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

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

[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

[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

[Introduce this section. You can provide a roadmap of the sections to come. —SS]

3.1 Verification and Validation Team

[Your teammates. Maybe your supervisor. You should do more than list names. You should say what each person's role is for the project's verification. A table is a good way to summarize this information. —SS]

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]

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 1: Verification and Validation Team Table

[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]

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]

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]

4 System Tests

[There should be text between all headings, even if it is just a roadmap of the contents of the subsections. —SS]

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 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

4.1.2 Area of Testing2

...

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 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

4.2.2 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

Title for Test

1. test-id1

   Type: Functional, Dynamic, Manual, Static etc.

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:

4.2.3 Area of Testing2

...

4.3 Traceability Between Test Cases and Requirements

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

5 Unit Test Description

[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

This is where you can place additional information.

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.

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.

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.

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.

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.