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

symbol	description
T	Test

[symbols, abbreviations, or acronyms — you can simply reference the SRS (SRS, 2025) tables, if appropriate —SS]

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

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

2 General Information

2.1 Summary

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

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

[State the challenge level (advanced, general, basic) for your project. Your challenge level should exactly match what is included in your problem statement. This should be the challenge level agreed on between you and the course instructor. You can use a pull request to update your challenge level (in TeamComposition.csv or Repos.csv) if your plan changes as a result of the VnV planning exercise. —SS]

[Summarize the extras (if any) that were tackled by this project. Extras can include usability testing, code walkthroughs, user documentation, formal proof, GenderMag personas, Design Thinking, etc. Extras should have already been approved by the course instructor as included in your problem statement. You can use a pull request to update your extras (in TeamComposition.csv or Repos.csv) if your plan changes as a result of the VnV planning exercise. —SS]

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

[Reference relevant documentation. This will definitely include your SRS and your other project documents (design documents, like MG, MIS, etc). You can include these even before they are written, since by the time the project is done, they will be written. You can create BibTeX entries for your documents and within those entries include a hyperlink to the documents. —SS]

[Don't just list the other documents. You should explain why they are relevant and how they relate to your VnV efforts. —SS]
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 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

[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 1: 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

[Plans for design verification —SS]

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

[Create a checklists? —SS]

The verification of our design will rely on reviews conducted by our fellow classmates and supervisor who will evaluate our design documents including our Module Guide (MG) and Module Interface Specification (MIS). They will also aid in performing consistency and traceability checks to ensure clear alignment between our requirements, design, and implementation.

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, our professor, Dr. Smith, will assess the technical aspects of the system, focusing on the machine learning architecture to ensure it is well designed.

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.

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 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 provided 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

[What tools are you using for automated testing. Likely a unit testing framework and maybe a profiling tool, like ValGrind. Other possible tools include a static analyzer, make, continuous integration tools, test coverage tools, etc. Explain your plans for summarizing code coverage metrics. Linters are another important class of tools. For the programming language you select, you should look at the available linters. There may also be tools that verify that coding standards have been respected, like flake9 for Python. —SS]

[If you have already done this in the development plan, you can point to that document. —SS]

[The details of this section will likely evolve as you get closer to the implementation. —SS]

The AI Generated Alternative (alt) 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 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

[If there is any external data that can be used for validation, you should point to it here. If there are no plans for validation, you should state that here. —SS]

[You might want to use review sessions with the stakeholder to check that the requirements document captures the right requirements. Maybe task based inspection? —SS]

[For those capstone teams with an external supervisor, the Rev 0 demo should be used as an opportunity to validate the requirements. You should plan on demonstrating your project to your supervisor shortly after the scheduled Rev 0 demo. The feedback from your supervisor will be very useful for improving your project. —SS]

[For teams without an external supervisor, user testing can serve the same purpose as a Rev 0 demo for the supervisor. —SS]

[This section might reference back to the SRS verification section. —SS]
The AI Generated Alternative (alt) 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 an image and generating alt text to validate that it meets their needs and requirements.

- **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 correct and captures the needs for the 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

[There should be text between all headings, even if it is just a roadmap of the contents of the subsections. —SS]

4.1 Tests for Functional Requirements

[Subsets of the tests may be in related, so this section is divided into different areas. If there are no identifiable subsets for the tests, this level of document structure can be removed. —SS]

[Include a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good here. —SS]

4.1.1 Area of Testing1

[It would be nice to have a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good here. If a section covers tests for input constraints, you should reference the data constraints table in the SRS. —SS]

Title for Test

1. test-id1

   Control: Manual versus Automatic

   Initial State:

   Input:

   Output: [The expected result for the given inputs. Output is not how you are going to return the results of the test. The output is the expected result. —SS]

   Test Case Derivation: [Justify the expected value given in the Output field —SS]

   How test will be performed:

2. test-id2

   Control: Manual versus Automatic

   Initial State:

   Input:

   Output: [The expected result for the given inputs —SS]

Test Case Derivation: [Justify the expected value given in the Output field —SS]

How test will be performed:

4.1.2 Area of Testing2

...

4.2 Tests for Nonfunctional Requirements

[The nonfunctional requirements for accuracy will likely just reference the appropriate functional tests from above. The test cases should mention reporting the relative error for these tests. Not all projects will necessarily have nonfunctional requirements related to accuracy. —SS]

[For some nonfunctional tests, you won't be setting a target threshold for passing the test, but rather describing the experiment you will do to measure the quality for different inputs. For instance, you could measure speed versus the problem size. The output of the test isn't pass/fail, but rather a summary table or graph. —SS]

[Tests related to usability could include conducting a usability test and survey. The survey will be in the Appendix. —SS]

[Static tests, review, inspections, and walkthroughs, will not follow the format for the tests given below. —SS]

[If you introduce static tests in your plan, you need to provide details. How will they be done? In cases like code (or document) walkthroughs, who will be involved? Be specific. —SS]

4.2.1 Area of Testing1

Title for Test

1. test-id1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input/Condition:

   Output/Result:

How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

4.2.2 Area of Testing2

...

4.3 Traceability Between Test Cases and Requirements

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

5 Unit Test Description

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

Module guide 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 Symbolic Parameters

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

6.4 Usability Survey Questions?

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

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?