Project Title: System Verification and Validation Plan for Software Engineering

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

November 1, 2023

Revision History

Date	Version	Notes
Oct 31	Jeremy	Added in draft for sect 4.1 system tests for functional requirements
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

[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

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

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]

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]

[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 Test Description

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]

User can make an account

1. FRT1-A1

   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

   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

4.1.2 ASL Learning Progression

1. FRT2-LP1

   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

   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

   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

   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

   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

   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

   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

   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

   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

[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]

[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]

4.2.1 Area of Testing1

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:

4.2.2 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?

[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

[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. 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?