System Verification and Validation Plan for Software Engineering

Team #1, Sanskrit Ciphers
Omar El Aref
Dylan Garner
Muhammad Umar Khan
Aswin Kuganesan
Yousef Shahin

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 (Author, 2019) tables, if appropriate —SS]

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

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

2 General Information

2.1 Summary

The software system being tested is **Sanskrit Manuscript Reconstruction Platform**, a web-based research tool designed to assist scholars in reconstructing fragmented Buddhist manuscripts through computational analysis. The system integrates machine learning, computer vision, and database management to enable users to upload digitized manuscript fragments, perform automated preprocessing and orientation correction, and generate similarity based match suggestions across fragments.

The platform consists of a front-end interface for visualization and user interaction, a back-end API for data management and orchestration, and multiple AI/ML components responsible for text extraction, edge and damage pattern analysis, and handwriting classification.

The verification and validation process will ensure that all core functionalities, including fragment upload, orientation correction, transcription assistance, and match discovery, perform correctly, reliably, and consistently across diverse manuscript inputs. It will also verify that data integrity, user authentication, and safety-critical requirements identified in the Hazard Analysis are met.

Testing will be conducted incrementally across all major milestones (Proof of Concept, Revision 0, and Revision 1) using a combination of unit testing, integration testing, usability testing, and system-level validation to confirm that the software meets its specified functional and non-functional requirements as outlined in the SRS.

2.2 Objectives

The primary objective of this Verification and Validation (V&V) Plan is to ensure that the **Sanskrit Manuscript Reconstruction Platform** meets its defined functional and non-functional requirements, operates reliably under typical research conditions, and achieves sufficient usability for its intended audience of Buddhist Studies scholars. The V&V process aims to build confidence in the correctness, stability, and scholarly utility of the system.

In-Scope Objectives:

- **Functional correctness:** Verify that all major workflows (fragment upload, image normalization, fragment matching, OCR transcription, and session management) produce accurate and expected outcomes.
- **Reliability:** Validate that the system performs consistently across multiple sessions and input types, with appropriate handling of errors, interruptions, and data corruption.
- **Data integrity and security:** Ensure that user data, manuscript fragments, and annotations are stored and transmitted securely, as outlined in the Safety and Security Requirements.
- **Usability and accessibility:** Evaluate that scholars can intuitively use the system to complete core reconstruction tasks with minimal training, supported by the usability study and user manual deliverables.
- **Model transparency:** Confirm that AI-generated suggestions (match rankings, OCR outputs, classifications) include interpretable confidence levels and require explicit human confirmation before acceptance.
- **Traceability:** Establish end-to-end verification between requirements, test cases, and hazard mitigations to ensure all critical safety-related functionalities are validated.

Out-of-Scope Objectives:

- **External library verification:** The correctness of third-party libraries and frameworks (e.g., OpenCV, PyTorch, TensorFlow) will be assumed based on their established reliability and community validation.
- **Extensive performance benchmarking:** While basic response-time and load-handling tests will be conducted, large-scale benchmarking under production conditions is beyond the project's resource and time constraints.

- **Cross-platform hardware testing:** The V&V plan focuses on web deployment via modern browsers and does not include testing on specialized hardware or legacy systems.
- **Formal verification:** Mathematical proofs or model checking of algorithms are not feasible within the capstone timeline and are deferred to potential future work.

The objectives and their limitations reflect the practical constraints imposed on the team (mainly time and resources) while ensuring comprehensive coverage of functionality, usability, and safety-critical elements central to the platform's success.

2.3 Challenge Level and Extras

This year's capstone projects do not include a challenge level. Evaluation will focus on system completeness, correctness, usability, and overall technical quality.

Extras:

Two approved extras are included as part of this project to enhance the evaluation of usability and maintainability.

- **User Manual:** A comprehensive user manual will be developed to assist scholars in using the platform effectively. It will describe the main workflows such as fragment upload, preprocessing, matching, and transcription, and provide troubleshooting guidance. The manual will also include annotated screenshots and short tutorials covering both basic and advanced usage scenarios.
- **Usability Report:** A formal usability evaluation will be conducted to assess the system's ease of use, learnability, and effectiveness for non-technical users. Feedback will be collected from scholars in Buddhist Studies and related fields through observation and structured task completion. The resulting report will summarize key usability metrics, identify design pain points, and provide recommendations for improvement.

These extras directly support the project's primary goals of accessibility and scholarly usability by ensuring that end users can efficiently interact with complex AI-driven tools through a clear, guided interface.

2.4 Relevant Documentation

This Verification and Validation Plan references all major project documents that define the system's motivation, requirements, development strategy, safety considerations, and user-facing deliverables for the **Sanskrit Manuscript Reconstruction Platform**. These documents collectively establish the foundation for testing and traceability throughout the project lifecycle.

- **Problem Statement and Goals Document** (Ciphers (2025d)) Defines the project motivation, objectives, and expected outcomes for reconstructing fragmented Buddhist manuscripts.
- **Development Plan** (Ciphers (2025b)) Outlines the workflow structure, team responsibilities, scheduling, coding standards, and continuous integration process.
- **Software Requirements Specification (SRS)** (Ciphers (2025e)) Specifies all functional and non-functional requirements, including detailed system components, interfaces, and prioritized behaviors.
- **Hazard Analysis Document** (Ciphers (2025c)) Identifies potential hazards, risk factors, and corresponding safety and security requirements that inform this V&V plan.
- **Design Documents (MG and MIS)** (Ciphers (2025a)) To be developed in later phases. These will provide architectural decomposition, module definitions, and design rationale.
- **User Manual** (Ciphers (2025g)) Provides detailed guidance for scholars on using the platform's core features, including uploading, preprocessing, matching, and transcription.
- **Usability Report** (Ciphers (2025f)) Summarizes results from usability testing and participant feedback to evaluate ease of use, learnability, and overall effectiveness.

Together, these documents establish a complete record of the project's development and verification process, ensuring traceability between requirements, design, testing, and evaluation.

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]

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]

[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]

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]

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]

3.6 Automated Testing and Verification Tools

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

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

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

3.7 Software Validation

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

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

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

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

[This section might reference back to the SRS verification section. —SS]

4 System 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

[This section should not be filled in until after the MIS (detailed design document) has been completed. —SS]

[Reference your MIS (detailed design document) and explain your overall philosophy for test case selection. —SS]

[To save space and time, it may be an option to provide less detail in this section. For the unit tests you can potentially layout your testing strategy here. That is, you can explain how tests will be selected for each module. For instance, your test building approach could be test cases for each access program, including one test for normal behaviour and as many tests as needed for edge cases. Rather than create the details of the input and output here, you could point to the unit testing code. For this to work, you code needs to be well-documented, with meaningful names for all of the tests. —SS]

5.1 Unit Testing Scope

[What modules are outside of the scope. If there are modules that are developed by someone else, then you would say here if you aren't planning on verifying them. There may also be modules that are part of your software, but have a lower priority for verification than others. If this is the case, explain your rationale for the ranking of module importance. —SS]

5.2 Tests for Functional Requirements

[Most of the verification will be through automated unit testing. If appropriate specific modules can be verified by a non-testing based technique. That can also be documented in this section. —SS]

5.2.1 Module 1

[Include a blurb here to explain why the subsections below cover the module. References to the MIS would be good. You will want tests from a black box perspective and from a white box perspective. Explain to the reader how the tests were selected. —SS]

1. test-id1

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input:

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

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

How test will be performed:

2. test-id2

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input:

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

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

   How test will be performed:

3. ...

5.2.2 Module 2

...

5.3 Tests for Nonfunctional Requirements

[If there is a module that needs to be independently assessed for performance, those test cases can go here. In some projects, planning for nonfunctional tests of units will not be that relevant. —SS]

[These tests may involve collecting performance data from previously mentioned functional tests. —SS]

5.3.1 Module ?

1. test-id1

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

Initial State:

Input/Condition:

Output/Result:

How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

5.3.2 Module ?

...

5.4 Traceability Between Test Cases and Modules

[Provide evidence that all of the modules have been considered. —SS]

References

Author Author. System requirements specification. https://github.com/..., 2019.

Team Sanskrit Ciphers. Design documents (mg and mis). https://github.com/DylanG5/sanskrit-cipher/blob/43484da11e4fb0f879f7bbf2377c21900e5b9838/docs/Design/SoftDetailedDes/MIS.pdf, 2025a. To be developed in later phases; describes system architecture and module decomposition.

Team Sanskrit Ciphers. Development plan. https://github.com/DylanG5/sanskrit-cipher/blob/43484da11e4fb0f879f7bbf2377c21900e5b9838/docs/DevelopmentPlan/DevelopmentPlan.pdf, 2025b. Describes workflow, scheduling, task assignments, and CI/CD strategy.

Team Sanskrit Ciphers. Hazard analysis document. https://github.com/DylanG5/sanskrit-cipher/blob/43484da11e4fb0f879f7bbf2377c21900e5b9838/docs/HazardAnalysis/HazardAnalysis.pdf, 2025c. Identifies potential hazards, risks, and mitigation strategies.

Team Sanskrit Ciphers. Problem statement and goals document. https://github.com/DylanG5/sanskrit-cipher/blob/43484da11e4fb0f879f7bbf2377c21900e5b9838/docs/ProblemStatementAndGoals/ProblemStatement.pdf, 2025d. Defines project motivation, scope, and expected outcomes.

Team Sanskrit Ciphers. Software requirements specification (srs). https://github.com/DylanG5/sanskrit-ciphers-requirements/blob/9938018b682709551d0a6248a0a9f6e5794112ee/index.pdf, 2025e. Specifies functional and non-functional requirements for the system.

Team Sanskrit Ciphers. Usability report. https://github.com/DylanG5/sanskrit-cipher/tree/43484da11e4fb0f879f7bbf2377c21900e5b9838/docs/Extras/UsabilityTesting, 2025f. Details findings from surveys highlighting effectiveness of platform.

Team Sanskrit Ciphers. User manual. https://github.com/DylanG5/sanskrit-cipher/blob/43484da11e4fb0f879f7bbf2377c21900e5b9838/docs/UserGuide/UserGuide.pdf, 2025g. Guides end users on how to operate the platform.

6 Appendix

This is where you can place additional information.

6.1 Symbolic Parameters

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

6.2 Usability Survey Questions?

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

Appendix — Reflection

[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?