create Account', with a username that does not exist in the database, upon attempting to access the system

Output: The expected result is a new Parent account is created

Test Case Derivation: The expected output is justified based on FR-A2 in the SRS document

How the test will be performed:

1. Select 'Create Account' to go to create account screen
2. Enter unique username that is not in the database
3. Enter account credentials (to complete account create process)
4. Parent account is created

- FR-ST-A4

Control: Manual

Initial State: User does not have a Parent account stored in the database

Input: Selection of 'Create Account', with a username that exists in the database, upon attempting to access the system

Output: The expected result is a new Parent account fails to be created

Test Case Derivation: The expected output is justified based on FR-A2 in the SRS document

How the test will be performed:

1. Select 'Create Account' to go to create account screen
2. Enter username that already exists in the database
3. System communicates the account could not be created
4. System prompts user to select a new username
5. Parent account is not created

- FR-ST-A5

Control: Manual

Initial State: User has Admin privileges, attempting to create a new Clinician account

Input: Admin user selects option to 'Create Account', with a username that does not exist in the database, upon attempting to access the system

Output: The expected result is a new Clinician account is created

Test Case Derivation: The expected output is justified based on FR-A3 in the SRS document

How the test will be performed:

1. Admin user is logged into their account
2. Select 'Create Account' to go to create account screen
3. Enter unique username that is not in the database
4. Enter account credentials (to complete account create process)
5. Clinician account is created

- FR-ST-A6

Control: Manual

Initial State: User has Admin privileges, attempting to create a new Clinician account

Input: Admin user selects option to 'Create Account', with a username that exists in the database, upon attempting to access the system

Output: The expected result is a new Clinician account fails to be created

Test Case Derivation: The expected output is justified based on FR-A3 in the SRS document

How the test will be performed:

1. Admin user is logged into their account
2. Select 'Create Account' to go to create account screen
3. Enter username that already exists in the database
4. System communicates the account could not be created

5. System prompts admin user to select a new username
6. Clinician account is not created

- FR-ST-A7

Control: Manual

Initial State: User is on their corresponding role's login page, with an account already created and stored in the database

Input: Unique username and corresponding password that exists in the database

Output: The expected result is a successful login to a user's account

Test Case Derivation: The expected output is justified based on FR-A4 in the SRS document

How the test will be performed:

1. On login screen
2. Enter unique username
3. Enter corresponding password
4. Select login to enter account
5. Logged into account

- FR-ST-A8

Control: Manual

Initial State: User is logged into their account

Input: Selection of 'logout'

Output: The expected result is a successful logout from a user's account

Test Case Derivation: The expected output is justified based on FR-A5 in the SRS document

How the test will be performed:

1. Logged into account
2. Select 'logout'
3. System logs user out of their account
4. Logout confirmation is displayed to the user

4.1.3 Data Processing and Display

This set of test cases will help confirm the system's data retrieval, report generation, and display functionalities to ensure the clinician experience aligns with the project's goals.

- FR-ST-DPD1

Control: Automatic

Initial State: Database is populated with processed assessment data.

Input: Query request for a specific patient's processed assessment data.

Output: The expected result is the successful retrieval of all relevant assessment data, displayed without errors within MAX_PROCESSING_TIME

Test Case Derivation: Ensures the system can retrieve data efficiently for report generation, meeting retrieval speed and completeness requirements.

How the test will be performed:

1. Query the database for a test patient's processed assessment results.
2. Measure and record retrieval time, ensuring it does not exceed MAX_PROCESSING_TIME.
3. Verify that all required data is retrieved and matches the stored information.

- FR-ST-DPD2

Control: Automatic

Initial State: Database has assessment data including flagged occurrences, timestamps, and patient performance metrics.

Input: Trigger for report generation based on a retrieved assessment dataset.

Output: The expected result is a generated report containing all required data within MAX_PROCESSING_TIME.

Test Case Derivation: Confirms that report generation is complete, accurate, and within performance constraints.

How the test will be performed:

1. Retrieve a patient's assessment data from the database.
2. Trigger report generation.
3. Confirm the report includes flagged occurrences, timestamps, and performance metrics.
4. Measure and confirm report generation time does not exceed MAX_PROCESSING_TIME.

- FR-ST-DPD3

Control: Manual

Initial State: Generated report available in the database.

Input: Clinician dashboard query to display the generated report.

Output: Report displayed in the clinician's dashboard with accurate formatting, charts, and tables, fully loaded within MAX_PROCESSING_TIME.

Test Case Derivation: Validates the report display function, ensuring usability and speed requirements are met.

How the test will be performed:

1. Query the clinician's dashboard to load the report.
2. Confirm that the report is displayed with correct charts, tables, and formatting.
3. Verify full loading of the report within MAX_PROCESSING_TIME.

- FR-ST-DPD4

Control: Manual

Initial State: Database has stored reports for previous sessions, each with a unique patient identifier.

Input: Clinician request to access a specific previously generated report.

Output: The expected result is successful retrieval and display of the requested report without errors, within MAX_PROCESSING_TIME.

Test Case Derivation: Ensures that clinicians can reliably access and view past reports, supporting longitudinal patient assessment.

How the test will be performed:

1. Query the database for a stored report using a unique patient identifier.
2. Verify that the retrieved report is complete and accurate.
3. Confirm that the report is displayed within MAX_PROCESSING_TIME.

4.1.4 Data Collection and Storage

The test cases below focus on ensuring reliable storage and retrieval of multimedia data, privacy compliance by excluding PII, accurate grouping under unique identifiers, and long-term report accessibility, meeting all data storage and organization requirements.

- FR-ST-DSC1

Control: Automatic

Initial State: Database is empty or initialized with test data.

Input: Multimedia files (video, audio, and structured data files).

Output: The expected result is a success message in console for both storing data and retriving data

Test Case Derivation: Ensures the database meets storage capacity and integrity requirements as per FR-DSC1 in SRS document.

How the test will be performed:

1. Insert a session containing multimedia files (video, audio, JSON) into the database.
2. Retrieve the session files and check for data integrity by comparing size of stored data with retrieved data.
3. Verify that the retrieved files are uncorrupted and correctly match the original files.

- FR-ST-DSC2

Control: Manual

Initial State: Database is set up to store assessment session data.

Input: Video, audio files, flagged occurrences, and timestamps for each assessment question.

Output: The expected result is the creation of a JSON file that contains flagged occurrences and timestamps which is stored alongside the data.

Test Case Derivation: Allows the system to store all important information for clinicians to use.

How the test will be performed:

1. Insert a test assessment session with video and audio files, flagged occurrences, and timestamps.
2. Query the database to retrieve the session's data and verify the presence of a JSON file with accuracy of flagged occurrences and timestamps.

- FR-ST-DSC3

Control: Automatic

Initial State: Database configured to prevent storage of personally identifiable information (PII).

Input: Attempted insertion of a record containing personally identifiable information.

Output: The expected result is that the database rejects any PII-containing

records and stores only anonymized data.

Test Case Derivation: Ensures compliance with privacy standards, verifying that no PII is stored in the database.

How the test will be performed:

1. Attempt to insert a record with PII (e.g., name, address).
2. Verify that the system blocks or anonymizes PII, preventing its storage.
3. Retrieve all clinician-accessible data and confirm the absence of PII.

- FR-ST-DSC4

Control: Manual

Initial State: Database initialized and ready for storing user session data.

Input: Multiple sessions, each with unique user identifiers.

Output: The expected result is that all session data is stored and grouped correctly according to the unique user identifiers.

Test Case Derivation: Confirms database's grouping and retrievable capabilites, ensuring accurate data organization.

How the test will be performed:

1. Insert multiple sessions into the database, each tagged with a unique user identifier.
2. Query the database for each user identifier and verify that all associated session data is correctly grouped.
3. Confirm no data is incorrectly associated or left unassociated.

- FR-ST-DSC5

Control: Manual

Initial State: Database initialized and ready to store reports with unique identifiers.

Input: Assessment report linked to a patient's unique identifier.

Output: The expected result is that the report is successfully stored, linked to the corresponding patient identifier, and retrievable for at least MAX_STORAGE_TIME.

Test Case Derivation: Verifies long-term storage and retrieval of assessment reports, supporting patient progress tracking.

How the test will be performed:

1. Insert a test report with a unique patient identifier into the database.
2. Retrieve the report using the identifier and confirm the report's accuracy and timestamp.

3. Check when database will be wiped out and when backups occure

4.1.5 Video and Audio Data Analysis

The test cases below focus on ensuring the video analysis model can reliably access session recordings, accurately detect and log speech disturbances, and correctly flag disturbances with timestamps, questions, and user responses to support efficient clinical review.

- FR-ST-VDA1

Control: Automatic

Initial State: Completed assessment sessions are available in the database, with video and audio recordings accessible for processing.

Input: Request by the analysis model to access video and audio data from a completed session.

Output: The expected result is that all videos requested are processed with a success message in the logs

Test Case Derivation: Verifies that the model has reliable access to stored multimedia data, which is critical for processing and analysis.

How the test will be performed:

1. Retrivee the video and audio recordings from several completed sessions.
2. Check that the model successfully accesses the multimedia files for each session without data access errors.
3. Verify that the files are correctly loaded for analysis with no corruption or access issues.

- FR-ST-VDA2

Control: Automatic

Initial State: The analysis model is initialized and ready to process test video and audio data.

Input: Video and audio data containing speech disturbances, interruptions, and other irregularities for analysis.

Output: The expected results is an accuracy of VERY_HIGH_SUCCESS_RATE in a JSON file for number of disturbances found by the model

Test Case Derivation: Confirms that the model's disturbance detection meets the accuracy requirement, reducing bias in the analysis process.

How the test will be performed:

1. Run the model on a test dataset containing known speech disturbances.
2. Compare the disturbances identified by the model with human observations

for accuracy validation.
3. Verify that the model achieves at least VERY_HIGH_SUCCESS_RATE accuracy in identifying and logging disturbances.

- FR-ST-VDA3

Control: Automatic

Initial State: Video and audio data with disturbances has been processed by the analysis model.

Input: Disturbances identified by the model, requiring flags with associated timestamps, assessment questions, and user answers.

Output: The expected result is an accuracy of VERY_HIGH_SUCCESS_RATE in a JSON file for timestamp accuracy

Test Case Derivation: Ensures clinicians can quickly access relevant parts of the assessment with accuracy, aiding in efficient diagnosis.

How the test will be performed:

1. Process a test assessment session with the model, identifying and flagging disturbances.
2. Retrieve flagged disturbances and confirm each has an accurate timestamp, associated question, and user response.
3. Compare the flagged data with human observations and verify at least VERY_HIGH_SUCCESS_RATE accuracy in the model's associations.

...

4.2 Tests for Nonfunctional Requirements

[The nonfunctional requirements for accuracy will likely just reference the appropriate functional tests from above. The test cases should mention reporting the relative error for these tests. Not all projects will necessarily have nonfunctional requirements related to accuracy. —SS]

[For some nonfunctional tests, you won't be setting a target threshold for passing the test, but rather describing the experiment you will do to measure the quality for different inputs. For instance, you could measure speed versus the problem size. The output of the test isn't pass/fail, but rather a summary table or graph. —SS]

[Tests related to usability could include conducting a usability test and survey. The survey will be in the Appendix. —SS]

[Static tests, review, inspections, and walkthroughs, will not follow the format for the tests given below. —SS]

[If you introduce static tests in your plan, you need to provide details. How will they be done? In cases like code (or document) walkthroughs, who will be involved? Be specific. —SS]

4.2.1 Look and Feel Requirements

These test cases ensure that all appearance and style requirements are addressed effectively, covering navigation, user-friendliness, brand consistency, visual appeal, and responsiveness.

- LF-ST-LFR1 (covers LF-AR1, LF-AR2, LF-AR4)

  Type: Dynamic

  Initial State: Platform initialized and navigable on a test device, prepared for user testing with adults and children.

  Input/Condition: Conduct user tests with participants performing core tasks like starting an assessment, navigating menus, and viewing results.

  Output/Results: Expected results include:

  - No more than three levels of navigation depth, and each screen presents no more than six main options.
  - MAX_SUCESS_RATE of users complete core tasks within five clicks.
  - VERY_HIGH_SUCCESS_RATE of children aged 6-12 can complete a sample assessment independently.

  How the test will be performed:

  1. Observe and record the number of clicks taken by each user to complete key tasks.
  2. Inspect the navigation structure to ensure it meets depth and option constraints.
  3. Test with children to verify completion of assessments independently.

- LF-ST-LFR2 (covers LF-AR3, LF-AR5, LF-SR1, LF-SR2)

  Type: Static and Dynamic

  Initial State: Platform with finalized color schemes, fonts, and brand assets loaded and available for inspection and user interaction tests.

  Input/Condition: Perform visual inspection and feedback collection, along with response-time measurements for interactive elements.

  Output/Results: Expected results include:

  - Compliance with brand guidelines, style guide, and use of no more than three calming, neutral/pastel tones.
  - Positive feedback from child participants indicating a calm, non-stressful

environment.

- MAX_SUCESS_RATE consistency in design across all pages.
- MAX_SUCESS_RATE of user interactions provide immediate feedback within SHORT_PROCESSING_TIME.

How the test will be performed:

1. Compare platform's design elements against the client's brand and style guidelines.
2. Collect feedback from child participants about the visual atmosphere.
3. Test and measure feedback response times for various interactions.

4.2.2 Usability and Humanity

The test cases below ensures that the system meets usability and humanity requirements for users to have an enjoyable and accessible experience.

- UH-ST-EOU1 (covers UH-EOU1, UH-EOU2, UH-LI1, UH-UP1, UH-AR1)

Type: Usability, Manual

Initial State: System is complete, functional, and ready for user interaction.

Input/Condition: Users complete one full assessment using the system.

Output/Results: User answers questions in the Usability Survey (6.2), and results are culminated

How the test will be performed:

1. User have access to the system
2. User completes one full assessment using the system
3. Upon completion of the assessment, user is requested to fill out a usability survey
4. Results are stored
5. Usability scores are averaged across users

- UH-ST-PI1 (covers UH-PI1)

Type: Static

Initial State: System, including assessments, have been completed.

Input/Condition: List of available languages to perform assessments in is available to be selected and listed

Output/Results: Count the number of available languages for the assessment

How the test will be performed:

1. View list of available languages
2. Count number of languages are available for the assessment

- UH-ST-LI2 (covers UH-LI2)

Type: Manual

Initial State: User documentation has been completed and made available to users.

Input/Condition: Link to documentation is available on the system's frontend interface, and can be accessed

Output/Results: Verify link takes user to access documentation

How the test will be performed:

1. Select 'documentation'
2. User goes to documentation screen
3. User has access to view up-to-date, available documentation

4.2.3 Operational and Environmental

The test cases below ensures that the system can be used in a variety of environments, along with the requirements for which users are expected to use the system within, and the capabilities and qualities the system has to interact with adjacent systems in the environment.

- OE-ST-EPE1 (covers OE-EPE1)

Type: Usability, Manual

Initial State: System is complete, functional, and ready for user interaction.

Input/Condition: Testing the system, including the assessment, on a variety of screen sizes

Output/Results: The system's displayed elements will scale appropriately to different screen sizes

How the test will be performed:

1. User logs into the system
2. User completes one full assessment using the system
3. Upon completion of the assessment, user is requested to fill out a usability survey
4. User answers questions about their screensize and if the test scaled accordingly
5. Results are stored for review

- OE-ST-WE1 (covers OE-WE1, OE-WE2)

Type: Dynamic, Manual

Initial State: System, including assessments, have been completed.

Input/Condition: User attempts to start system setup

Output/Results: Device verification displayed on-screen, informing the user that the environment they're in is suitable for the assessment

How the test will be performed:

1. Select 'system setup'
2. System checks if connected to the internet
3. System checks audio input is not noisy
4. System checks video input is clear
5. System displays to the user their device is ready for the assessment to be used in the current environment

- OE-ST-IA1 (covers OE-IA1)

Type: Functional, Dynamic

Initial State: System is connected to external server for retrieving and storing data

Input/Condition: Assessment is complete, and results need to be stored

Output/Results: Verify results are stored in the external server

How the test will be performed:

1. Complete assessment
2. Access external server
3. Check if results have been uploaded to server
4. Access results to ensure data has been uploaded successfully

4.2.4 Maintainability and Support Requirements

These test cases ensure the platform meets its maintenance, support, and adaptability requirements effectively.

- MS-ST-MSA1 (covers MS-MR1, MS-SR1)

Type: Static and Dynamic

Initial State: Modular platform architecture with access to the component's source code. A direct link to GitHub is also available on the platform.

Input/Condition: Perform updates to individual components and simulate user feedback submissions via the GitHub repository.

Output/Results: Expected results include:

- Each component update does not exceed 10 lines of code edited outside the updated module.
- Users can submit issues and feature requests directly to GitHub, categorized as issues, feature requests, or feedback.

How the test will be performed:

1. Perform code updates on isolated components and verify changes are contained within 10 lines outside the component.
2. Test submission flow to GitHub, verifying links are accessible and that issues and requests are categorized correctly.

- MS-ST-MSA2 (covers MS-SR2)

Type: Dynamic

Initial State: Platform initialized with a tutorial accessible from the homepage.

Input/Condition: New user group follows the tutorial to complete primary tasks (e.g., starting an assessment).

Output/Results: Expected results include:

- HIGH_SUCCESS_RATE of users can complete core tasks correctly after following the tutorial.

How the test will be performed:

1. Guide users through the tutorial and observe task completion rates.
2. Collect feedback on tutorial clarity and assess if 90% of users can independently complete tasks.

- MS-ST-MSA3 (covers MS-AR1)

Type: Dynamic

Initial State: Platform accessible across various devices and screen sizes, from mobile (MIN_SCREEN_SIZE) to desktop (MAX_SCREEN_SIZE).

Input/Condition: Load and navigate the platform across multiple devices to evaluate responsive design and functionality.

Output/Results: Expected results include:

- MAX_SUCESS_RATE of essential features are fully functional and readable across all screen sizes tested.

How the test will be performed:

1. Access the platform on various screen sizes and test for display, layout, and functionality.
2. Verify that all features are usable and adapt responsively without readability or functionality loss.

4.2.5 Cultural Requirements

These tests ensure that the platform respects cultural sensitivities and provides full bilingual support, enhancing inclusivity and accessibility for diverse user groups.

- CU-ST-CUR1 (covers CU-CR1)

Type: Static and Dynamic

Initial State: Platform content (language and imagery) is finalized and presented for review.

Input/Condition: A cultural consultant reviews all language and imagery, and user acceptance testing gathers feedback from a diverse set of users.

Output/Results: Expected results include:

- MAX_SUCESS_RATE of reviewed content is confirmed as culturally sensitive with no instances of offensive language or imagery.

How the test will be performed:

1. A cultural consultant examines all text and imagery for potential cultural insensitivity.
2. Conduct user acceptance testing with a diverse group and gather feedback on the platform's cultural sensitivity.
3. Validate that all feedback confirms no culturally insensitive content.

- CU-ST-CUR2 (covers CU-CR2)

Type: Static

Initial State: Platform is available in both English and Mandarin, with all interface elements and assessments translated.

Input/Condition: Switch between language settings to review each text, instruction, and assessment in both languages.

Output/Results: Expected results include:

- MAX_SUCESS_RATE of content is fully translated and functional in both English and Mandarin with no untranslated elements.

How the test will be performed:

1. Navigate through the platform in both English and Mandarin settings, verifying translations for each section.
2. Confirm that all assessments, instructions, and interface elements are accurately translated and free from language discrepancies.

4.2.6 Performance

The test cases below ensures that the system meets essential performance metrics, including quick page load times, low latency in video and audio recording, high video resolution, and efficient report generation.

- PR-ST-SL1

Type: Dynamic

Initial State: System initialized, strong internet connection

Input/Condition: User navigates to different web pages within the system.

Output/Results: The expect output is that each web page loads fully with all functionalities within MAX_LOAD_TIME.

How the test will be performed:

1. Navigate to various web pages within the system.
2. Measure the time taken for each page to load completely.
3. Confirm that all pages load within MAX_LOAD_TIME.

- PR-ST-SL2

Type: Static

Initial State: Video and audio recording session initialized.

Input/Condition: Audio and video session of user performing gestures while talking

Output/Results: The expected output is the latency between actions and recorded playback remains under SHORT_PROCESSING_TIME, ensuring synchronization.

How the test will be performed:

1. Begin recording a session.
2. Have the user perform timed actions while recording.
3. Play back the recording and measure the latency between actions and their corresponding timestamps.
4. Confirm latency does not exceed SHORT_PROCESSING_TIME.

- PR-ST-SL3

Type: Dynamic

Initial State: System configured to store video data.

Input/Condition: Video recorded and stored from assessment an session.

Output/Results: The exepect output is the video quality is at least AVERAGE_RESOLUTION upon when retriving or storing.

How the test will be performed:

1. Record a session and store the video.
2. Retrieve the stored video and verify its resolution.
3. Confirm the resolution is AVERAGE_RESOLUTION or higher.

The test cases below focus on verifying the system's precision in detecting and analyzing speech and gesture disturbances, ensuring timestamp alignment, and maintaining 100% accuracy in assessment data.

- PR-ST-PA1

Type: Manual, Dynamic

Initial State: Video and Audio analysis model is loaded with sample audio data.

Input/Condition: Data containing speech patterns and video with known disturbances.

Output/Results: The expected output is that the analysis achieves VERY_HIGH_SUCCESS_RATE in detecting speech and gesture disturbances.

How the test will be performed:

1. Load test data files with known disturbances into the analysis model.
2. Compare detected disturbances with human-reviewed observations.
3. Verify that the model correctly identifies at least VERY_HIGH_SUCCESS_RATE of disturbances.

- PR-ST-PA3

Type: Static

Initial State: Timestamp function is synchronized with real-time actions.

Input/Condition: User performs actions in the recorded session.

Output/Results: The expected output is that the timestamps delay within SHORT_PROCESSING_TIME of the real-time action.

How the test will be performed:

1. Record a session with specific user actions.
2. Analyze timestamps and compare them to the actual timing of the actions.
3. Confirm each timestamp falls within SHORT_PROCESSING_TIME margin of the real action.

- PR-ST-PA4

Type: Manual, Static

Initial State: Assessment answer key is loaded.

Input/Condition: Manual verification of the answer key's accuracy.

Output/Results: The expected output is that the answer key is MAX_SUCESS_RATE.

How the test will be performed:

1. Manually review each entry in the assessment answer key.
2. Check for errors or inconsistencies in each answer.
3. Confirm all answers are correct, ensuring MAX_SUCESS_RATE.

The test cases below validate the system's ability to handle errors, back up data reliably, and enforce strict input validation.

- PR-ST-RFT1

Type: Dynamic

Initial State: System is operational with error handling in place.

Input/Condition: User initiates actions known to cause common errors.

Output/Results: The expected output is the system displays clear error messages for at least VERY_HIGH_SUCCESS_RATE of the common errors encountered.

How the test will be performed:

1. Simulate common user errors, such as invalid inputs or incorrect file uploads.
2. Observe system response and displayed error messages.
3. Verify clarity and accuracy of error messages in at least VERY_HIGH_SUCCESS_RATE.

- PR-ST-RFT2

Type: Dynamic

Initial State: Database backup processes configured and operational in the system.

Input/Condition: Monthly data backup event.

Output/Results: The expected output is that the system performs a data backup within a MONTHLY_BACKUP timeframe on the first of each month.

How the test will be performed:

1. A data backup is triggered.
2. Record the duration of the backup process.
3. Confirm that backup completes within MONTHLY_BACKUP.

The test cases below ensure that the system can handle the expected user load, data storage needs, and simultaneous uploads without performance degradation.

- PR-ST-CR1

Type: Dynamic

Initial State: System initialized with maximum user capacity parameters.

Input/Condition: System loaded with MIN_USERS accounts.

Output/Results: The expected result is that the system operates stably and manages all accounts without issues.

How the test will be performed:

1. Create and load MIN_USERS accounts into the system.
2. Monitor system performance metrics, including stability and response time.
3. Confirm system maintains stable performance.

- PR-ST-CR2

Type: Static

Initial State: System is initialized with required storage capacity.

Input/Condition: Data stored in the database approaches MIN_STORAGE annually.

Output/Results: The expected result is that the system accommodates MIN_STORAGE of data without loss of performance.

How the test will be performed:

1. Load MIN_STORAGE data into database.
2. Monitor database performance metrics, such as access time and error rate.
3. Confirm system's ability to manage MIN_STORAGE without performance impact.

The test cases below confirm that the system can scale effectively to accommodate an increasing user base, data volume, and computational needs over time.

- PR-ST-SE1

Type: Static

Initial State: System operational with current allocated user amount.

Input/Condition: Increase user base by YEARLY_INCREASE_PERCENTAGE.

Output/Results: The expected result is that the system maintains performance while handling user growth.

How the test will be performed:

1. Simulate a YEARLY_INCREASE_PERCENTAGE increase in the user base by increase user base parameters by YEARLY_INCREASE_PERCENTAGE.
2. Monitor system metrics such as response time and error rate.
3. Confirm that the system operates within acceptable performance metrics post-increase.

The test cases below verify the system's reliability across updates and compatibility with major operating systems.

- PR-ST-LR1

Type: Static

Initial State: System release build in use, with ongoing development updates.

Input/Condition: System stability monitored over successive updates.

Output/Results: The expected result is that the system maintains a failure rate below LOW_FAILURE_RATE in release builds.

How the test will be performed:

1. Conduct routine tests on the release build during ongoing development updates.
2. Monitor system logs for errors or malfunctions.
3. Verify that the failure rate remains below LOW_FAILURE_RATE.

- PR-ST-LR2

Type: Static

Initial State: System configured for compatibility testing across platforms.

Input/Condition: System loaded on Windows, macOS, Linux, Android, and iOS.

Output/Results: The expected result is that the system functions correctly across all platforms.

How the test will be performed:

1. Run the system on each specified operating system.
2. Perform standard operations and monitor user experience on each platform.
3. Confirm that the system operates without issues on all platforms.

4.2.7 Area of Testing2

...

4.3 Traceability Between Test Cases and Requirements

[Provide a table that shows which test cases are supporting which requirements. —SS]

Table 3: Traceability Table Between System Test Cases and Functional Requirements

	FR-ST-A1	FR-ST-A2	FR-ST-A3	FR-ST-A4	FR-ST-A5	FR-ST-A6	FR-ST-A7	FR-ST-A8	FR-ST-DSC1	FR-ST-DSC2	FR-ST-DSC3	FR-ST-DSC4	FR-ST-DSC5	FR-ST-VDA1	FR-ST-VDA2	FR-ST-VDA3	FR-ST-DPD1	FR-ST-DPD2	FR-ST-DPD3	FR-ST-DPD4	FR-ST-SS1	FR-ST-SS2	FR-ST-SS3	FR-ST-SS4	FR-ST-SS5	FR-ST-AI1	FR-ST-AI2	FR-ST-AI3	FR-ST-AI4	FR-ST-AI5	FR-ST-AI6	FR-ST-AI7
FR-A1	X	X																													
FR-A2			X	X																											
FR-A3					X	X																									
FR-A4							X																								
FR-A5								X																							
FR-SS1																					X										
FR-SS2																						X									
FR-SS3																							X								
FR-SS4																								X							
FR-SS5																									X						
FR-AI1																										X					
FR-AI2																											X				
FR-AI3																												X			
FR-AI4																													X		
FR-AI5																														X	
FR-AI6																															X
FR-AI7																															X
FR-DSC1									X																						
FR-DSC2										X																					
FR-DSC3											X																				
FR-DSC4												X																			
FR-DSC5													X																		
FR-VADA1														X																	
FR-VADA2															X																
FR-VADA3																X															
FR-DPD1																	X														
FR-DPD2																		X													
FR-DPD3																			X												
FR-DPD4																				X											

Table 4: Traceability Table Between System Test Cases and Nonfunctional Requirements

System test	LF-ST-LFR1	LF-ST-LFR2	UH-ST-EOU1	UH-ST-PI1	UH-ST-LI2	PR-ST-SL1	PR-ST-SL2	PR-ST-SL3	PR-ST-PA1	PR-ST-PA3	PR-ST-PA4	PR-ST-RFT1	PR-ST-RFT2	PR-ST-CR1	PR-ST-CR2	PR-ST-SE1	PR-ST-LR1	PR-ST-LR2	OE-ST-EPE1	EO-ST-WE1	OE-ST-IA1	MS-ST-MSA1	MS-ST-MSA2	MS-ST-MSA3	CU-ST-CUR1	CU-ST-CUR1	SR-ST-AC1	SR-ST-AC2	SR-ST-AC3	SR-ST-AC4	SR-ST-IM1	CR-ST-D1	SR-ST-P1	SR-ST-P2	SR-ST-P3
LF-AR1	X																																	
LF-AR2	X																																	
LF-AR3		X																																
LF-AR4	X																																	
LF-AR5		X																																
LF-SR1		X																																
LF-SR2		X																																
UH-EOU1			X																															
UH-EOU2			X																															
UH-PI1				X																														
UH-LI1			X																															
UH-LI2					X																													
UH-UP1			X																															
UH-AR1			X																															
PR-SL1						X																												
PR-SL2							X																											
PR-SL3								X																										
PR-SL4																																		
PR-SCL1																																		
PR-PA1									X																									
PR-PA2									X																									
PR-PA3										X																								
PR-PA4											X																							
PR-PA5											X																							
PR-RFT1												X																						
PR-RFT2													X																					
PR-RFT3												X																						
PR-CR1														X																				
PR-CR2															X																			
PR-CR3														X																				
PR-CR4														X																				
PR-SE1																X																		
PR-SE2															X																			
PR-SE3															X																			
PR-LR1																	X																	
PR-LR2																		X																
OE-EPE1																			X															
OE-WE1																				X														
OE-WE2																				X														
OE-IA1																					X													
MS-MR1																						X												
MS-SR1																						X												
MS-SR2																							X											
MS-AR1																								X										
SR-AC1																										X								
SR-AC2																											X							
SR-AC3																												X						
SR-AC4																													X					
SR-P1																															X			
SR-P2																																X		
SR-P3																																		X
SR-IM1																														X				
CU-CR1																								X										
CU-CR2																									X									
CR-STD1																														X				

5 Unit Test Description

Section 5, Unit Test Description, will be filled in prior to the revision 0 deliverable. The team decided to leave in the template instructions as a guide to indicate our desried return to the document.
[This section should not be filled in until after the MIS (detailed design document) has been completed. —SS]

[Reference your MIS (detailed design document) and explain your overall philosophy for test case selection. —SS]

[To save space and time, it may be an option to provide less detail in this section. For the unit tests you can potentially layout your testing strategy here. That is, you can explain how tests will be selected for each module. For instance, your test building approach could be test cases for each access program, including one test for normal behaviour and as many tests as needed for edge cases. Rather than create the details of the input and output here, you could point to the unit testing code. For this to work, you code needs to be well-documented, with meaningful names for all of the tests. —SS]

5.1 Unit Testing Scope

[What modules are outside of the scope. If there are modules that are developed by someone else, then you would say here if you aren't planning on verifying them. There may also be modules that are part of your software, but have a lower priority for verification than others. If this is the case, explain your rationale for the ranking of module importance. —SS]

5.2 Tests for Functional Requirements

[Most of the verification will be through automated unit testing. If appropriate specific modules can be verified by a non-testing based technique. That can also be documented in this section. —SS]

5.2.1 Module 1

[Include a blurb here to explain why the subsections below cover the module. References to the MIS would be good. You will want tests from a black box perspective and from a white box perspective. Explain to the reader how the tests were selected. —SS]

1. test-id1

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input:

   Output: [The expected result for the given inputs —SS]

Test Case Derivation: [Justify the expected value given in the Output field —SS]

How test will be performed:

2. test-id2

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input:

   Output: [The expected result for the given inputs —SS]

   Test Case Derivation: [Justify the expected value given in the Output field —SS]

   How test will be performed:

3. ...

5.2.2 Module 2

...

5.3 Tests for Nonfunctional Requirements

[If there is a module that needs to be independently assessed for performance, those test cases can go here. In some projects, planning for nonfunctional tests of units will not be that relevant. —SS]

[These tests may involve collecting performance data from previously mentioned functional tests. —SS]

5.3.1 Module ?

1. test-id1

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input/Condition:

   Output/Result:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

5.3.2 Module ?

...

5.4 Traceability Between Test Cases and Modules

[Provide evidence that all of the modules have been considered. —SS]

References

Author Author. System requirements specification. https://github.com/..., 2019.

6 Appendix

Below is additional information relevant to the document.

6.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

Variable Name	Value
MAX_PROCESSING_TIME	10 seconds
SHORT_PROCESSING_TIME	0.5 seconds
MAX_SUCESS_RATE	100%
VERY_HIGH_SUCCESS_RATE	95%
HIGH_SUCCESS_RATE	90%
NUM_CODE_LINES	10
MIN_SCREEN_SIZE	4 inches
MAX_SCREEN_SIZE	27 inches
MAX_STORAGE_TIME	7 years
MAX_LOAD_TIME	3 seconds
AVERAGE_RESOLUTION	720p
MONTHLY_BACKUP	4 hour
MIN_USERS	2000 users
MIN_STORAGE	10TB
YEARLY_INCREASE_PERCENTAGE	10%
LOW_FAILURE_RATE	1%

Table 5: Variable Names and Values

6.2 Usability Survey Questions?

The following questions depict the first draft of the team's usability.
The usability survey will be conducted after participants have engaged with testing and using the system for at least one iteration of the assessment.

Please select the statement that best describes your experience for each of the following:

1. Learning how to use the system was easy:

   Strongly Disagree Disagree Neutral Agree Strongly Agree

2. Setting up the system was easy:

Strongly Disagree Disagree Neutral Agree Strongly Agree

3. I found the assessment interface easy to use:

   Strongly Disagree Disagree Neutral Agree Strongly Agree

4. Navigating the interface was easy:

   Strongly Disagree Disagree Neutral Agree Strongly Agree

5. All the button interactions reacted and responded how I thought they should:

   Strongly Disagree Disagree Neutral Agree Strongly Agree

6. The information on screen was easy to read and understand:

   Strongly Disagree Disagree Neutral Agree Strongly Agree

7. I like the organization of the assessment interface:

   Strongly Disagree Disagree Neutral Agree Strongly Agree

8. I enjoyed my overall experience using the TeleHealth Insights platform:

   Strongly Disagree Disagree Neutral Agree Strongly Agree

Answer the following:

9. What was the most difficult part of using the platform?

Insert answer here...

10. Did you encounter any bugs/problems while using the platform? If so, what were they?

Insert answer here...

11. What was your favourite part of the experience? Why?

Insert answer here...

12. What was your least favourite part of the experience? Why?

Insert answer here...

13. Were there any aspects of the platform that you found unnecessary? Why?

Insert answer here...

14. Which part of the platform needs the most improvement? Why?

Insert answer here...

15. What would you like changed to make the platform easier to use?

    Insert answer here...

16. What is the device you ran the system on?

    Insert answer here...

17. Did the screen's visuals scale appropriately to the screen size?

    Insert answer here...

18. Do you have any additional feedback?

    Insert answer here...

Appendix — Reflection

[This section is not required for CAS 741 —SS]

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning.

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?

   Mitchell: One of the things that went well during this deliverable was splitting up the System Tests for both Functional Requirements and Nonfunctional Requirements. The team decided to assign the functional and nonfunctional tests to the team members that wrote those particular requirements in the SRS. This allowed the team to use their prior knowledge on the requirements to develop detailed test plans to accurately test the requirements. This also meant that team members that were familiar with the requirements knew the limitations of the tests, and could improve upon given feedback to strengthen the plan further.

   Jasmine: I think the team worked well together as usual, especially when it came to distributing tasks. Our team was very efficient and decided to split test cases for the functional and nonfunctional requirements in the same way we split up writing the requirements for the SRS, and the rest of the document was split up fairly quickly within a team meeeting. Another thing that went well while writing this deliverable is organzation. Now that we have written several pieces of documentation, the team is comfortable navigating LaTex formatting, as well as GitHub project and issue tracking.

   Promish: One thing that went well during this deliverable was how comfortable everyone became using GitHub. We created issues for each task, and handling merge conflicts was faster. As a team, we quickly divided the work, understanding that writing system tests for the requirements each of us specialized in was the most efficient approach. We also recognized the importance of consistent naming for our tests to maintain traceability with our SRS, so we decided to include "ST" in the middle of our SRS requirements tags.

2. What pain points did you experience during this deliverable, and how did you resolve them?

   Mitchell: One of the pain points I experienced during this deliverable was understanding the format needed for the System Tests section. I found it difficult to get started because of the formality of the tests. As well, I wanted to make sure that the tests were consistent among team members, and a difference in formatting would be difficult for someone reading the document to understand. To resolve this, the team decided to follow a consistent formatting outline, and follow along a sample so that everyone knew what test cases should look like for this deliverable.

   Jasmine: One pain point during this deliverable was clarification of document instructions, such as what area testing is or what exactly the symbolic constraints were. We resolved this by preparing questions to ask the TA during our informal TA meeting, and looking at examples from other capstone projects from the same course completed in previous years.

   Promish: A challenge we faced during this deliverable was the large number of tests we had to write within a limited timeframe. With only a week to work on the document, we didn't have time for extra meetings with our advisor and TA to clarify certain sections as much as I would have liked. Overall, the team did a great job with the time and resources available, but some parts of the document were a bit unclear, especially regarding formatting and focus. The rubric was also quite general, making it harder to understand the exact requirements.

3. What knowledge and skills will the team collectively need to acquire to successfully complete the verification and validation of your project? Examples of possible knowledge and skills include dynamic testing knowledge, static testing knowledge, specific tool usage, Valgrind etc. You should look to identify at least one item for each team member.

   Mitchell: One of the pieces of knowledge and skills the team collectively needs to acquire to successfully complete the verification and validation of the project is understanding how to properly perform usability testing. This will be important to the team because it is one of the project's chosen extras, so understanding usability testing is crucial for the success of the project. As well, it will inform us on how users actually engage, interact, and understand our system. Another skill the team will need to acquire is dynamic testing using different frameworks. It will be beneficial to perform dynamic testing to get verify each of our desired outcomes of tests against the actual outcomes. This will provide the team with accurate, reliable results and conclusions.

Jasmine: The team will need skills in both dynamic and static testing to successfully complete the verification and validation of the project, including knowledge of effective test case creation and debugging. Being familiar with tools like automated test frameworks and static analysis tools that our team can use for our project will be important to improving efficiency and accuracy in our testing processes. Additionally, understanding quality assurance practices and common code review techniques will help make sure our validation is thorough and reliable.

Promish: As a team, we'll need to focus on learning to use the frameworks outlined in section 3.6 to automate our unit tests. We'll also need to understand how to evaluate our SRS and the processes for making changes. Lastly, we'll need to know where testing fits into continuous development and how to follow Agile practices effectively.

4. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?

   Mitchell: One approach to acquire knowledge for usability testing is referring to SFWRENG 4HC3 course notes, as usability testing is one of the major topics the course focuses on, with lots of examples and details. To obtain knowledge for dynamic tests, I plan on learning pytest to learn about dynamic testing in Python, which will be the team's language of choice for developing machine learning models.

   Jasmine: I will choose to focus on static testing knowledge since it is something I usually overlook and would like to improve on. To acquire knowledge in static testing, one approach would be to use online static testing tutorials and resources, especially platforms with specialized online courses such as Coursera or Udacity. Another approach would be to use an open-source static analysis tool and apply it to previous projects or example codebases to get hands on experience identifying common code issues, warnings, and even security vulnerabilities.

   Promish: One of the best ways to learn a framework is to read its documentation or watch YouTube tutorials. After that, testing it out in a separate IDE with a small example would be helpful for something like Pytest. For a better understanding of Agile development, we could look into resources that explain Agile principles and methods. Additionally, consulting our TA, supervisor, and professor would be valuable for understanding how our team can function effectively.