System Verification and Validation Plan for Bridging Gaps: AI for Diagram Accessibility

Team 22, Reading4All
Nawaal Fatima
Dhruv Sardana
Fiza Sehar
Moly Mikhail
Casey Francine Bulaclac

January 6, 2026

Revision History

Date	Version	Notes
Oct 27, 2025	1.0	Initial version of document.

Contents

List of Tables

1 Symbols, Abbreviations, and Acronyms

Table 1: Abbreviations and Acronyms Used in the Verification and Validation Plan

Acronym	Definition
AI	Artificial Intelligence
API	Application Programming Interface
AODA	Accessibility for Ontarians with Disabilities Act
AR	Accessibility Requirement
CAD	Canadian Dollar
CI	Continuous Integration
CD	Continuous Development
CR	Cultural/Regulatory Requirement
DP	Development Plan
FR	Functional Requirement
GUI	Graphical User Interface
HA	Hazard Analysis
LFR	Look and Feel Requirement
LR	Legal Requirement
MG	Module Guide
MIS	Module Interface Specification
ML	Machine Learning
MS	Maintenance Support Requirement
NFR	Nonfunctional Requirement
NVDA	NonVisual Desktop Access (screen reader)
OER	Operational and Environmental Requirement
OCR	Optical Character Recognition
PII	Personally Identifiable Information
PR	Performance Requirement
R	Requirement
RL	Release Requirement
RFT	Reliability and Fault Tolerance Requirement
SCR	Safety-Critical Requirement
SRS	Software Requirements Specification
SR	Security Requirement
SSO	Single Sign-On
TLS	Transport Layer Security
UI	User Interface
UHR	Usability and HuYmanity Requirement
VnV	Verification and Validation
WCAG	Web Content Accessibility Guidelines

This document outlines the method *Reading4All* team will take to ensure the software built meets the intended requirements. To make certain that the product is built correctly (verification) and that the right product is built (validation) *Reading4All* team has structured this document to reflect our plan. We achieve this by providing our objectives, and laying out a comprehensive test plan while referring to the Software Requirements Specification (SRS), Module Guide (MG) and Module Interface Specification (MIS).

First, Section 2 outlines what software is being tested with its expected functionality. Here, the objectives and relevant documentation is mentioned. Next, the plan for verification and validation is thoroughly explored, touching on the previously mentioned documents. Finally, each test is mapped to one or more requirements highlighted in the SRS, ensuring th *Reading4All* team delivers what we promised. At the time of writing this document, unit tests (Section 5) was not applicable as it was too early in the project timeline.

2 General Information

2.1 Summary

The software being tested is called *Reading4All*. This software will utilize artificial intelligence (AI) and machine learning (ML) techniques to provide detailed, context-informed alternative text for complex technical images, specifically those found in post-secondary Science, Technology, Engineering and Mathematics (STEM) course materials. *Reading4All* will allow users to upload images and then it will automatically produce corresponding alternative text (alt text), that meets the described criteria in Appendix 6.1 . The system is intended for use by McMaster University students and faculty, therefore it will include user validation through McMaster's sign-on, ensuring that only verified users can access the *Reading4All* system. In addition, the system will allow users to edit the generated alternative text, view a history of uploaded images and their alternative text within that session and download the final outputs in their desired formats.

2.2 Objectives

The objective of this Verification and Validation (VnV) plan is to build confidence in the correctness, accessibility and usability of the *Reading4All* system. The plan focuses on ensuring that the system generates, accurate, detailed and contextually appropriate alternative text for complex STEM images, as this is the main functionality of the software. It also aims to verify that the systems interface is accessible an usable for individuals with visual impairments, who are one of the main users of the software. The last object is to demonstrate effective and accessible usability in secondary features such as editing the outputted alternative text, viewing session history and file downloading. Verifying and validating these objectives is essential to ensure that users can benefit from th *Reading4All* system and it can effectively fulfill its goal of making STEM diagrams more accessible.

Some objectives are out of scope from this VnV plan due to the time and resource limitations of the project. The external libraries that might be used in the development of the system, including PyTorch, TensorFlow, Scikit-Learn, Pandas and frontend frameworks, will not be verified by our team, as they will be assumed to have been tested and validated by their implementation teams. The McMaster sign-on authentication service will also not be tested, as its maintained and run by the university.

2.3 Challenge Level and Extras

The challenge level of the project is set at a *general* level that will include two additional components (extras). The first additional component will be a Norman's Principles report that will evaluate the design of our final product to ensure that it optimizes usability and accessibility. The second additional component will be a user manual that will include comprehensive documentation to guide users in using the final product effectively.

2.4 Relevant Documentation

The system's Verification and Validation (VnV) plan will reference the following documents to aid in the project's assessment and testing:

1. **Software Requirements Specification** (SRS (2025)): This document outlines the key components for the VnV plan as it details the

functional and non-functional requirements of the product. Ensuring that our testing satisfies these requirements is essential to meeting the goals of the project.

2. **Design Document - Module Guide** (MG (2025)): This document outlines how the system is divided into separate module and their respective functions. This structure helps the VnV Plan by making it easier to test, trace, and confirm that each module works correctly and meets the requirements.

3. **Design Document - Module Interface Specification** (MIS (2025)): This document outlines how the modules of the system works and how they interact with each other. This aids in our VnV plan as it defines how to validate the testing of interactions between the individual parts of the system.

3 Plan

This section outlines the strategies and techniques that will be used for verifying and validating the *Reading4All* system, throughout the varying development phases. This includes verifying and validating the SRS, the design, the VnV plan, and the implementation. This section also defines the *Reading4All* team's responsibilities in completing this plan and implementing the specified strategies, to ultimately ensure the system meets our requirements and objectives.

3.1 Verification and Validation Team

The VnV team will ensure that all SRS requirements are thorough, accurately implemented and tested to ensure our system meets them. Table 2 defines each *Reading4All* team members role in confirming that these requirements are well defined and fulfilled in the system. These roles will remain the same across different verification tasks, allowing for consistency and a deeper understanding of each focus area.

Table 2: Verification and Validation Responsibility Breakdown

Name	**Focus Area**	**Responsibility**
Moly Mikhail and Fiza Sehar	Functional Requirements Leads	Verifies all corresponding requirements in the SRS are thorough, clearly defined, attainable and measurable. Completes manual testing on the software to ensure that the functional requirements are met. Also oversees any written unit tests to ensure they correctly test the system and intended functionality.
Nawaal Fatima	User Testing Preparation Lead	Coordinates the user testing sessions, and oversees the planning and execution of these sessions.
Casey Francine Bulaclac	Usability and Humanity Requirements Lead	Verifies all corresponding requirements in the SRS are thorough, clearly defined, attainable and measurable. Completes manual testing to ensure all the related requirements are met. Also verifies that WCAG 2.1 crtierias are met, prior to automated testing.
Dhruv Sardana	Performance, Operational and Environmental Requirements	Verifies that corresponding requirements are thorough, clearly defined, attainable and measurable. Also, validates that the implemented system meets these requirements through extensive testing.
All *Reading4All* Team Members	Look and Feel Requirements Lead	All team members are responsible for verifying that the corresponding requrieemnts in the SRS are thorough, clearly defined, attainable and measurable. The team will validate that the final system meets the specified requirements prior to user testing.
Ms. Jingchuan Sui (Supervisor)	Overall Usability and Quality of Alt Text Reviewer, User Testing Material Reviewer	Reviews and provides feedback to team on the overall usability of the system and the quality of the generated alternative text. Reviews user testing materials to ensure they align with accessibility standards and assits the team in completing the automated evaluation of the system against WCAG 2.1 criteria.

3.2 SRS Verification

The verification of the *Reading4All* SRS will be completed through a team review, peer review, and supervisor feedback, in that order. Feedback from each step will be incorporated iteratively to ensure the document is reviewed and the version presented to our supervisor reflects the most accurate and complete requirements. The following details the strategies used for SRS verification:

- **Team Review**: As the *Reading4All* team has already reviewed the outlined requirements in the SRS document, a team review will consist of each team member independently evaluating the document against the checklist provided in Appendix 6.2.
- **Peer Review**: A peer review will be completed by Team 10 (One of a Kind) to provide feedback on our SRS document. They will review the document against the same checklist provided in Appendix 6.2, ensuring consistency between how both teams evaluate the document. This process will allow external reviewers who understand the documents technical details and software engineering context to provide feedback on the completeness, understandability and the overall quality of the requirements.
- **Supervisor Feedback**: We will dedicate one of our weekly meetings with our supervisor to verifying the SRS document. During this meeting, will explain each functional and non functional requirement to ensure every requirement has been documented. This process will help our supervisor gain an understanding of the specific requirements we have outlined and keep them in mind during future validation of our system.

3.3 Design Verification

The design verification will be accomplished through the following:

- **Peer Review:** The verification of our design will rely on reviews conducted by our fellow classmates who will evaluate our design documents including our Module Guide (MG) and Module Interface Specification (MIS) using the following checklist in Appendix 6.4. The peer review will be conducted in Week 10 and Week 16 following the due date for submission of these design documents. Any items on the checklist not satisfied by our team on these documents will be addressed through making issues on GitHub.

- **Formal Review:** In a formal review, our supervisor, Ms. Jing, will verify that the user interface design supports accessibility in accordance with WCAG 2.1 standards. Additionally, Capstone teaching assistants and instructors will assess the technical aspects of the system, focusing on the machine learning architecture to ensure it is well designed. The formal review will also be conducted in Week 10 and Week 16 following the due date for submission of these design documents.

3.4 Verification and Validation Plan Verification

The Verification and Validation Plan Verification for Reading4All will be completed through a team and peer review, as well as mutation testing to assess the effectiveness and correctness of our tests. Furthermore, any criteria not met, or failed mutation tests identified from this process will be documented as GitHub issues, to ensure they are tracked and resolved. The following details the strategies used for verifying the VnV plan:

- **Team and Peer Review:** The *Reading4All* team will complete an internal review to ensure that the plan is complete and aligns with the SRS requirements, as well as other documents. Both our team and peer review will be completed by reviewing the V&V document against the checklist provided in Appendix 6.3.

- **Mutation Testing:** Mutation testing will be used to determine whether our unit tests correctly identify errors within the system. Small errors will be intentionally be introduced into the prototype code related to the main functionalities to check whether the test cases can detect unexpected behavior.

3.5 Implementation Verification

The implementation verification for *Reading4All* will utilize the unit tests defined in section 5, system tests defined in section 4, static analyzers and code inspections. Test results, coverage metrics and pull request reviews will be collected as evidence that verification activities have been completed. The following details the strategies used for verifying the implementation plan:

- **Unit and System Tests**: The unit and system tests will be run to ensure the system behaves as expected. The unit tests will check that individual functions within he implementation are working correctly, helping us narrow down possible issues, and thoroughly test functions with different inputs. The system tests will verify that the implementation fulfills all functional and non-functional requirements.

- **Static Analyzers**: Static analyzers will be used to automatically review any committed code for errors, and ensure the implementation meets the specified coding standard. Tools such as Flake8 and Coverage.py will be integrated into our development process. Flake8 will identify syntax and style issues, while Coverage.py can verify that any new code written has unit tests associated. Additionally, all unit tests will be executed automatically through GitHub Actions whenever new code is pushed or a pull request is opened. This will ensure that recent changes do not introduce any errors into previously completed features. Using these static analyzers and workflows will help us to continuously verify our implementation, ensuring it meets the design requirements, functional and non functional requirements.

- **Code Inspections**: As part of our development process, when a feature is completed, team members are required to open a pull request. The pull request highlights all the code changes made and is reviewed by other team members, before merging. This process ensures that the multiple team members are aware of the changes being made, and provides an opportunity to give feedback on the implementation decisions. Additionally it also helps verify that the code fulfills the intended functionality.

3.6 Automated Testing and Verification Tools

The programming language being used in the front-end will be Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS), and the back-end will be programmed using Python for machine learning applications. The

AI Generated Alternative Text tool will make use of the following automated testing and verification tools:

- **Automated Accessibility Testing**: Automated web accessibility testing tools such as the WAVE Web Accessibility Evaluation Tool will be used to verify that the tool adheres to the Web Content Accessibility Guidelines (WCAG) 2.1 guidelines. These tools will automatically scan for accessibility issues such as missing alternative text and color contrast violations.
- **Automated Performance Testing**: `cProfile`, a python standard library, will be used to determine the execution time of different functions within the system. This helps to identify bottlenecks in the code and ensure that the system stays within the expected time limits as stated in our performance requirements.
- **Unit Testing Framework, Linters, Coverage Tools, and Continuous Integration**: These tools have been detailed and outlined in the Expected Technologies section of our Development Plan (DP (2025)) document under Table 3. These tools will ensure to help us follow our coding standards of PEP8 and help verify our defined programming languages.

3.7 Software Validation

The AI Generated Alternative Text tool will be validated through the following:

- **User Testing**: Stakeholders of this project including students experiencing visual and cognitive disabilities will perform realistic tasks such as uploading multiple images and downloading the alternative text to validate that it meets their needs and requirements. The team will be present during these sessions to observe and note any challenges the users face. The participants will also be asked follow-up questions highlighted in Appendix 6.6 to gain additional feedback and insight on their overall experience of using the tool.
- **Formal Reviews**: Reviews will be conducted with the stakeholders of the project and our supervisor, Ms. Jing, to validate that the requirements outlined in our SRS document are fulfilled and met by our tool.

Additionally, the Rev 0 demo with Ms. Jing will serve as a validation review and allow the team to receive feedback to improve the tool.

4 System Tests

This section details tests to cover all requirements as listed in the Software Requirements Specification (SRS) for the *Reading4All* team's project. The tests ensure that the system performs to the predefined standards and meets user needs.

4.1 Tests for Functional Requirements

The tests below cover all six functional requirements as defined in the SRS document. The following tests will help validate that the system meets the requirements.

Tests for Functional Requirements

FR-ST 1. **Control:** Automatic.
Initial State: System is running and ready to accept an image upload.
Input: Upload files in the following formats:

- Valid: diagram1.jpeg, diagram2.png
- Invalid: diagram3.gif, diagram4.pdf

Output: The system accepts .jpeg and .png images and displays an error message (e.g., "Invalid file type") for .gif and .pdf uploads.
Case Derivation: According to the **FR 1** criterion, the system must accept JPEG and PNG formats and reject all others with proper feedback. Therefore, valid formats are processed, and invalid formats trigger an error.
How test will be performed: The test can be performed by uploading a sample set of image files in different formats (JPEG, PNG, GIF, PDF) to the system. The system's responses will be observed to confirm that only JPEG and PNG files are accepted,

while others trigger an appropriate error message.

FR-ST 2. **Control:** Automatic
Initial State: System running with image upload functionality active.
Input: Upload a set of test diagrams (diagram1.png, diagram2.jpeg).
Output: The system generates alternative text descriptions for each uploaded image that meet pre-determined quality or clarity criteria (e.g., contains key diagram elements, concise description, no missing components).
Case Derivation: As specified by **FR 2** criterion, the system must correctly process JPEG and PNG files while rejecting all other formats. Therefore, the expected outcome is that valid images are accepted without error, and invalid formats trigger a clear feedback message to the user.
How test will be performed: The test can be performed by uploading a test set of sample diagrams to the system and reviewing the generated alternative text. The generated text will be compared against predetermined quality criteria or expected reference outputs to verify accuracy and completeness.

FR-ST 3. **Control:** Manual
Initial State: Alternative text has been generated for at least one uploaded image.
Input: Use screen readers such as NVDA, JAWS, and VoiceOver to read the outputted alternative text.
Output: Alternative text is fully read aloud by at least the most common screen readers without truncation, misreading, or formatting errors.
Case Derivation: Given the **FR 3** criterion for compatibility with commonly used screen readers, the expected outcome is that the generated alternative text will be fully readable and correctly interpreted by tools such as NVDA, JAWS, and VoiceOver without truncation or mispronunciation.
How test will be performed: The test can be performed by enabling common screen readers such as NVDA, JAWS, and VoiceOver

to read the generated alternative text aloud. Observations will confirm whether the text is read fully, clearly, and without formatting or accessibility issues.

FR-ST 4. **Control:** Manual
Initial State: Generated alternative text is visible to the user.
Input: User edits the outputted text (adds words, deletes sentences, modifies phrasing) and saves changes.
Output: The system reflects the user's edits accurately and stores the updated version without loss of data or formatting errors.
Case Derivation: As stated in **FR 4** criterion, users must be able to modify any part of the generated alternative text and save their changes. The expected result is that all edits are accurately captured, stored, and displayed without data loss or formatting issues.
How test will be performed: The test can be performed by selecting the generated alternative text and performing a series of edits—adding, deleting, and modifying words—then saving the changes. The output will be reviewed to ensure that edits are accurately reflected and retained.

FR-ST 5. **Control:** Automatic
Initial State: User is logged in and has uploaded at least one image with generated alt text.
Input: Upload more than `SESSION_HISTORY_MAX` images sequentially within the same session, then navigate through the session interface.
Output: The most recent `SESSION_HISTORY_MAX` uploaded images and their corresponding alt texts remain visible and accessible until the user logs out or the session ends. Older history records are not displayed.
Case Derivation: According to **FR 5** criterion, users must be able to view previously uploaded images and their corresponding alternative texts within the same session. To ensure a high system performance, only the `SESSION_HISTORY_MAX` most recent records are displayed.

Therefore, the expected outcome is that users can access and review only the `SESSION_HISTORY_MAX` prior uploads without reloading or re-uploading them during an active session.
How test will be performed: The test can be performed by uploading multiple images within the same session, then navigating across different pages or refreshing the interface. The test will verify that all uploaded images and their corresponding alternative texts remain visible until the session ends.

FR-ST 6. **Control:** Automatic
Initial State: Login page displayed.
Input: Access for login is defined below:

- Valid credentials: McMaster University email and password
- Invalid credentials: non-McMaster email or incorrect password

Output: Access granted only to users with valid McMaster credentials. Invalid attempts are rejected with an appropriate error message (e.g., "Invalid login credentials").
Case Derivation: As outlined in the **FR 6** criterion, only users with verified McMaster University credentials should gain access to system features. The expected outcome is that valid users can log in successfully, while unauthorized or invalid attempts are denied with an appropriate error message.
How test will be performed: The test can be performed by attempting to log in using both valid McMaster University credentials and invalid credentials. The system's behavior will be reviewed to confirm that only verified users gain access, while invalid attempts produce an appropriate error message.

FR-ST 7. **Control:** Automatic
Initial State: User logged in and an active session is established.
Input: User remains logged in for more than `SESSION_TIMEOUT` without performing any actions.
Output: The session expires automatically and the user is prompted to login again before being able to access the `Reading4All` system

again. **Case Derivation:** According to **FR 5** criterion, users must be able to view previously uploaded images and their corresponding alternative texts within the same session. To ensure the `Reading4All` system is as secure as possible, a user session must expire after the `SESSION_TIMEOUT` period of inactivity. This will help prevent unauthorized users from accessing the system. without reloading or re-uploading them during an active session.
How test will be performed: The test can be performed logging into the system and not completing any actions for a duration exceeding the `SESSION_TIMEOUT` and then attempting to access the system. This test will verify that a session is expiring as expected and users are prompted to login again.

4.2 Tests for Nonfunctional Requirements

The tests below cover all non-functional requirements as defined in the SRS document. The following tests will help validate that the system meets the requirements.

Tests for Non-Functional Requirements

NFR-ST 1. **Text Resizing and Contrast Accessibility**

Type: Non-Functional, Manual, Dynamic.
Covers: LFR-AR1, LFR-AR2, LFR-AR3, LFR-AR4.
Initial State: Interface displayed in a standard browser with accessibility tools enabled.
Input/Condition: The user adjusts browser zoom to the maximum allowed and reviews color usage.
Output/Result: Text resizes correctly without overlap; information is not conveyed by color alone; contrast meets accessibility thresholds; all images have alternative text.

How test will be performed: The tester will manually adjust zoom levels, use color contrast tools, and run screen reader tests to confirm accessibility compliance.

NFR-ST 2. **Interface Style and Branding Verification**.

Type: Non-Functional, Manual, Dynamic.
Covers: LFR-SR1, LFR-SR2, LFR-SR3.
Initial State: System interface displayed in a browser.
Input/Condition: Visual inspection of layout, font, colors, branding, and adherence to Norman's design principles.
Output/Result: Interface maintains modern and simple style, McMaster branding is present without interfering with usability, and design elements comply with Norman's principles.

How test will be performed: Tester will review the interface against the style guide, branding requirements, and Norman's design checklist to ensure compliance.

NFR-ST 3. **Usability Efficiency and Learnability**.

Type: Non-Functional, Manual, Dynamic.
Covers: UHR-EUR1, UHR-EUR2, UHR-EUR3, UHR-EUR4
Initial State: System interface available to first-time and returning users.
Input/Condition: Users perform key actions: login, upload images, generate alt text.
Output/Result: Users complete tasks, recall steps after a break, receive feedback within FEEDBACK_TIME, and can correct errors easily.

How test will be performed: Conduct usability sessions with participants performing all major tasks while timing actions, recording feedback response, and monitoring error recovery.

NFR-ST 4. **Alt Text Storage and Personalization Options**

Type: Non-Functional, Manual, Dynamic.
Covers: UHR-PIR1.
Initial State: Alt text generated for uploaded image
Input/Condition: User chooses to copy or download generated alt text
Output/Result: Alt text is successfully copied to clipboard or downloaded as .txt

How test will be performed: Tester generates alt text and selects each option, confirming that the system executes the chosen storage method correctly.

NFR-ST 5. **Accessibility for Screen Readers and Low-Vision Users**

Type: Non-Functional, Manual, Dynamic.
Covers: UHR-LR1, UHR-AR1, UHR-AR2.
Initial State: Interface accessible with popular screen readers
Input/Condition: Users with screen readers upload images and generate alt text.
Output/Result: Users successfully obtaines generated alt text within the time limit and can navigate the interface efficiently.

How test will be performed: Conduct sessions with low-vision participants using NVDA, JAWS, or VoiceOver and measure task completion time and success rates.

NFR-ST 6. **Performance – Alt Text Generation Time**

Type: Non-Functional, Automated, Dynamic.
Covers: PR-SL1, PR-SL2.
Initial State: System under typical load conditions
Input/Condition: Upload images of various sizes (small and large).
Output/Result: Generated alt text returned within specified thresholds for each image size; UI responds within T_UI_RESP.

How test will be performed: Automated scripts upload images and record generation time and UI response time; results plotted to verify performance meets criteria.

NFR-ST 7. **Safety and Timeout Handling**

Type: Non-Functional, Manual, Dynamic.
Covers: PR-SR-HA1, PR-SR-HA2, PR-SR-HA3.
Initial State: System is ready for alt text generation.
Input/Condition: Simulate long-running or stalled image processing.
Output/Result: User is notified of timeout, option to retry; incomplete data is deleted; messages do not reveal technical details.

How test will be performed: Tester simulates timeouts and verifies notifications, data cleanup, and absence of technical information in messages.

NFR-ST 8. **Alt Text Accuracy and Usability Evaluation**

Type: Non-Functional, Manual, Dynamic
Covers: PR-PAR1, PR-PAR2, PR-PAR3, CR-LR1
Initial State: Alt text generated for uploaded images.
Input/Condition: Users rate generated alt text for sufficiency, length, readability, and usability.
Output/Result: Ratings meet R_SUFFICIENCY, R_LENGTH, and R_USABILITY_MEDIAN thresholds.

How test will be performed: Conduct structured user evaluation using rating scales; calculate statistics to confirm compliance with quality thresholds.

NFR-ST 9. **Robustness to Invalid Inputs and Fault Recovery**

Type: Non-Functional, Manual/Automated, Dynamic
Covers: PR-RFT1, PR-RFT2
Initial State: System is running normally.
Input/Condition: Upload unsupported, corrupted files, or simulate backend process failures.
Output/Result: Clear error messages displayed; system recovers within T_RECOVERY.

How test will be performed: Tester uploads invalid files and observes error handling; automated test simulates isolated failures to confirm automatic recovery.

NFR-ST 10. **Concurrent Usage and Storage Capacity**

Type: Non-Functional, Automated, Dynamic
Covers: PR-CR1, PR-CR2
Initial State: System deployed in test environment.
Input/Condition: Simulate multiple simultaneous users and upload images.
Output/Result: Supports CAP_CONCURRENT users with response times $\leq$ CAPACITY_RTIME; storage handles CAP_STORAGE datasets.

How test will be performed: Load testing scripts simulate multiple concurrent requests and image uploads; performance and storage usage monitored.

NFR-ST 11. **System Extensibility and Maintainability**

Type: Non-Functional, Manual, Dynamic
Covers: PR-SER1, PR-LR1, PR-LR2, MS-MNT1, MS-MNT2, MS-MNT3, MS-AD1, MS-AD2, MS-AD3
Initial State: Existing modular system codebase deployed
Input/Condition: Apply updates to modules, configuration, or AI models.
Output/Result: System maintains functionality; new modules integrate without breaking existing components; changes tracked.

How test will be performed: Tester modifies components and configuration files, runs automated CI/CD tests, and verifies integration of new modules.

NFR-ST 12. **Security, Access, and Network Restrictions**

Type: Non-Functional, Manual/Automated, Dynamic
Covers: SR-AR1, SR-AR2, SR-IR1, SR-IR2, SR-PR1, SR-PR2, SR-AU1, SR-AU2, SR-IM1, SR-IM2
Initial State: System deployed with Single Sign-On (SSO) and HTTPS enabled.
Input/Condition: Attempt unauthorized access, upload images, and inspect logs.
Output/Result: Only authorized McMaster users gain access; encrypted communication enforced; uploaded images deleted; PII filtered; logs access restricted; unsupported files rejected; external networks blocked.

How test will be performed: Testers attempt invalid logins, inspect encrypted traffic, upload images and check deletion, validate moderation filters, and verify network restrictions and audit log access.

NFR-ST 13. **Cultural and Professional Content Compliance**

Type: Non-Functional, Manual, Dynamic

Covers: CR1, CR2, CR3
Initial State: Alt text is generated by the system.
Input/Condition: The generated alt text is reviewed.
Output/Result: Text is neutral, inclusive, contextually accurate, and professional.

How test will be performed: Tester inspects a variety of outputs for bias, unnecessary cultural references, and tone appropriateness.

NFR-ST 14. **Compliance and Regulatory Verification**

Type: Non-Functional, Manual, Dynamic
Covers: CR-LR1, CR-SCR1, CR-SCR2
Initial State: System generates alt text and manages uploaded image.
Input/Condition: Validate against AODA, WCAG 2.1, and institutional privacy policies.
Output/Result: Generated alt text meets accessibility standards; uploaded files handled per policy; documentation available for stakeholders.

How test will be performed: Run accessibility and privacy compliance tests; inspect system logs and documentation for adherence.

NFR-ST 15. **Environmental and Device Compatibility**

Type: Non-Functional, Manual, Dynamic
Covers: OER-EP1, OER-EP2, OER-WE1, OER-WE2
Initial State: System installed on multiple devices and OS platforms.
Input/Condition: System is accessed on varying devices and OS platforms.
Output/Result: System functions reliably across devices and platforms; maintains network connectivity.

How test will be performed: Testers access the system on Windows, Mac, and Linux with various browsers; verify full functionality.

NFR-ST 16. **Image Alt Text Accuracy Metrics**

Type: Non-Functional, Automated, Dynamic
Covers: PR-VAL1, PR-VAL2
Initial State: Alt text generation is operational.
Input/Condition: Process the images test set.
Output/Result: Accuracy metrics calculated; performance within target thresholds.

How test will be performed: Automated evaluation against ground truth alt text; calculate accuracy, precision, and recall.

NFR-ST 17. **Privacy of Uploaded Images**

Type: Safety-Critical, Manual and Automated, Dynamic
Covers: PR-SCR1
Initial State: User session active; image upload interface loaded.
Input/Condition: Upload test images without opting to save.
Output/Result: Uploaded images removed from temporary storage; no images stored in database.

How test will be performed: Upload images, end user session (logout or timeout), inspect temporary storage and databases to verify images are deleted and no personally identifiable data remains.

NFR-ST 18. **Offensive or Biased Alt Text Prevention**

Type: Safety-Critical, Automated, Dynamic
Covers: PR-SCR2
Initial State: Alt text generation is operational.
Input/Condition: Process diverse image test set, including sensitive content.
Output/Result: Generated alt text contains no offensive, biased, or harmful language.

How test will be performed: Generate alt text for each test image, pass outputs through moderation filters, confirm 0 percentage flagged content.

NFR-ST 19. **WCAG 2.1 Level AA Accessibility Compliance**

Type: Safety-Critical, Manual and Automated, Dynamic
Covers: PR-SCR3

Initial State: Interface is loaded and interactive.
Input/Condition: Navigate and interact with all interface elements.
Output/Result: Interface meets WCAG 2.1 Level AA criteria; no visual strain or accessibility barriers.

How test will be performed: Perform automated accessibility scans (e.g., axe, Lighthouse) and manual checks for color contrast, text resizing, keyboard navigation, and visual comfort to confirm compliance.

NFR-ST 20. **Screen Reader Interoperability**

Type: Functional, Manual and Automated, Dynamic
Covers: OER-IAS1
Initial State: Alt text generation is operational.
Input/Condition: Generate alt text and access it using major screen readers (NVDA, JAWS, VoiceOver.)
Output/Result: Alt text is correctly parsed and read aloud without formatting issues.

How test will be performed: Input generated alt text to NVDA, JAWS, and VoiceOver; verify correct pronunciation, formatting, and comprehension; record any errors or misread content.

NFR-ST 21. **Supported Image Format Processing**

Type: Functional, Automated, Dynamic
Covers: OER-IAS2
Initial State: Image upload interface is operational.
Input/Condition: Upload images in JPG, JPEG, and PNG formats.
Output/Result: System correctly processes images and generates accurate alt text.

How test will be performed: Prepare a set of test images in each supported format, upload to the system, generate alt text, and validate output accuracy against expected descriptions.

NFR-ST 22. **Automated Accessibility Validator Integration**

Type: Functional, Manual and Automated, Dynamic
Covers: OER-IAS3
Initial State: System is operational with validator interface enabled.
Input/Condition: Trigger accessibility validation using WAVE or Axe.
Output/Result: Validation reports are successfully generated and accessible through the interface.

How test will be performed: Run alt text through integrated validation tools; confirm report generation, correct display in UI, and accurate reflection of accessibility issues.

NFR-ST 23. **Web Tool Deployment**

Type: Productization, Manual and Automated, Dynamic
Covers: OER-PR1
Initial State: System is hosted and deployed on a test server.
Input/Condition: Access web tool from multiple institutional environments.
Output/Result: Tool is accessible, functional, and validated through institutional testing.

How test will be performed: Deploy the system on institutional server; verify accessibility, authentication, alt text generation, and performance; document compliance with institutional standards.

NFR-ST 24. **Core Feature Verification Before Release**

Type: Release, Manual and Automated, Dynamic
Covers: OER-RL1
Initial State: System is fully implemented in test environment.
Input/Condition: Execute image analysis, text generation, and accessibility validation modules.
Output/Result: All core features function correctly; verification and validation documentation confirms compliance.

How test will be performed: Run full test suite for each core feature; document results, record any failures, and confirm all functional requirements are met prior to release.

NFR-ST 25. **Release Readiness for Capstone Demonstration**

Type: Release, Manual and Automated, Dynamic
Covers: OER-RL2
Initial State: System is deployed to staging environment.
Input/Condition: Conduct a full system walkthrough aligned with March 2026 Capstone schedule.
Output/Result: System is fully functional, accessible, and ready for demonstration.

How test will be performed: Perform end-to-end functional testing, accessibility verification, and deployment validation; ensure all components operate as expected and system is stable for final presentation.

NFR-ST 26. **Performance and Metrics Logging**

Type: Maintenance Support, Automated, Dynamic
Covers: MS-SUP1
Initial State: System operational with monitoring/logging enabled.
Input/Condition: Execute standard workflows including image uploads, alt text generation, and API interactions.
Output/Result: Metrics for API calls, latency, error rates, and model confidence scores are logged and accessible.

How test will be performed: Perform typical user operations; monitor logs and dashboards; export reports to verify all key metrics are accurately recorded, securely stored, and available for analysis.

4.3 Traceability Between Test Cases and Requirements

Table 3 highlights the traceability between each test case listed above and the functional and non-functional requirements in our SRS document.

Table 3: Connections between *Reading4All* system tests and SRS requirements

Test ID	**Requirement ID(as per SRS)**

FR-ST 1	FR-1
FR-ST 2	FR-2
FR-ST 3	FR-3
FR-ST 4	FR-4
FR-ST 5	FR-5
FR-ST 6	FR-6
NFR-ST 1	LFR-AR1, LFR-AR2, LFR-AR3, LFR-AR4.
NFR-ST 2	LFR-SR1, LFR-SR2, LFR-SR3
NFR-ST 3	UHR-EUR1, UHR-EUR2, UHR-EUR3, UHR-EUR4
NFR-ST 4	UHR-PIR1
NFR-ST 5	UHR-LR1, UHR-AR1, UHR-AR2
NFR-ST 6	PR-SL1, PR-SL2
NFR-ST 7	PR-SR-HA1, PR-SR-HA2, PR-SR-HA3
NFR-ST 8	PR-PAR1, PR-PAR2, PR-PAR3, CR-LR1
NFR-ST 9	PR-RFT1, PR-RFT2
NFR-ST 10	PR-CR1, PR-CR2
NFR-ST 11	PR-SER1, PR-LR1, PR-LR2, MS-MNT1, MS-MNT2,MS-MNT3, MS-AD1, MS-AD2, MS-AD3
NFR-ST 12	SR-AR1, SR-AR2, SR-IR1, SR-IR2, SR-PR1, SR-PR2, SR-AU1, SR-AU2, SR-IM1, SR-IM2
NFR-ST 13	CR1, CR2, CR3
NFR-ST 14	CR-LR1, CR-SCR1, CR-SCR2
NFR-ST 15	OER-EP1, OER-EP2, OER-WE1, OER-WE2
NFR-ST 16	PR-VAL1, PR-VAL2
NFR-ST 17	PR-SCR1
NFR-ST 18	PR-SCR2
NFR-ST 19	PR-SCR3
NFR-ST 20	OER-IAS1
NFR-ST 21	OER-IAS2
NFR-ST 22	OER-IAS3
NFR-ST 23	OER-PR1
NFR-ST 24	OER-RL1

NFR-ST 25	OER-RL2
NFR-ST 26	MS-SUP1

5 Unit Test Description

This section is not currently applicable and will be completed at a later time.

5.1 Unit Testing Scope

5.2 Tests for Functional Requirements

5.3 Tests for Non-Functional Requirements

5.4 Traceability Between Test Cases and Modules

References

Development plan for bridging gaps: Ai for diagram accessibility, 2025. URL https://github.com/4G06-CAPSTONE-2025/Reading4All/blob/main/docs/DevelopmentPlan/DevelopmentPlan.pdf.

Module guide for bridging gaps: Ai for diagram accessibility, 2025. URL https://github.com/4G06-CAPSTONE-2025/Reading4All/blob/main/docs/Design/SoftArchitecture/MG.pdf.

Module interface specification for bridging gaps: Ai for diagram accessibility, 2025. URL https://github.com/4G06-CAPSTONE-2025/Reading4All/blob/main/docs/Design/SoftDetailedDes/MIS.pdf.

Software requirements specification for bridging gaps: Ai for diagram accessibility, 2025. URL https://github.com/4G06-CAPSTONE-2025/Reading4All/blob/main/docs/SRS-Volere/SRS.pdf.

6 Appendix

6.1 Evaluation Metrics Summary

The following table summarizes the evaluation metrics that will be used to assess the quality and effectiveness of the alternative text generated by th *Reading4All* system. Each metric includes its scale type, acceptable range, and a brief description of its purpose.

Table 4: Evaluation Metrics Summary

Metric Name	**Scale Type**	**Acceptable Range**	**Summary Description**
Sufficiency of Description	Categorical (1–3)	≥ 3 (Sufficient)	Does the alt text convey enough information to achieve the intended objective?
Length Appropriateness	Categorical (1–3)	≥ 3 (Proper Length)	Is the alt text concise yet complete (not too short or overly verbose)?
Accessibility / Usability	Numerical (0–3)	≥ 2 (Acceptable)	Assistive-technology compatibility and clarity; aligns with WCAG 2.1 Level AA use.
Learning Impact	Numerical (0–3)	≥ 2 (Positive)	Does the alt text support or enhance user understanding in learning contexts?
Qualitative Feedback Notes	Textual	N/A	Free-form comments on clarity, tone, and suggested improvements.

6.2 SRS Team and Peer Review Checklist

Stakeholders and Users:

- All relevant stakeholders are listed and explained
- Personas clearly explain the stakeholders pain points, needs and their relationship to system being designed.

Mandated Constraints:

- All solution constraints are clearly explained, attainable, and measurable.

Functional and Non-Functional Requirements:

- Each requirement has a unique identifier.
- All core system features have corresponding functional requirement.
- Each functional requirement is clearly defined and measurable.
- Numerical constraints (ex, number of steps or completion time) are realistic and achievable.
- All the functional requirement are unique, and do not conflict with one another
- All the functional requirements can be traced to a business and product use case.
- Each non-functional requirements is clearly defined and measurable.
- All usability and accessibility needs are addressed by a requirement.
- Performance related numerical constraints are achievable.
- All the Non-functional requirements are unique, and do not conflict with one another.

6.3 Verification and Validation Plan Verification Checklist

General Document Criteria:

- Mission critical qualities are thoroughly discussed and referenced throughout the plan.

- Relevant documents such as SRS and HA are referenced and connected to plan.

SRS Verification:

- Verification process is thorough and includes key stakeholders.
- Provided checklist can guide review process and bring attention to important parts of document.
- Describes a plan for documenting and implementing feedback.
- Criteria for evaluating SRS quality is defined.
- The data collected as evidence for VnV is clear.

Design Verification:

- Design review methods are specified, explained and justified.
- The process for documenting and resolving design review feedback is described.
- The data collected as evidence for VnV is clear.
- Automated testing and verification tools are specified.

VnV Plan Verification:

- Verification methods are specified, explained and justified.
- Mutation testing will be used to verify the effectiveness of unit tests.
- The data collected as evidence for VnV is clear.

System Tests for Requirements:

- All test cases are detailed and specify input data.
- Survey questions are outlined for usability testing.
- System tests connect and cover all system requirements.
- Traceability between test cases and requirement is clear and documented.

6.4 Design Documents Checklist

Module Guide:

- Each identified module follows the “one module, one secret” rule.
- “Uses” relation forms a clear hierarchy and represents dependency.
- Secrets are expressed as nouns or concepts, not actions.
- Traceability matrix shows that every requirement is satisfied by at least one module.
- Traceability matrix shows that every module satisfies at least one requirement.
- Traceability matrix shows that every likely change maps to a module.
- Behaviour-Hiding modules trace back to requirements.
- Software-Decision Hiding modules introduce necessary design concepts.
- Each Software-Decision Hiding module supports at least one Behaviour-Hiding module.
- Anticipated changes include all likely changes from SRS.

Module Interface Specification:

- Data-only modules are modelled as exported types.
- Modules with state and behaviour are correctly defined as ADTs.
- Single-instance modules are correctly identified as Abstract Objects.
- Behaviour-only modules are defined as Library Modules (no state).
- Generic modules use the Generic keyword appropriately.
- Abstract Objects include proper initialization methods and assumptions.
- Exported constants are literal, compile-time values.

- Modified Hoffmann and Strooper notation (or equivalent) is used consistently.
- All local functions are used somewhere in the module specification.
- Each access program has a clear purpose (output or state change).
- State transitions clearly indicate what changes and where.
- State invariants hold before and after access program execution.
- Specification is consistent, essential, general, and independent of implementation details.
- Every module in the Module Guide (MG) appears in the MIS.

6.5 Symbolic Parameters

Table 5 defines constants and their values for requirements and test cases used in the *Reading4All* system.

Table 5: Symbolic Constants used in the *Reading4All* System

Name	**Value**
T_ALT_GEN_SMALL	3 seconds(s)
T_ALT_GEN_LARGE	8 s
T_UI_RESP	300 miliseconds (ms)
R_SUFFICIENCY	85%
R_LENGTH	90%
R_USABILITY_MEDIAN	3 (rating)
R_USABILITY_MIN	2 (rating)
T_ERROR_HANDLE	2 s
T_RECOVERY	5 s
CAP_CONCURRENT	2 requests
CAP_STORAGE	500 images/day
MAINT_TIME	2 person-days/quarter
COMPAT_VERSIONS	2 releases
IMG_SIZE_BIG	10 MEGABYTES (MB)
IMG_SIZE_SMALL	2 MEGABYTES
TLS_VERSION	1.2
FILE_DELETE_TIME	60 s
FILE_TYPES	.png, .jpg, .jpeg, .svg, .webp
NETWORK_SOURCE_POLICY	McMaster SSO tokens or IP ranges only
TEAM_SIZE	5 students
HOURS_RESEARCH	40 hours
HOURS_BACKEND	120 hours
HOURS_FRONTEND	80 hours
HOURS_TESTING	60 hours
HOURS_DOCS	30 hours
HOURS_TOTAL	330 hours
HOURS_PROJECT	1,320 person-hours
COST_PER_HOUR	$20/hour(CAD)
COST_TOTAL	$26,400 CAD
COST_ACTUAL	$0 CAD

COST_INCENTIVE_MIN	$100 CAD
COST_INCENTIVE_MAX	$150 CAD
MAX_ZOOM_PERCENTAGE	200%
MIN_CONTRAST_RATIO	4.5:1
MAX_UPLOAD_STEPS	5 steps
MAX_MINUTES	5 minutes
USERS_SUCCESS_PERCENT	80%
MAX_ERROR_RECOVER	2 seconds
LEARNING_PERCENT	90%
MAX_LEARNING_MINUTES	5 minutes
MIN_COMPENSATION_DOLLARS	$100 CAD
MAX_COMPENSATION_DOLLARS	$150 CAD
LATEST_RELEASES_NUM	last 3 versions
MOST_COMMON_SR	top 3 screen readers
SFWR_RELEASES	previous 2 releases
FEEDBACK_TIME	1 second
CAPACITY_RTIME	10 seconds
SESSION_TIMEOUT	2 hours
SESSION_HISTORY_MAX	10 images

6.6 Usability Survey Questions

UA-Q 1. *What operating system and its version are you using? (i.e. Windows, MacOS, etc.)*

Purpose: This is to collect information on the user to assess the compatibility across different platforms and whether accessibility or usability issues are system-dependent.

UA-Q 2. *What browser are you using? (i.e. Google Chrome, Mozilla Firefox, etc.)*

Purpose: This is to collect information on the user to assess the compatibility across different browsers and whether accessibility or usability issues are dependent on the browsers.

UA-Q 3. *What assistive technologies or tools are you using (i.e., screen readers, magnifiers, voice control), and what specific model or version, if applicable?*

Purpose: This collects information on the types and versions of assistive technologies used by participants to understand how different tools interact with the web tool and to identify any issues specific to certain technologies.

UA-Q 4. *Was the length of the generated alternative text description too short, sufficient, or too long?*

Purpose: This is to gain feedback on whether the length of the text description is sufficient enough for the user to gain understanding on the learning objective of the uploaded image.

UA-Q 5. *Did the generated alternative text contain too much irrelevant information? If so, how did this affect your ability to learn?*

Purpose: This is to gain feedback on whether the alternative text contained irrelevant information that deviated from the main learning objective of the uploaded image. It is also to understand what difficulties the stakeholder experiences when there is too much irrelevant information.

UA-Q 6. *Was the generated alternative text contain over-detailed? If so, how did this affect your ability to learn?*

Purpose: This is to determine whether the generated alternative text includes excessive detail that may hinder user comprehension or distract from the key information needed for effective learning.

UA-Q 7. *On a scale of 0-4, is the generated alternative text presented in an accessible way? (0 = not accessible at all, 4 = very accessible)*

Purpose: This is to ensure that the AI-generated alternative text is presented in a manner that aligns with accessibility standards and can be easily perceived, understood, and utilized by users relying on assistive technologies such as screen readers.

UA-Q 8. *On a scale of 0–4, how easy was it for you to use and understand the generated alternative text? (0 = not easy at all, 4 = very easy)*

Purpose: This is to assess whether the generated alternative text is presented in a way that supports ease of use and whether it allows users to easily interact with the interface and understand the content with minimal confusion or effort.

UA-Q 9. *What is your primary purpose for using this web tool, and how do you typically use it during your tasks or learning activities?*

Purpose: This is to understand the user's main goals and usage patterns when interacting with the web tool and to identify any areas for improvement in how the web tool's features can better support their needs.

UA-Q 10. *What changes or improvements would you suggest to make this web tool more accessible and enhance your learning experience, if any?*

Purpose: This question aims to collect additional user feedback on accessibility and usability improvements that may not have been addressed in the previous questions of this survey.

6.7 Glossary of All Terms

Accuracy The degree to which generated descriptions capture the image's content correctly.

AI (artificial intelligence) Techniques that enable computers to perform tasks that normally require human intelligence.

Alt Text (Alternative Text) Textual description of non-text content such as images that allow accessibility tools such as screen readers to convey the content.

AODA (Accessibility for Ontarians with Disabilities Act) Ontario law aimed at improving accessibility for people with disabilities by removing and preventing barriers when designing.

API (Application Programming Interface) Rules and protocols that allows different software programs to communicate with each other.

Backend Server components handling processes of an application that users don't see.

Benchmarking The comparing of performance or quality of one's system against known systems or datasets.

Contrast Ratio Luminance difference between text and background required by WCAG 2.1.

Dataset Bias Systematic skew in training data that can harm fairness or accuracy of the model.

Edge Case Uncommon input or scenario that the system must handle safely.

FIPPA (Freedom of Information and Protection of Privacy Act) Ontario privacy law affecting the university data in the Authentication process.

Frontend User interface in the browser that handles input, feedback, and accessibility features.

Git/Github Version control and collaboration platform.

HTTP/HTTPS Web protocols in which HTTPS adds a transport layer security encryption for integrity and privacy.

Issue (Github) Tracked unit of task, bug, or feature with discussion and linkage to commits in Github.

JAWS (Job Access with Speech) A screen reader software available on Windows.

JSON (JavaScript Object Notation) / YAML (Yet Another Markup Language) Human-readable data formats used for configs and API payloads.

Latency Time from user action such as uploading an image to a system response or alt text generation

Low Vision Reduced level of vision that interferes with daily activities and is to be considered in designing the user interface and testing.

Manual Accessibility Testing Human review or testing of user interface and alt text.

Modularity Separating user interface, vision, language, and validation for maintainability.

NVDA (NonVisual Desktop Access) A free screen reader available on Windows.

OCR (Optical Character Recognition) Extracts embedded text in images or diagrams.

PII (Personally Identifiable Information) Data that identifies a person and must not appear in outputs or logs for security.

Screen Magnifier Assistive technology to enlarge screen content.

Screen Reader Assistive technology that reads text aloud.

Session A session in the *Reading4All* is defined as the time interval beginning when a user successfully logs in and is authenticated and ending when the user explicitly logs out or the session expires due to inactivity or timeout. Sessions have a maximum duration defined by `SESSION_TIMEOUT`. Session data which includes the users uploaded images and any generated, as well as modified alt-text is stored in the system only for the duration of this session.

Session History Record of user uploads and their associated alt text during the current session. The session history will initially store up to `SESSION_HISTORY_MAX` images and their associated alt text. Once this limit is exceeded, older records will be discarded to ensure that the *Reading4All* does not experience any performance limitations.

Stakeholder Anyone affected by or influencing the system.

Technical Diagram An informational visual used in post-secondary course materials.

TLS (Transport Layer Security) Protocol providing encryption and integrity.

WCAG 2.1 (Web Content Accessibility Guidelines) International standard for accessible web content.

WCAG Levels (A/AA/AAA) Different conformance tiers to WCAG 2.1 where Level AA is the target for the project.

Appendix — Reflection

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:

Team Reflection

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, Valgrind etc. You should look to identify at least one item for each team member.**

    - Acquiring knowledge on the different types of testing (i.e. mutation testing, load testing) and how each are applicable to different aspects of our system.
    - The knowledge on ways to conduct specific tests. For example, how to set up load tests and appropriately recognizing the differences between the procedures of conducting the tests.
    - Learning more about using coverage.py will be important in ensuring we can understand the reports in order to add more tests if needed.
    - The team requires more knowledge on integration of testing with our models. This includes building test suites that verify our

model's outputs.

- Knowledge on writing efficient unit tests that are relevant and improves code coverage. Additionally, gaining more insight on using PyTest will help us become more efficient in testing our code.

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?**

   - Different types of testing (Casey): First approach is to research the different types of testing and understanding how each type is used in specific cases. Additionally, reading over the defined types of testing our project is using and looking ahead to how the tests will be implemented will help to understand testing better.
   - Procedures of testing (Dhruv): First approach is going through 3S03 course notes to understand testing. Additionally, looking at tutorials or python notes on how these tests are typically conducted.
   - Coverage.py (Moly): First approach is to read and watch tutorials on how to use coverage.py and its different features. The second approach is to create a small project that will run coverage.py and use the results to see where unit tests are missing.
   - Integration of Testing (Nawaal): First approach is to consult 3S03 notes and figure out what the largest type of testing is needed for the project. Additionally, consultion Dr. Smith and Dr. David for any additional information if needed.
   - Unit Tests and PyTest (Fiza): First approach is to look at PyTest documentation and gain more insight on the library. Also, looking at sample projects that uses PyTest and going through 3S03 course notes will help in learning more about unit testing.

Fiza Sehar - Reflection

1. **What went well while writing this deliverable?**

We effectively organized the verification and validation framework for *Reading4All* by referencing both functional and non-functional requirements from the SRS. We collaborated to design clear, structured unit tests, ensuring traceability and consistency across modules.

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

   This section was not required, but we completed it in detail before realizing that, which limited our focus on other sections. This caused minor conflicts regarding time and task distribution, which we resolved through open communication and by redistributing responsibilities for future deliverables. We decided to created internal rubrics and clearer priorities to stay organized and avoid similar issues moving forward.

Moly Mikhail - Reflection

1. **What went well while writing this deliverable?**
   I believe many aspects went well while writing this deliverable. Firstly, starting our work prior to the interim TA presentation was really helpful. This provided an opportunity to look ahead at the sections and begin tackling them. While doing so, many questions and areas of confusion arose, so having our upcoming meeting with our TA was really helpful. Ultimately, this meeting allowed us to clear up any confusion about sections with the document, giving the group much more confidence. Another thing that went well throughout this deliverable is having team check-ins and reviews. This allowed all team members to gain insight into the other parts of the document and ensure we are aligned with the content.

2. **What pain points did you experience during this deliverable, and how did you resolve them?**
   One pain point I experienced writing this deliverable was fully understanding the difference between verification and validation. This made it challenging to differentiate between the parts in section 3. I resolved this pain point by reviewing lecture content, researching more about validation and verification techniques and finally clarifying with our TA during the interim presentation. Having a good understanding about the difference between validation and verification was essential in completing this deliverable. Once I completed my assigned sections, I

reflected back on if I had completed the appropriate activities, ensuring it correctly aligned with what the section needed.

Casey Francine Bulaclac - Reflection

1. **What went well while writing this deliverable?**
   What went well during this deliverable was gaining a better understanding of the difference between verification and validation and how each plays a role in making sure our system works as intended. It also went smoothly linking our VnV activities back to the requirements in the SRS, which helped us see clearly how each test connects to what the system is supposed to do. Overall, this deliverable helped our team prepare for how to properly test the product we will be developing, and also helped us gain insight on how verification and validation ensure that our system meets both its functional requirements and user needs.

2. **What pain points did you experience during this deliverable, and how did you resolve them?** A pain point while writing the VnV plan was trying to differentiate between the different sections of Section 3. For example, at the beginning stages, it was hard to understand the difference between Implementation Verification and Software Validation. To resolve this, I made sure to ask for clarification from our TA to ensure that I understood each section properly and also used sample VnV documents from teams in previous years as guidance when writing these sections. Another pain point experienced during this deliverable was a miscommunication regarding the division of work, specifically, assigning Section 5 as a task during the initial stage when it was not yet required. To resolve this, our team held a meeting to address the communication gap and established a goal to ensure that all members clearly understand the scope and requirements of each section before starting future deliverables.

Dhruv Sardana - Reflection

1. **What went well while writing this deliverable?**
   While writing this deliverable, it helped me to gauge a good understanding of the system requirements document (SRS) and how each requirement can be tested through various test cases. This deliverable also helped me understand the importance of traceability between requirements and test cases, ensuring that each requirement is adequately

covered by at least one test case. Besides that I was responsibel for writing the Unit Tests section which we later got to know wasnt required for this deliverable but it still was though provoking and helped me understand different modules of the system in depth.

2. **What pain points did you experience during this deliverable, and how did you resolve them?** A pain point I experienced during this deliverable was the initial confusion regarding the scope of the Unit Test Description section. I initially thought it was required for this deliverable, which led to some misallocation of time and effort. Due to this, it caused a few minor conflicts which helped us to shape our focus better for future deliverables. Later on, I was assigned to restructure the POC plan and review the tracebility of System Tests to SRS document which helped me understand the requirements better.

Nawaal Fatima - Reflection

1. **What went well while writing this deliverable?** One thing that went really well was hoping to start our work early. We met early, which wasn't the case and getting a head start allowed us to become familiar with the sections and plan how to tackle them efficiently. Another positive aspect was improving our familiarity with Git. Working collaboratively on the repository helped us manage changes better and keep track of contributions, which made the teamwork smoother and more organized.

2. **What pain points did you experience during this deliverable, and how did you resolve them?** One pain point I experienced was some miscommunication and expectation mismatches while working as a team. Specifically the instructions for unit tests not being communicated until a couple of days before the deadline left a bad taste in my mouth. To resolve this, we had a huge team meeting, several check-ins and clarified responsibilities, making sure everyone was aligned on expectations. This improved collaboration and ensured that the final deliverable met the group's standards.