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

October 27, 2025

Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

[The intention of the VnV plan is to increase confidence in the software. However, this does not mean listing every verification and validation technique that has ever been devised. The VnV plan should also be a **feasible** plan. Execution of the plan should be possible with the time and team available. If the full plan cannot be completed during the time available, it can either be modified to "fake it", or a better solution is to add a section describing what work has been completed and what work is still planned for the future. —SS]

[The VnV plan is typically started after the requirements stage, but before the design stage. This means that the sections related to unit testing cannot initially be completed. The sections will be filled in after the design stage is complete. the final version of the VnV plan should have all sections filled in. —SS]

Contents

List of Tables

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

List of Figures

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

1 Symbols, Abbreviations, and Acronyms

Symbolic Constants

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
COST_TOTAL	$26,400 CAD
COST_ACTUAL	$0 CAD
COST_INCENTIVE_MIN	$100 CAD

COST_INCENTIVE_MAX	$150 CAD
MAX_ZOOM_PERCENTAGE	200% *Table continues on next page*
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	3
MOST_COMMON_SR	3
SFWR_RELEASES	2

This document ... [provide an introductory blurb and roadmap of the Verification and Validation plan —SS]

2 General Information

2.1 Summary

[Say what software is being tested. Give its name and a brief overview of its general functions. —SS]

The software being tested is called Reading4All. This software will utilize artificial intelligence (AI)/ 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 automatically produce corresponding alternative text, that meet the outlined criteria. 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

[State what is intended to be accomplished. The objective will be around the qualities that are most important for your project. You might have something like: "build confidence in the software correctness," "demonstrate adequate usability." etc. You won't list all of the qualities, just those that are most important. —SS]

[You should also list the objectives that are out of scope. You don't have the resources to do everything, so what will you be leaving out. For instance, if you are not going to verify the quality of usability, state this. It is also worthwhile to justify why the objectives are left out. —SS]

[The objectives are important because they highlight that you are aware of limitations in your resources for verification and validation. You can't do everything, so what are you going to prioritize? As an example, if your system depends on an external library, you can explicitly state that you will

assume that external library has already been verified by its implementation team. —SS]

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 the 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 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 docu-

ment 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

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

3.1 Verification and Validation Team

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

Table 2: Verification and Validation Responsibility Breakdown

Name	Focus Area	Responsibility
Moly Mikhail and Fiza Sehar	Functional Requirements Tester	Completes manuel testing on the software to ensure that 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	Coordinates the user testing sessions, and oversees the planning and execution of the sessions.
Casey Francine Bulaclac	Usability Requirements Tester	Completes manuel testing to ensure all the defined usability requirements are met. Also references WCAG 2.1 crtieria is met, prior to automated testing.
Dhruv Sardana	Look and Feel Requirements	Evaluates the systems design and interface to ensure the outlined requreiments are met. Also references WCAG 2.1 criteria is met, prior to automated testing.
Ms. Jingchuan Sui (Supervisor)	Overall Usability and Quality of Alternative Text Reviewer	Reviews and provides feedback to team on the overall usability of the system and the quality of the generated alternative text. Also helps team complete automating evaluation of the system against WCAG 2.1 criteria.

3.2 SRS Verification

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

[If you have a supervisor for the project, you shouldn't just say they will read over the SRS. You should explain your structured approach to the review. Will you have a meeting? What will you present? What questions will you ask? Will you give them instructions for a task-based inspection? Will you use your issue tracker? —SS]

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.

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 6.1.

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 6.1, 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.

[Maybe create an SRS checklist? —SS]

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 6.3. 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 is an artifact that should also be verified. Techniques for this include review and mutation testing. —SS]

[The review will include reviews by your classmates —SS]

[Create a checklists? —SS]

The Verification and Validation plan will be verified to ensure its completeness, and alignment with previous documents such as our SRS, HA and development plan. This will be completed through a team and peer review.

Team Review

The team will perform a review to ensure that the unit testing described within the document effectively tests the system and achieves the defined objectives. This review will involve the team collaboratively going through the document, identifying areas for improvement and making the necessary changes before completing further validation activities.

Peer Review

A peer review will be completed by Team 10 (One of a Kind) to provide feedback on our V&V plan. They will review the document to ensure that the sections are consistent and aligned, as well as to identify any missing unit tests.

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.

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 pro-

vided in Appendix 6.2.

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

[You should at least point to the tests listed in this document and the unit testing plan. —SS]

[In this section you would also give any details of any plans for static verification of the implementation. Potential techniques include code walkthroughs, code inspection, static analyzers, etc. —SS]

[The final class presentation in CAS 741 could be used as a code walkthrough. There is also a possibility of using the final presentation (in CAS741) for a partial usability survey. —SS]

The implementation verification for Reading4All will utilize the unit tests defined in section 5, static analyzers and code inspections.

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 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.

- **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.

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 6.5 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 Team 22'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.(I need to add in more summary text here)

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 logged in and has uploaded at least one image with generated alt text.
Input: Upload multiple images sequentially within the same session, then navigate through the session interface.
Output: All previously uploaded images and their corresponding alt texts remain visible and accessible until the user logs out or the session ends.
Case Derivation: According to **FR 5** criterion, all uploaded images and their corresponding alternative texts should remain visible within the same session. Therefore, the expected outcome is that users can access and review all 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.

4.2 Tests for Nonfunctional Requirements

Need a small summary blurb here pls. - NF

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 efficiently, recall steps after a break, receive feedback within 1 second, 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 generate alt text within the time limit and can navigate 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. **Information Display Clarity**

Type: Non-Functional, Manual, Dynamic
Covers: UHR-LR2
Initial State: Interface displays system messages and information
Input/Condition: User navigates through tasks without developer guidance
Output/Result: Only essential information is displayed; technical details are hidden

How test will be performed: Tester inspects interface during task completion to confirm that technical or unnecessary details are not visible to the user.

NFR-ST 7. **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 8. **Safety and Timeout Handling**

Type: Non-Functional, Manual, Dynamic
Covers: PR-SR-HA1, PR-SR-HA2, PR-SR-HA3
Initial State: System ready for alt text generation
Input/Condition: Simulate long-running or stalled image processing
Output/Result: User 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 9. **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 10. **Robustness to Invalid Inputs and Fault Recovery**

Type: Non-Functional, Manual/Automated, Dynamic
Covers: PR-RFT1, PR-RFT2
Initial State: System 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 11. **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 datasets
Output/Result: Supports CAP_CONCURRENT users with response times $\leq$ 10s; storage handles CAP_STORAGE datasets

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

NFR-ST 12. **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 13. **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 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 14. **Cultural and Professional Content Compliance**

Type: Non-Functional, Manual, Dynamic
Covers: CR1, CR2, CR3
Initial State: Alt text generated by the system
Input/Condition: Review generated alt text samples
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 15. **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 data
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 16. **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: Access system in standard classroom, office, or cloud environments
Output/Result: System functions reliably across devices, platforms, and typical indoor conditions; 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 17. **Documentation and Logging Availability**

Type: Non-Functional, Manual, Dynamic
Covers: MS-LOG1, MS-LOG2, MS-LOG3
Initial State: System operational with logging enabled
Input/Condition: Generate alt text and simulate user actions
Output/Result: Comprehensive logs maintained; documentation available for developers and users; logs do not contain PII

How test will be performed: Tester performs sample actions; inspects logs and documentation for completeness and privacy compliance.

NFR-ST 18. **Error Handling and Feedback Clarity**

Type: Non-Functional, Manual, Dynamic
Covers: PR-RFT3, PR-RFT4
Initial State: System running normally
Input/Condition: Trigger recoverable and non-recoverable errors
Output/Result: Clear, concise, and non-technical error messages displayed; user able to recover or retry

How test will be performed: Tester induces known errors (e.g., network failure, invalid file upload) and verifies displayed messages and recovery options.

NFR-ST 19. **Data Retention and Auto-Deletion Compliance**

Type: Non-Functional, Manual, Dynamic
Covers: SR-DTR1, SR-DTR2
Initial State: User data uploaded and processed
Input/Condition: System retains data past defined retention period
Output/Result: Data automatically deleted per policy; no unauthorized access possible

How test will be performed: Tester checks timestamped data and verifies auto-deletion after expiration; attempts access post-deletion to confirm security.

NFR-ST 20. **System Logging and Audit Trail**

Type: Non-Functional, Manual/Automated, Dynamic
Covers: MS-LOG4, MS-LOG5
Initial State: System operational with logging enabled
Input/Condition: Perform multiple system operations and user logins
Output/Result: All critical actions logged securely; logs immutable and auditable

How test will be performed: Tester executes operations and reviews logs for completeness, accuracy, and security compliance.

NFR-ST 21. **System Recovery and Failover**

Type: Non-Functional, Manual/Automated, Dynamic
Covers: PR-RFT5, PR-RFT6
Initial State: System operating under normal load
Input/Condition: Simulate server crash, database failure, or network outage
Output/Result: System recovers automatically; user experience minimally impacted; data integrity maintained

How test will be performed: Tester simulates failures and verifies recovery procedures, system availability, and data integrity.

NFR-ST 22. **AI Model Update and Version Control**

Type: Non-Functional, Manual, Dynamic

Covers: PR-SER2, MS-MNT4
Initial State: AI model deployed with version control
Input/Condition: Apply new model updates
Output/Result: New model integrates without breaking functionality; rollback possible if necessary

How test will be performed: Tester deploys updates in staging environment and verifies system functions; tests rollback procedure.

NFR-ST 23. **Backup and Restore Functionality**

Type: Non-Functional, Manual, Dynamic
Covers: PR-RST1, PR-RST2
Initial State: System operational with recent backups
Input/Condition: Restore system from backup after simulated failure
Output/Result: System restores to prior state; no data loss beyond last backup

How test will be performed: Tester triggers backup restoration and verifies system state and data consistency.

NFR-ST 24. **Load Handling and Stress Testing**

Type: Non-Functional, Automated, Dynamic
Covers: PR-CR3, PR-CR4
Initial State: System deployed with baseline load
Input/Condition: Simulate increasing concurrent users and transactions beyond expected peak
Output/Result: System maintains acceptable response times and throughput; no critical failures

How test will be performed: Automated scripts generate load; monitor CPU, memory, and response times; confirm thresholds not exceeded.

NFR-ST 25. **Cross-Browser Compatibility**

Type: Non-Functional, Manual, Dynamic
Covers: OER-BC1, OER-BC2
Initial State: System accessible from modern browsers

Input/Condition: Access system via Chrome, Edge, Firefox, and Safari
Output/Result: Interface renders correctly; functionality consistent across browsers

How test will be performed: Tester manually accesses system from each browser and verifies UI, functionality, and performance.

NFR-ST 26. **Mobile Responsiveness**

Type: Non-Functional, Manual, Dynamic
Covers: OER-MR1, OER-MR2
Initial State: System deployed in browser and mobile view
Input/Condition: Access system from smartphones and tablets of different resolutions
Output/Result: UI scales correctly; touch interactions functional; alt text generation accessible

How test will be performed: Tester uses multiple devices and orientations, verifying layout, functionality, and accessibility.

NFR-ST 27. **Data Encryption in Transit and Storage**

Type: Non-Functional, Automated, Dynamic
Covers: SR-PR3, SR-PR4
Initial State: System with HTTPS and encrypted database enabled
Input/Condition: Upload images and generate alt text
Output/Result: All data encrypted during transit and storage; unauthorized decryption prevented

How test will be performed: Automated scripts intercept and attempt decryption; confirm encryption standards met.

NFR-ST 28. **Accessibility Compliance Documentation**

Type: Non-Functional, Manual, Static
Covers: CR-LR2, CR-LR3
Initial State: Documentation available in project repository
Input/Condition: Review compliance documentation for WCAG 2.1 and AODA

Output/Result: Documentation fully describes accessibility features, test results, and compliance levels

How test will be performed: Tester reads documentation and confirms all standards, features, and results documented accurately.

NFR-ST 29. **Scalability of Storage and Processing**

Type: Non-Functional, Automated, Dynamic
Covers: PR-CR5, PR-CR6
Initial State: System running with normal load
Input/Condition: Upload datasets with sizes increasing up to 10x expected usage
Output/Result: System handles scaling without degradation; storage and processing remain within thresholds

How test will be performed: Automated scripts simulate large datasets; monitor CPU, memory, and response times; confirm thresholds not exceeded.

NFR-ST 30. **Session Management and Timeout Enforcement**

Type: Non-Functional, Automated, Dynamic
Covers: SR-AR3, SR-AR4
Initial State: User logged in
Input/Condition: Remain idle for session timeout period
Output/Result: User automatically logged out; session data cleared

How test will be performed: Tester leaves session idle and verifies automatic logout and data clearance.

NFR-ST 31. **Internationalization and Localization**

Type: Non-Functional, Manual, Dynamic
Covers: OER-LC1, OER-LC2
Initial State: Interface configured with language options
Input/Condition: Switch language settings to French, Spanish, or other supported languages
Output/Result: UI, messages, and alt text generation output properly localized

How test will be performed: Tester changes language settings and inspects all interface components and generated alt text.

NFR-ST 32. **Logging Level and Error Traceability**

Type: Non-Functional, Manual, Dynamic
Covers: MS-LOG6, MS-LOG7
Initial State: System logging configured
Input/Condition: Trigger minor, major, and critical errors
Output/Result: Logs record errors with severity and timestamp; traceability from log to incident possible

How test will be performed: Tester induces errors and reviews logs for completeness, severity, and traceability.

NFR-ST 33. **Maintainability – Code Documentation**

Type: Non-Functional, Manual, Static
Covers: MS-MNT5, MS-MNT6
Initial State: System code deployed
Input/Condition: Review code comments, README, and module documentation
Output/Result: Documentation sufficient for new developer onboarding; functions and classes clearly described

How test will be performed: Tester reviews repository, checks code comments and documentation quality, and attempts to understand functionality.

NFR-ST 34. **Reliability and Uptime Metrics**

Type: Non-Functional, Automated, Dynamic
Covers: PR-REL1, PR-REL2
Initial State: System deployed in production environment
Input/Condition: Continuous operation over 30 days
Output/Result: Uptime $\geq$ 99.5 percentage; critical errors $\leq$ 0.5 percentage

How test will be performed: Automated monitoring scripts track system availability, error rates, and generate uptime reports.

NFR-ST 35. **Audit and Regulatory Reporting**

Type: Non-Functional, Manual, Static
Covers: CR-SCR3, CR-SCR4
Initial State: System operational with logging and data retention policies
Input/Condition: Generate audit report
Output/Result: Report includes user access, data retention, and compliance with standards

How test will be performed: Tester generates reports and verifies completeness and compliance with regulatory standards.

NFR-ST 36. **Usability Feedback Collection**

Type: Non-Functional, Manual, Dynamic
Covers: UHR-FB1, UHR-FB2
Initial State: System includes feedback collection mechanisms
Input/Condition: Users submit feedback after alt text generation
Output/Result: Feedback stored and categorized; system able to analyze usability trends

How test will be performed: Tester submits sample feedback and verifies storage, categorization, and accessibility of feedback for review.

NFR-ST 37. **System Responsiveness Under Load**

Type: Non-Functional, Automated, Dynamic
Covers: PR-CR7, PR-CR8
Initial State: System operational with normal load
Input/Condition: Increase concurrent user activity to 2x expected peak
Output/Result: UI response time $\leq$ 3s for 95 percentage of requests.

How test will be performed: Automated load scripts simulate multiple users; system response times recorded and analyzed.

NFR-ST 38. **Third-Party API Integration Stability**

Type: Non-Functional, Manual/Automated, Dynamic
Covers: PR-API1, PR-API2
Initial State: System connected to external APIs
Input/Condition: Perform repeated alt text generation requests
Output/Result: API responses consistent; errors handled gracefully; system continues normal operation

How test will be performed: Tester triggers repeated API calls and monitors system stability, error handling, and response consistency.

NFR-ST 39. **User Interface Localization Accuracy**

Type: Non-Functional, Manual, Dynamic
Covers: OER-LC3, OER-LC4
Initial State: Multi-language system interface
Input/Condition: Switch to each supported language
Output/Result: All messages, labels, and generated content correctly localized; no truncation or overlap

How test will be performed: Tester switches language settings and inspects UI and content for correctness and readability.

NFR-ST 40. **Cross-Platform File Upload Compatibility**

Type: Non-Functional, Manual, Dynamic
Covers: OER-FC1, OER-FC2
Initial State: System accessible from Windows, Mac, Linux
Input/Condition: Upload images from each platform
Output/Result: All images successfully uploaded and processed; no format-specific errors

How test will be performed: Tester uploads images from each platform and verifies correct processing.

NFR-ST 41. **Accessibility Feature Toggle Verification**

Type: Non-Functional, Manual, Dynamic
Covers: LFR-AR5, LFR-AR6
Initial State: Accessibility features available
Input/Condition: Enable and disable accessibility options such

as high-contrast mode and text resizing
Output/Result: Features function as intended without affecting core operations

How test will be performed: Tester toggles options and confirms UI adapts correctly and tasks remain fully functional.

NFR-ST 42. **Session Persistence Across Devices**

Type: Non-Functional, Manual, Dynamic
Covers: SR-SES1, SR-SES2
Initial State: User logged in on one device
Input/Condition: Log in from a second device
Output/Result: Session persists correctly; no data loss or overlap occurs

How test will be performed: Tester logs in from multiple devices and verifies session continuity and data integrity.

NFR-ST 43. **Authentication and Authorization Robustness**

Type: Non-Functional, Automated, Dynamic
Covers: SR-SEC1, SR-SEC2
Initial State: System user management enabled
Input/Condition: Attempt login with valid and invalid credentials, and role-based access attempts
Output/Result: Only authorized users gain access; failed attempts logged and blocked

How test will be performed: Tester executes login attempts and role-specific actions, verifying proper access control and logging.

NFR-ST 44. **Backup Frequency and Integrity Verification**

Type: Non-Functional, Automated, Dynamic
Covers: PR-RST3, PR-RST4
Initial State: System operational with scheduled backups
Input/Condition: Check backup creation and integrity after scheduled intervals
Output/Result: Backup files are created, complete, and restorable

How test will be performed: Automated scripts validate

backup presence, completeness, and successful restore operations.

NFR-ST 45. **System Latency under Concurrent Requests**

Type: Non-Functional, Automated, Dynamic
Covers: PR-CR9, PR-CR10
Initial State: System operating normally
Input/Condition: Simulate multiple concurrent requests exceeding peak load
Output/Result: Average latency remains below 2 seconds for 95**How test will be performed:** Automated load testing scripts measure response times and record metrics.

NFR-ST 46. **Accessibility Feature Consistency Across Pages**

Type: Non-Functional, Manual, Dynamic
Covers: LFR-AR7, LFR-AR8
Initial State: System accessible via web interface
Input/Condition: Navigate through all interface pages while accessibility features enabled
Output/Result: All pages reflect selected accessibility settings consistently

How test will be performed: Tester manually navigates pages and confirms consistent application of accessibility features.

NFR-ST 47. **Data Validation Accuracy**

Type: Non-Functional, Automated, Dynamic
Covers: SR-DV1, SR-DV2
Initial State: System receiving data input
Input/Condition: Submit valid and invalid data sets
Output/Result: Invalid data rejected with proper messages; valid data accepted correctly

How test will be performed: Automated tests feed data and verify validation, error messages, and acceptance of correct entries.

NFR-ST 48. **System Performance Logging**

Type: Non-Functional, Automated, Dynamic
Covers: MS-LOG8, MS-LOG9
Initial State: System operational with monitoring enabled
Input/Condition: Conduct normal and peak operations
Output/Result: Performance metrics recorded accurately; logs available for analysis

How test will be performed: Automated scripts simulate usage and check log accuracy, completeness, and integrity.

NFR-ST 49. **System Maintainability Through Modular Design**

Type: Non-Functional, Manual, Static
Covers: MS-MNT7, MS-MNT8
Initial State: System code deployed
Input/Condition: Review code modules and interdependencies
Output/Result: Modules decoupled; maintenance tasks simplified; documentation available

How test will be performed: Tester inspects codebase and verifies module independence and clear documentation.

NFR-ST 50. **System Monitoring Dashboard Functionality**

Type: Non-Functional, Manual, Dynamic
Covers: PR-MON1, PR-MON2
Initial State: Monitoring dashboard active
Input/Condition: Generate alerts and simulate metric spikes
Output/Result: Dashboard reflects accurate status and metrics in real-time

How test will be performed: Tester triggers alerts and monitors dashboard for correct visualization and metrics.

NFR-ST 51. **Accessibility Compliance Reporting**

Type: Non-Functional, Manual, Static
Covers: CR-LR4, CR-LR5
Initial State: Accessibility features and logging enabled
Input/Condition: Generate compliance report
Output/Result: Report shows coverage of WCAG 2.1 stan-

dards; any non-compliance highlighted

How test will be performed: Tester reviews reports and validates against accessibility standards checklist.

NFR-ST 52. **Data Privacy and GDPR Compliance**

Type: Non-Functional, Manual, Dynamic
Covers: SR-PR5, SR-PR6
Initial State: User data stored
Input/Condition: Access and delete personal data
Output/Result: Personal data protected and deletions effective; no unauthorized access

How test will be performed: Tester attempts access and deletion operations, verifying compliance with privacy standards.

NFR-ST 53. **High Availability Through Redundancy**

Type: Non-Functional, Automated, Dynamic
Covers: PR-REL3, PR-REL4
Initial State: System deployed with redundant servers
Input/Condition: Simulate server failure
Output/Result: System continues operation with minimal downtime; failover verified

How test will be performed: Tester disables primary server and observes continuity of service.

NFR-ST 54. **Notification Delivery and Logging**

Type: Non-Functional, Manual, Dynamic
Covers: PR-NOT1, PR-NOT2
Initial State: Notification system active
Input/Condition: Trigger various system notifications
Output/Result: Notifications sent promptly; logged and traceable

How test will be performed: Tester triggers notifications and verifies delivery, logging, and accuracy.

NFR-ST 55. **Accessibility User Preferences Persistence**

Type: Non-Functional, Manual, Dynamic

Covers: LFR-AR9, LFR-AR10
Initial State: Accessibility preferences set by user
Input/Condition: Log out and log back in
Output/Result: Preferences persist and applied automatically

How test will be performed: Tester logs out/in and verifies consistent application of user accessibility settings.

NFR-ST 56. **Version Control and Deployment Audit**

Type: Non-Functional, Manual, Static
Covers: MS-MNT9, MS-MNT10
Initial State: System deployed with version control
Input/Condition: Review deployment history
Output/Result: All deployments documented; rollback history available

How test will be performed: Tester inspects deployment logs and confirms version tracking and rollback capability.

NFR-ST 57. **Image Processing Throughput**

Type: Non-Functional, Automated, Dynamic
Covers: PR-CR11, PR-CR12
Initial State: System idle
Input/Condition: Upload batch of images for alt text generation
Output/Result: All images processed within expected time threshold

How test will be performed: Automated tests measure processing time for large image batches.

NFR-ST 58. **System Health Check Automation**

Type: Non-Functional, Automated, Dynamic
Covers: PR-MON3, PR-MON4
Initial State: System running
Input/Condition: Scheduled automated health checks
Output/Result: Alerts generated for any anomalies; uptime maintained

How test will be performed: Automated health scripts exe-

cute and log system metrics, notifying tester on anomalies.

NFR-ST 59. **User Onboarding Accessibility Guidance**

Type: Non-Functional, Manual, Dynamic
Covers: LFR-AR11, LFR-AR12
Initial State: New user account created
Input/Condition: User accesses onboarding tutorial
Output/Result: Tutorial accessible via screen reader and keyboard navigation

How test will be performed: Tester follows tutorial using accessibility tools and verifies all steps are readable and navigable.

NFR-ST 60. **Data Consistency Across Multiple Instances**

Type: Non-Functional, Automated, Dynamic
Covers: SR-DV3, SR-DV4
Initial State: System running on multiple instances
Input/Condition: Modify data concurrently
Output/Result: Data remains consistent across all instances

How test will be performed: Automated tests simulate concurrent data operations and verify consistency.

NFR-ST 61. **Accessibility Keyboard Shortcut Support**

Type: Non-Functional, Manual, Dynamic
Covers: LFR-AR13, LFR-AR14
Initial State: System accessible via keyboard
Input/Condition: Navigate and perform tasks using only keyboard
Output/Result: All interface elements operable; focus indicators clear

How test will be performed: Tester completes tasks without mouse and verifies navigation and operation.

NFR-ST 62. **End-to-End Encryption Verification**

Type: Non-Functional, Automated, Dynamic
Covers: SR-PR7, SR-PR8
Initial State: Data transfer enabled

Input/Condition: Transmit sensitive data
Output/Result: Data encrypted in transit and decrypted only by intended recipient

How test will be performed: Automated tests attempt interception; validate encryption and decryption correctness.

NFR-ST 63. **System Scalability for Concurrent Users**

2mm] **Type:** Non-Functional, Automated, Dynamic
Covers: PR-CR13, PR-CR14
Initial State: System running normally
Input/Condition: Simulate up to 5x expected user load
Output/Result: System handles load without significant performance degradation

How test will be performed: Automated load scripts generate peak traffic; monitor system response and resource usage.

NFR-ST 64. **Comprehensive Audit Trail Accessibility**

Type: Non-Functional, Manual, Static
Covers: MS-LOG10, MS-LOG11
Initial State: System logs active
Input/Condition: Access audit trail
Output/Result: Logs readable, searchable, and meet regulatory standards

How test will be performed: Tester queries audit trail and confirms completeness and accessibility.

NFR-ST 65. **Image Alt Text Accuracy Metrics**

Type: Non-Functional, Automated, Dynamic
Covers: PR-VAL1, PR-VAL2
Initial State: Alt text generation operational
Input/Condition: Process test image 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 66. **End-to-End System Security Review**

Type: Non-Functional, Manual, Static
Covers: SR-SEC3, SR-SEC4
Initial State: System deployed in production
Input/Condition: Conduct security assessment
Output/Result: All vulnerabilities identified; security policies validated

How test will be performed: Tester reviews system architecture, performs penetration tests, and verifies adherence to security standards.

4.3 Traceability Between Test Cases and Requirements

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

5 Unit Test Description

5.1 Unit Testing Scope

The purpose of unit testing for *Reading4All* is to verify the correctness, robustness, and accessibility compliance of individual components prior to system integration. Each module is tested independently using automated test scripts (PyTest) and deterministic input fixtures.

Modules in Scope:

- Image Upload & Validation Module
- Alt-Text Generation Module
- Accessibility and UI Compliance Module
- Security and Privacy Module

Modules Out of Scope: Third-party OCR engines, pretrained vision/language models, and external McMaster authentication services are assumed to be validated independently. Only the thin wrappers and internal interactions with these APIs are tested here.

5.2 Tests for Functional Requirements

5.2.1 Module 1 — Image Upload and Validation

Goal: Ensure uploaded images meet all input constraints for format, size, and type, and that invalid files are rejected gracefully.

1. UT1-UploadValidImage
   Type: Functional, Dynamic, Automatic
   Initial State: Application running; no active uploads.
   Input: Valid PNG image, 1 MB in size.
   Output: File accepted; confirmation message displayed; metadata stored in temporary session.
   Test Case Derivation: Confirms compliance with input constraints (JPEG/PNG $\leq$ 10 MB).
   How test will be performed: Run automated pytest verifying HTTP 200 response and valid JSON schema.

2. UT2-UploadInvalidFileType
   Type: Functional, Dynamic, Automatic
   Initial State: No uploads.
   Input: Unsupported file type (e.g., `.pdf`).
   Output: Error message "Unsupported file format" returned; no data stored.
   Test Case Derivation: Validates enforcement of file type constraint and secure rejection.
   How test will be performed: Send POST request with invalid MIME type; assert error 400 and log entry.

3. UT3-UploadOversizedFile
   Type: Functional, Dynamic, Automatic
   Initial State: No uploads.
   Input: PNG file $>$ 10 MB.
   Output: Upload rejected with clear error; no file stored.
   Test Case Derivation: Confirms handling of maximum size threshold.
   How test will be performed: Simulate multipart upload; verify memory cleanup and error alert.

5.2.2 Module 2 — Alt-Text Generation

Goal: Validate that image inference and text generation components produce deterministic, relevant, and correctly formatted alternative text.

1. UT4-GenerateAltText
   Type: Functional, Dynamic, Automatic
   Initial State: Valid image uploaded and accessible to model service.
   Input: Image containing labeled diagram.
   Output: Non-empty descriptive string within 3–8 s latency window.
   Test Case Derivation: Confirms compliance with generation timing and content sufficiency.
   How test will be performed: Mock ML service; assert response schema and timing $<$ T_ALT_GEN_SMALL.

2. UT5-EditAltText
   Type: Functional, Dynamic, Manual
   Initial State: Alt-text successfully generated.
   Input: User edits description and saves.
   Output: Edited text replaces old version in session storage.
   Test Case Derivation: Ensures edit functionality modifies session data only.
   How test will be performed: Selenium automation of UI; assert saved value persists on reload.

3. UT6-HandleEmptyAltText
   Type: Functional, Dynamic, Automatic
   Initial State: Image uploaded yields no model output.
   Input: Blank model return.
   Output: Error message and retry option; no text stored.
   Test Case Derivation: Confirms graceful failure and user notification.
   How test will be performed: Patch model API to return empty string; validate error log.

5.2.3 Module 3 — Accessibility and UI Compliance

Goal: Validate that all UI components meet accessibility and usability criteria (keyboard navigation, zoom, color contrast, alt text labeling).

1. UT7-KeyboardNavigation
   Type: Functional, Dynamic, Automatic
   Initial State: Application home screen loaded.
   Input: Simulated Tab and Enter key presses.
   Output: All focusable elements reachable; no trap detected.
   Test Case Derivation: Confirms WCAG 2.1 Success Criterion 2.1.1.
   How test will be performed: Automated Axe/WAVE accessibility scan with keyboard simulation.

2. UT8-ContrastValidation
   Type: Functional, Static, Automatic
   Initial State: Deployed UI snapshot available.
   Input: CSS stylesheet.
   Output: All color pairs $\geq$ 4.5:1 contrast ratio.
   Test Case Derivation: Confirms MIN_CONTRAST_RATIO threshold met.
   How test will be performed: Run Lighthouse CI contrast-check script.

3. UT9-ZoomResilience
   Type: Functional, Manual
   Initial State: Browser window at 100%.
   Input: Zoom increased to 200%.
   Output: Interface remains fully visible and interactive.
   Test Case Derivation: Ensures compliance with MAX_ZOOM_PERCENTAGE.
   How test will be performed: Manual inspection + screen-reader pass.

5.2.4 Module 4 — Security and Privacy

Goal: Verify that all authentication, encryption, and data-deletion procedures uphold confidentiality and integrity requirements.

1. UT10-LoginAuthentication
   Type: Functional, Dynamic, Automatic
   Initial State: No user session.
   Input: Valid McMaster SSO credentials.
   Output: Access granted; session token stored.
   Test Case Derivation: Ensures access restriction and session linking.
   How test will be performed: Mock OAuth SSO; assert 200 and JWT valid.

2. UT11-RejectUnauthorizedAccess
   Type: Functional, Dynamic, Automatic
   Initial State: No valid session token.
   Input: API request to /generate endpoint.
   Output: HTTP 401 Unauthorized.
   Test Case Derivation: Confirms secure access control.
   How test will be performed: Post request without auth header; verify denial and log entry.

3. UT12-TemporaryFileDeletion
   Type: Functional, Dynamic, Automatic
   Initial State: Completed alt-text generation.
   Input: Wait > 60 s.
   Output: Uploaded file deleted from temporary directory.
   Test Case Derivation: Verifies privacy compliance (FILE_DELETE_TIME).
   How test will be performed: Check directory contents before/after timeout.

5.3 Tests for Non-Functional Requirements

5.3.1 Module 5 — Performance and Reliability

Goal: Ensure responsiveness, stability, and error handling meet defined thresholds.

1. UT13-LatencyBenchmark
   Type: Dynamic, Automatic
   Input/Condition: Upload 5 images ≤ 2 MB each concurrently.
   Output/Result: Mean response ≤ 8 s; no timeouts.
   How test will be performed: Stress-test script measuring T_ALT_GEN_LARGE; record average latency.

2. UT14-FaultRecovery
   Type: Dynamic, Automatic
   Input/Condition: Force backend process crash.
   Output/Result: Recovery ≤ 5 s; no user data loss.
   How test will be performed: Docker restart test; verify persistence logs.

5.3.2 Module 6 — Usability and Accessibility Metrics

Goal: Quantify user interaction quality and accessibility performance.

1. UT15-UsabilitySurvey
   Type: Manual, Empirical
   Input/Condition: Ten participants complete key tasks.
   Output/Result: Median usability rating $\geq 3/4$; no responses < 2.
   How test will be performed: Controlled observation using evaluation rubric.

2. UT16-ScreenReaderCompatibility
   Type: Functional, Dynamic, Manual
   Input/Condition: Generate alt text and read using NVDA, JAWS, VoiceOver.
   Output/Result: All screen readers announce output correctly.
   How test will be performed: Manual auditory confirmation + accessibility log capture.

5.4 Traceability Between Test Cases and Modules

Test ID	**Module / Feature Tested**	**Supported Requirement(s)**
UT1–UT3	Image Upload and Validation	FR1, PR-RFT1
UT4–UT6	Alt-Text Generation	FR2, FR4, PR-PAR1, PR-PAR2
UT7–UT9	Accessibility and UI Compliance	LFR-AR1–AR4, UHR-EUR1–4
UT10–UT12	Security and Privacy	SR-AR1, SR-PR1, PR-SCR1–PR-SCR3
UT13–UT14	Performance and Reliability	PR-SL1–2, PR-RFT2
UT15–UT16	Usability and Accessibility Metrics	UHR-LR1, UHR-AR1, OER-IAS1

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

This is where you can place additional information.

6.1 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.

6.2 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 V&V 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 V&V is clear.
- Automated testing and verification tools are specified.

V&V 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 V&V 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.

6.3 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.

6.4 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

6.5 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.

6.6 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 History Record of user uploads and generated alt text during the current session.

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

[This section is not required for CAS 741 —SS]

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

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?
2. What pain points did you experience during this deliverable, and how did you resolve them?
3. What knowledge and skills will the team collectively need to acquire to successfully complete the verification and validation of your project? Examples of possible knowledge and skills include dynamic testing knowledge, static testing knowledge, specific tool usage, Valgrind etc. You should look to identify at least one item for each team member.
4. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?

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.