Project Title: System Verification and Validation Plan for Software Engineering

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

November 3, 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
Nov 2	Stanley	Added design verification plans
Nov 2	Andrew	Updated Sections 3.4-3.7
Nov 3	Edward	Added all of Section 2
Nov 3	Andrew, Cassidy, Jeremy, Stanley	Added Reflections
Nov 3	Cassidy	Added section 1 and usability survey questions
Nov 3	Edward	Added Reflection
Nov 3	Jeremy	Added in sect 4.3 table matricies

Contents

List of Tables

1 Symbols, Abbreviations, and Acronyms

Table 1: Naming Conventions and Terminology

Term, Abbreviation, or Acronym	**Description**
A	Shorthand for Assumption
ASL	Shorthand for American Sign Language. It is a form of sign language primarily used in the US and in parts of Canada
ASLingo	The commercial name for the project
CV	Refers to Computer Vision, the field of technology that involves processing visual input to achieve various means.
CR	Shorthand for 'Cultural Requirements', a subsection of Non-Functional Requirements.
HSR	Shorthand for 'Health and Safety Requirements', a subsection of Non-Functional Requirements.
FR	Shorthand for Functional Requirements
LR	Shorthand for 'Legal Requirements', a subsection of Non-Functional Requirements.
LFR	Shorthand for 'Look and Feel Requirements', a subsection of Non-Functional Requirements.
MSR	Shorthand for 'Maintainability and Support Requirements', a subsection of Non-Functional Requirements.
OER	Shorthand for 'Operational and Environmental Requirements', a subsection of Non-Functional Requirements.

OpenCV	Refers to the Open Computer Vision Library library available for free to developers in order to develop Computer Vision applications.
PR	Shorthand for 'Performance Requirements', a subsection of Non-Functional Requirements.
SR	Shorthand for 'Security Requirements', a subsection of Non-Functional Requirements.
UHR	Shorthand for 'Usability and Humanity Requirements', a subsection of Non-Functional Requirements.

2 General Information

2.1 Summary

As a maching learning-based image recognition web app, ASLingo has many areas to be tested. The overall software will be broken down into modules. There will be a front-end, a back-end, a database, and a machine learning model which all need to be separately tested, along with physical hardware and compatibility. End-to-end testing may be required as well, which the team will determine down the line.

2.2 Objectives

This document aims to outline the testing plan for ASLingo in order to create a functional and reliable product for users that aligns with the specified requirements. The team seeks to build confidence in stakeholders and users that the software is correct and meets or exceeds the initial intended goals, resulting in an overall satisfactory user experience.

2.3 Relevant Documentation

Below is a list of the relevant documentation referenced within the Verification and Validation Plan.

The Development Plan outlines the roles of each team member and the areas that each member will focus on. This breakdown of team responsibilities allows the team to assign testing roles accordingly. This document also contains the tools that the team plans on using for testing.

The Software Requirements Specification lists the functional and non-functional requirements which will aid in testing by formulating a testing plan to meet each requirement. Non-functional requirements should be tested such that the fit criteria are met.

The Hazard Analysis identifies failure modes to determine the implementation strategies to mitigate them. These will be used as a part of the testing plan to ensure that the failures are covered.

The Module Guide divides the software into modules. The team will build the testing plan around the modules.

The Module Interface Specification further decomposes the software's modules into specific access routines. The team will build the testing plan such that each function and routine works as intended.

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 the codebase of the overall project.

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.

3.3 Design Verification Plan

Design verification will also be done in-group before each respective milestone. Peer reviewers and TAs will provide additional feedback after the deadline. Verification goals for the design includes, but is not limited to; accessibility, user experience, error handling, etc.
We will also perform iterative testing in between milestones with the McMaster Sign Language Club. This is to ensure that new features added to the design are user friendly and intuitive to use from the perspective of the average user. As such, we will designate members of the McMaster Sign Language Club as the primary personas for our user interface design, in an effort to develop a user experience as human-centered as possible.

To summarize, the basic verification workflow for our design will be as follows:
Before a milestone deadline:

- In-group verification

After a milestone deadline:

- TA feedback
- Peer review feedback

Periodically between milestones:

- McMaster SLC iterative stakeholder testing

3.4 Verification and Validation Plan Verification Plan

The current approach to validate our Verification and Validation plan is to rely on third party reviews from other teams as well as mutation testing to verify whether our tests work as intended. The mutation tests will aim to accomplish the following:

- ☐ Did the test manage to detect the mutation and behave accordingly? (I.e. pass/fail when it should)

- ☐ Did the test's behaviour stay consistent with its behaviour prior to the mutation's introduction, and if so, was this expected?
- ☐ Did the test accomplish provide sufficient information to act upon the failure induced by the mutation, and if so, would this be enough in the event of a non-controlled failure?

3.5 Implementation Verification Plan

The current implementation verification plan is a combination of both static and dynamic testing. We plan to utilize static methods like regular code walkthroughs and inspections before merges with the main branch at the end of project milestones. These reviews should be performed by at minimum 2-3 team members. Dynamic testing methods include the fundamental system tests mentioned in the following section 4.1 for both functional and non-functional-requirements of the final product. Should any issue arise through either avenue of of implementation verification, the team as a whole should work on and approve the final solution that is developed as a result.

3.6 Automated Testing and Verification Tools

As already mentioned prior in the Development Plan, Back-end Unit Testing plans to make use of the Pytest Unit Testing Framework with the 'pytest-cov' plugin to provide code coverage metrics. Front-end Unit Testing Plans to make use of Jest for both Unit Testing and for Code Coverage Metrics, utilizing its built in '–coverage' tool upon runtime.

For code verification, this was also mentioned previously in the Development plan. We plan to follow PEP8 Python coding standard and use the 'Flake8' linter to help enforce and check if our upholds said standard. As for the JavaScript front-end, we plan to follow the Airbnb Style Guide. Additionally, as this was not discussed within the Development Plan, the use of the 'ESLint' linter to help enforce such standards.

3.7 Software Validation Plan

The software will be validated utilizing a mix of Black Box/Usability Testing and White Box Testing. The Black Box/Usability Testing will be conducted by external groups to our core team, ideally by members of our stakeholder

group i.e. the McMaster University Sign Language Club as they are the closest to our target demographic of individuals who are interested in learning sign language. This will give us the most accurate test data for the end-users who will be using the final product. The Black Box Testing sessions will solely consist of instruction to form a specific hand sign, their formation of the hand sign for input and a resulting correct or not output.

White Box Testing will be performed internally by team members throughout the course of development. Post every milestone, at minimum 2 members will run through the system and test for the key functionality that was added in the milestone. Since team members are the ones who have the intimate understanding of the core functionality of the system, it only makes sense for use to be the ones who perform the testing.

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, FR12

5. FRT2-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. FRT2-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 6.1. 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 6.1. 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 6.1. 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

6.1. 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

Functional Requirements to System Test Requirements

	FR Req.												
System Test	1	2	3	4	5	6	7	8	9	10	11	12	13
FRT1-A1			X										
FRT1-A2				X									
FRT1-A3													X
FRT2-LP1		X											
FRT2-LP2						X							
FRT2-LP3							X						
FRT2-LP4							X					X	
FRT2-LP5								X					
FRT2-LP6										X			
FRT3-U1									X				
FRT4-HW1	X												
FRT4-HW2											X		

	UHR			PR		
System Test	1	2	3	1	2	3
NFRT1-UT1	X					
NFRT1-UT2	X					
NFRT1-UT3		X				
NFRT1-UT4			X			
NFRT1-PT1				X		
NFRT1-PT2					X	
NFRT1-PT3						X

Non Functional Requirements to System Test Requirements

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]

6 Appendix

6.1 Usability Survey Questions

User Experience Survey

1. How was it to start using the application on a scale of 1 - 10?
   [1 = very hard to start, 10 = very easy to start]

2. How was it for you to learn how to complete the first lesson on a scale of 1 - 10?
   [1 = very hard to learn, 10 = very easy to learn]

3. How comfortable were you with using the application at your current ASL skill level on a scale of 1-10?
   [1 = very uncomfortable to use, 10 = very comfortable to use]

4. How was it to change your profile information on a scale of 1 - 10?
   [1 = very hard to change, 10 = very easy to change]

5. How did you find your overall learning experience using the ASLingo application on a scale of 1 - 10?
   [1 = very bad experience, 10 = very good experience]

6.2 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.

   (a) Andrew Kil: Black Box
   I'll be mainly focusing on Dynamic Testing Methods like Usability testing for when we bring it the general user base. Being able

to successfully elicit feedback from our test groups (i.e. the McMaster Sign Langauge Club) is a vital step in bridging the gap between developers and the market teat the system is designed for.

(b) Cassidy Baldin: Pytest, Jest and Linters
I will be mainly working on the back and front end of the project, so the skills I will need to acquire to test both of these aspects are the use of pyTest and Jest to ensure the project is working as intended. I also have little experience using different linters for these frameworks, so this will also be an apsect I will have to learn for this project.

(c) Edward Zhuang: Pytest and ML Model Testing
As my focus is on the backend and the OpenCV component, I will be responsible for testing these parts. Our team agreed to use Pytest as our Python unit testing framework, so I'll have to learn that. As well, I will focus on training and testing the machine learning model so that it is suitable for a functional product. I have a bit of experience with model testing in the past, but not for image recognition, so I will need to learn the ins and outs of that.

(d) Jeremy Langner: Manual & Automated Testing
Knowledge of when to use manual vs automated testing is necessary for VnV plan since the system tests for Functional rquiremenents have to be clearly seperated. This can be done by practicing how to understand the specific requirement and corresponding use case and then applying basic software testing princples to ensure what is testable manually and what can be done with automation frameworks and how. Typically any requirments with manual interaction or system integration will require manual testing and any coding or business logic requirements can be done with automation.

(e) Stanley Chan: Jest Framework
Since I'll mainly be working on the frontend component of the project, I'd like to focus on learning some JavaScript testing li-

braries, specifically Jest. As I also want to be involved partly in the computer vision aspect, I'd like to familiarize myself with linters for both JavaScript and Python to standardize code practices throughout the project.

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?

   (a) Andrew Kil: Black Box
   The main approach I will take to learn the necessary skills to carry out the Usability trials is through online resources found trough the internet. Naturally the standard of professionalism I have for these resources would be held to at the very least those who have ran Usability testing sessions for a successful project that was brought to market. Another approach is to attempt to seek out and consult those previously mentioned individuals those online solicitation for more sound advice. Of the two aforementioned methods, I will most likely be pursuing the former due to its ease of access as well as time efficiency.

   (b) Cassidy Baldin: Pytest, Jest and Linters
   To acquire these skills, I will combine the use of online written tutorials, videos by experts in these frameworks, as well as talking to others I know who have hands on experience with these tools to understand how they work and how to use them efficiently. I will also be able to rely on some of my previous experience with testing tools from other courses to learn these new ones.

   (c) Edward Zhuang: Pytest and ML Model Testing
   There are many online Pytest tutorials that demonstrate how to unit test Python software, which I will use to learn it. As for testing machine learning models, I already have some previous experience. On top of that, there are tutorials on YouTube that show how to effectively train and test machine learning models,

which I will use.

(d) Jeremy Langner: Manual & Automated Testing
Learning how to understand requirements has been learned with SFWRENG 3RA3 and with many subsequent courses but also comes up in many online coding problems or problem solving skills learned in real world/industry. Software testing principles have been learned in class in SFWRENG 3S03 and can also be learned thorugh various online sources like Guru99, GeeksForGeeks, JavaTPoint etc. The difficulty with this project will be the large intergation between our systems which will be great practice

(e) Stanley Chan: Jest Framework
Documentation and other online resources will be the main source of learning specific syntax and practices regarding testing libraries and linters. Additionally, material from previously taken courses like 3S03: Software Testing will be put into practice for concepts like code coverage, white box testing, etc.