Project Title: System Verification and Validation Plan for Software Engineering

Team 15, ASLingo
Andrew Kil
Cassidy Baldin
Edward Zhuang
Jeremy Langner
Stanley Chan

November 2, 2023

Revision History

Date	Version	Notes
Oct 31	Jeremy	Added in draft for sect 4.1 system tests for functional requirements
Nov 1	Stanley	Added sections 3, 3.1, and 3.2
Nov 1	Jeremy	Updated sec 4.1 with goals of each and added in new item for new FR
Nov 2	Cassidy	Updated sec 4.2 with tests for non-functional requirements
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

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

List of Figures

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

1 Symbols, Abbreviations, and Acronyms

symbol	description
T	Test

[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 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. —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

The following section aims to outline and describe the team's verification and validation plan over the course of the project. Parts of the project that are planned to be tested and verified will be the SRS, Design, and VnV Plan documents, as well as software testing.

3.1   Verification and Validation Team

Name	Roles and Responsibilities
Stanley Chan	Frontend verification, Computer Vision verification, SRS Verification
Andrew Kil	Frontend end-to-end testing, Computer Vision verification, Design Verification
Cassidy Baldin	Fullstack end-to-end testing, backend black box testing, Unit testing
Edward Zhuang	Backend performance testing, Computer Vision unit testing, SRS Verification
Jeremy Langner	Fullstack unit testing, frontend black box testing, VnV Plan Verification
McMaster SLC	Providing feedback during project development

3.2   SRS Verification Plan

SRS Verification will be done through in-group reviews performed by designated SRS verifiers (as mentioned in the VnV team table) and through TAs and peer reviewers from other groups.

SRS verifiers in the group will first review the SRS before the submission date, then the group will collectively iron out any potential issues that may arise from TA and peer review feedback.

[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]

2

3.4 Verification and Validation Plan Verification Plan

[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]

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]

4 System Test Description

4.1 Tests for Functional Requirements

4.1.1 Authentication

1. FRT1-A1

   Goal: User can make an account

   Control: Manual

   Initial State: User does not have a registered account with email. Currently on create an account page.

   Input: User enters their name, a valid email, a valid password containing 8 characters with at least one upper case, one number, and one special character into each appropriately labeled text field.

   Output: System displays successful account registration complete and prompts the user to sign in.

Test Case Derivation: Users need an account to record their progress.

How test will be performed: Manual test via user without an account and has a valid email.

Functional Requirement: FR3

2. FRT1-A2

Goal: User can sign into account

Control: Manual

Initial State: User is on the sign in page with a registered account.

Input: User enters email and associated password with registered account.

Output: System checks for existence of email and correct associated password with email and based on authentication signs in the users or displays and invalid credentials and prompts a resign in attempt.

Test Case Derivation: Valid credentials are required to sign in

How test will be performed: Manual user with an account will attempt to sign in with valid credentials and invalid credentials.

Functional Requirement: FR4

3. FRT1-A3

Goal: User can reset password to account

Control: Manual

Initial State: User is on the sign in page with a registered account.

Input: User requests reset password then inputs email

Output: System checks for existence of email and sends an email with a unique link to reset password credentials.

Test Case Derivation: Valid credentials are required to sign in

How test will be performed: Manual user with an account will attempt to reset password then resign in with old and new password.

Functional Requirement: FR13

4.1.2 ASL Learning Progression

1. FRT2-LP1

   Goal: User performs ASL signs

   Control: Manual

   Initial State: The user is in an ASL course prompt question or diagnosis and has a functioning webcam approved by application

   Input: User displays hand signs

   Output: The system output error if cannot recognize hands in camera view

   Test Case Derivation: System needs to be able to recognize ASL hand signs for functioning user progression and learning

   How test will be performed: Manual test with user testing hand signs within diagnostic and a progression course

   Functional Requirement: FR2

2. FRT2-LP2

   Goal: Complete diagnostic quiz

   Control: Manual

   Initial State: The system will prompt newly registered users with a diagnostic quiz to determine current ability

   Input: User goes to home page

   Output: The system starts the diagnostic quiz for user to compelete

   Test Case Derivation: Users need an initial ability level to provide next learning steps

   How test will be performed: Manual test via user creating a valid account

   Functional Requirement: FR6

3. FRT2-LP3

   Goal: User attempts progression based course

   Control: Automated

Initial State: The system creates unique progression courses with varying difficulty for user skill development

Input: User completes their diagnostic quiz or completes a progression course

Output: The system generates new progression based on completed progression and displays the new course to be completed

Test Case Derivation: Users shall be able to attempt new progression courses to improve their ASL skill

How test will be performed: Automated testing can be used to generate sample courses from given user

Functional Requirement: FR7

4. FRT2-LP4

Goal: User gets progression course based on previous course completions

Control: Automated

Initial State: The system creates unique progression courses with varying difficulty for user skill development

Input: User completes their diagnostic quiz or completes a progression course

Output: The system generates new progression based on completed progression and displays the new course to be completed

Test Case Derivation: Users shall be able to attempt new progression courses to improve their ASL skill

How test will be performed: Automated testing can be used to generate sample courses from given user

Functional Requirement: FR7

5. FRT3-LP5

Goal: System saved user progress

Control: Automated

Initial State: User is signed in to their registered account

Input: User completes their diagnostic quiz or completes a progression course

Output: The system saves the course and their results

Test Case Derivation: Users should be progressing in their skill thus the system needs to save their history of completions and approximate skill level

How test will be performed: Automated testing can be used to ensure courses are saved upon completion

Functional Requirement: FR8

6. FRT3-LP6

Goal: Get user sign feedback

Control: Manual

Initial State: User is currently working on an ASL prompt question

Input: User attempts appropriate sign

Output: The system determines if their sign action is accurate and displays message conveying the accuracy

Test Case Derivation: Users shall be able to see within a progression if they are doing the sign correct

How test will be performed: Manual user testing will occur with knowledge of ASL signs

Functional Requirement: FR10

Web Application

1. FRT3-U1

Goal: Access web application

Control: Manual

Initial State: User has a modern web browser.

Input: User enters the web app url into url textbox.

Output: System loads ASLingo homepage

Test Case Derivation: Users need to access web app for functionality.

How test will be performed: Manual test with user entering input.

Functional Requirement: FR9

Hardware

1. FRT4-HW1

   Goal: Access web camera

   Control: Manual

   Initial State: User has working built in or external webcam and recognized by their operating system.

   Input: User begins progression course which begins with a camera verification

   Output: System displays camera output or error if it cannot recognize camera.

   Test Case Derivation: Users need to access web camera for functionality.

   How test will be performed: Manual test with user starting course using working webcam.

   Functional Requirement: FR1

2. FRT4-HW2

   Goal: Monitor web camera useability

   Control: Manual

   Initial State: User has working built in or external webcam and recognized by their operating system.

   Input: User is currently within a progression course and camera error arises

   Output: System displays camera output or error if it cannot recognize camera

   Test Case Derivation: Users need to access web camera for functionality

   How test will be performed: Manual test with user currently in progression course that initiates camera error

   Functional Requirement: FR11

4.2 Tests for Nonfunctional Requirements

4.2.1 Usability Requirements

User Testing

1. NFRT1 - UT1

   Goal: The user is able to start the application with little to no prior training.

   Type: Dynamic, Manual

   Initial State: The user is given a link to the where the application is being hosted.

   Input: The user should be able to open and start the application without needing to ask for help or training first.

   Output/Result: The user is able to successfully open the application to start the learning process without prior training given by the testing team.

   How test will be performed: A sampling of representitative users will attempt to open and start the application and will pass the test if they can complete this task with no prior training. They will then score their user experience in the survey found in Appendix 6.2. The test should result in an overal average of users scoring 75% or higher for this part of the survey.

   Non-Functional Requirement: UHR1

2. NFRT1 - UT2

   Goal: The user is able to understand how to use complete a lesson with little to no prior training.

   Type: Dynamic, Manual

   Initial State: The user signs into the web application with account information and is ready to start learning ASL.

   Input: After the user signs into the application with account information, they will start trying to learn ASL using the various learning tools in the application with no prior training.

   Output/Result: The mechanics of the application should not hinder the user's progress of learning, and the user should feel as though the

learning tools are intuitive and easy to navigate. The user should be able to successfully complete the first inroductory lesson without being confused by the mechanics of the application.

How test will be performed: A sampling of representitative users will attempt to use the application and will pass the test if they can complete their first lesson without needing to ask for help with how the application works. They will then score their user experience in the survey found in Appendix 6.2. The test should result in an overal average of users scoring 75% or higher for this part of the survey.

Non-Functional Requirement: UHR1

3. NFRT1 - UT3

Goal: The user is able to understand how to use the application with various hearing abilities and little to no prior training.

Type: Dynamic, Manual

Initial State: The user signs into the application to learn ASL.

Input: The user will attempt to complete the first introductory lesson regardless of their level of hearing ability.

Output: The user should be able to successfully complete the intorductory lesson without needing to be able to hear the instructions to complete the introductory tasks required of them.

How test will be performed: A sampling of representative users with various hearing abilities will attempt to use the program and will pass the test if they can successfully complete the first lesson of the program. They will then score their user experience in the survey found in Appendix 6.2. The test should result in an overall average of users scoring 75% or higher for this part of the survey.

Non-Functional Requirement: UHR2

4. NFRT1 - UT4

Goal: The user is able to personalize their account with little to no prior training.

Type: Functional, Dynamic, Manual

Initial State: The user is signed into their account and wants to change some of their information.

Input: The user will attempt to change their information in their account.

Output: The user should be able to change their information in their account without much difficulty.

How test will be performed: A sampling of representative users will attempt to change key features of their account like username or phone number and be able to save their changes without asking for help. They will then score their user experience in the survey found in Appendix 6.2. The test should result in an overall average of users scoring 75% or higher for this part of the survey.

Non-Functional Requirement: UHR3

4.2.2 Performance Requirements

Performance Testing

1. NFRT2 - PT1

   Goal: The application should respond to user input within 1 second.

   Type: Functional, Dynamic, Manual

   Initial State: The application prompts the user with a question during a lesson.

   Input: The user should respond to the application's prompt.

   Output: The system should register the user's input and respond to the user quickly.

   How test will be performed: A user will respond to the prompt given by the application. The generated output must take less than 1 second to be produced. This test will be performed multiple times, and the run time for 95% of these tests must be less than 1 second for the test to pass.

   Non-Functional Requirement: PR1

2. NFRT2 - PT2

   Goal: The application should be able to accurately determine if the user has signed the correct response to the prompt 95% of the time.

   Type: Dynamic, Manual

Initial State: The application prompts the user with a question during a lesson.

Input: The user should sign in response to the application's prompt.

Output: The application should register the user's signed input and determine if they have signed the required action correctly.

How test will be performed: A user will respond to the prompt by signing in front of the application. The application should then determine if the user has signed the correct respond or not, and output a message telling the user what the application determined. This test will be performed multiple times, and the application should correctly identify if the user is correct or not 95% of the time.

Non-Functional Requirement: PR2

3. NFRT2 - PT3

Goal: The system should show the user if their input needs to be adjusted.

Type: Manual, Static

Initial State: The application prompts the user with a question during a lesson.

Input: The user attempts to sign in response to the application's prompt but their camera is not set up properly.

Output: The application should register that it cannot be confident in determining if the sign the user is displaying is accurate, so it should prompt the user to fix their camera settings before they are allowed to continue.

How test will be performed: A user will try to respond to the prompt by signing in front of the application, but with the user out of the focus of the camera. The application should not be able to determine if the user's sign is correct or not and should prompt the user to adjust their camera settings. This test will be performed multiple times with various input and camera conditions to ensure the requirement has been tested effectively. The developers will verify that the user is given feedback where appropriate.

Non-Functional Requirement: PR3

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

14

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?

User Experience Survey

[This is a section that would be appropriate for some projects. —SS]

Appendix — Reflection

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

Appendix — Reflection

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

1. 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 etc. You should look to identify at least one item for each team member.

2. 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?