System Verification and Validation Plan for Software Engineering

Team 2, SyntaxSentinals
Mohammad Mohsin Khan
Lucas Chen
Dennis Fong
Julian Cecchini
Luigi Quattrociocchi

November 3, 2024

Revision History

Date	Version	Notes
November 1	1.0	Initial documentation

[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 (Khan et al., 2024d) tables, if appropriate —SS]

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

This document outlines the strategies and processes used to ensure that the plagiarism detection system developed by the SyntaxSentinals team meets all functional and non-functional requirements. The primary goal of this plan is to build confidence in the correctness, usability, and performance of the system. It also focuses on identifying and mitigating potential risks, ensuring that the final product aligns with academic and competition standards. This document is organized as follows. The general information section provides an overview of the objectives, challenges, and relevant project documents used throughout the V&V process. The plan section describes the roles and responsibilities of the team members and the tools used for automated testing and verification. The system tests section lists the tests performed for both functional and non-functional requirements, with traceability to the SRS. The unit test description section details the unit testing scope, the modules tested, and the strategies for covering edge cases. Finally, the appendix contains symbolic parameters, survey questions (if applicable), and any other relevant information to support the V&V process. This plan will evolve as the project progresses, with updates following the completion of the detailed design and implementation phases.

2 General Information

2.1 Summary

The software being tested is a plagiarism detection system designed to identify similarities in Python code submissions, called SyntaxSentinels. This system utilizes natural language processing (NLP) techniques to analyze code semantics, preventing common circumvention methods like adding benign lines or altering variable names. Its core function is to allow users to input code snippets and receive a plagiarism report containing similarity scores, which, when compared to a threshold, indicate the likelihood of plagiarism. This tool is primarily intended for use in academic and competitive environments to promote fairness and integrity in code submissions.

2.2 Objectives

The primary objectives of this V&V plan are to:

- Build confidence in the programs correctness by ensuring alignment with the SRS requirements.
- Show that we have met the documented safety and security requirements (SR-SAF1- SR-SAF5) in the Hazard Analysis document.
- Demonstrate adequate usability in the program by conducting functional and non-functional tests mentioned in this document.

Out of Scope:

- Validation of any external libraries will be assumed to be handled by their maintainers.

2.3 Challenge Level and Extras

This project has been classified as having a General difficulty level, as agreed with the course instructor. Planned extras include:

- A user manual for instructors and administrators.
- Benchmarking of the tool's effectiveness compared to MOSS.

2.4 Relevant Documentation

The following documents are critical to the development and V&V efforts for this project.

- **Software Requirements Specification (SRS)**: Defines the project's requirements, guiding both verification and validation. (Khan et al., 2024d).
- **User Guide**: Provides operational instructions, relevant for usability testing. (Khan et al., 2024e).
- **Module Guide (MG)**: Outlines the system's architecture, essential for design verification. (Khan et al., 2024b).
- **Module Interface Specification (MIS)**: Details the internal modules and interfaces, critical for unit testing. (Khan et al., 2024c).
- **Hazard Analysis**: Identifies potential risks, guiding validation efforts for safety and security. (Khan et al., 2024a).

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 Plan

[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 Plan

[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 Plan

[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 Plan

[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 Plan

[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

Below are tests for functional and non-functional requirements stated in the relevant SRS (Khan et al., 2024d).

4.1 Tests for Functional Requirements

Each test section addresses a particular initial condition and input set. Therefore, each test has a particular set of outputs that the system should produce for it. A functional requirement of section 9 in the SRS (Khan et al., 2024d) is only associated with a test if the output set of the test corresponds to the fit criterion of that functional requirement, and the fit criterion should be an observable component of a system state, such as a display of a return item or notification presented. Through this association, every functional requirement is provided a test that is capable of verifying it, thereby ensuring the test areas fully cover the functional requirements.

4.1.1 Test Area 1

Applies to all FRs involving input of the system necessary for the plagiarism analysis. This currently covers FR-1 in subsection 9.1 of the SRS (Khan et al., 2024d).

Input For Analysis Test

1. test-FR1-1

   Control: Manual.

   Initial State: system is active, spot for giving system code snippet is open with zero or more existing snippets currently added.

   Input: code snippet file(s).

Output: System signifies input was received and continues activity without error, awaiting further action directives (such as giving more files or initiating analysis).

Test Case Derivation: The system will look to continually receive code snippets from user as part of a necessary step to initiate plagiarism analysis.

How test will be performed: system will be opened in the appropriate spot for receiving code and receive files in different amounts, one at a time, and will be inspected for failure in this process (types: dynamic, functional).

4.1.2 Test Area 2

Applies to all FRs involving immediate output of plagiarism analysis. This currently covers FR-2, FR-4, and FR-5 in subsection 9.1 of the SRS (Khan et al., 2024d).

Typical Detection Result Test

1. test-FR2,4,5-1

   Control: Manual.

   Initial State: system is active, has received 2 or more code snippets, and spot for initiating analysis is open.

   Input: code snippet files and command to initiate analysis (button, enter, etc.).

   Output: System presents list of similarity scores for all code snippet pairings along with corresponding threshold scores in a viewable fashion to user and any code snippet pairing that has exceeded its threshold has been flagged. System remains active, awaiting for action directives on what to do.

   Test Case Derivation: The system should possess an algorithm that analyzes code snippets and produces similarity scores and threshold scores in return. These scores should be produced between every pairing of code snippets as the plagiarism is a relative assessment between

one code piece and another. The system must able to pass this algorithm received code snippets and use its results to flag any pairing that exceeded its threshold before returning from its analysis state to the user for further interaction.

How test will be performed: system will be opened in spot to initiate analysis with 2 or more code snippets already inserted into it, and command for starting analysis will be passed from which point errors will be inspected for up until similarity scores and thresholds have been presented in viewable form, from which point code flaggings should also be available to ascertain. (type: dynamic, functional).

4.1.3 Test Area 3

Applies to all FRs involving guide documentation generation for user. This currently covers FR-3 in subsection 9.1 of the SRS (Khan et al., 2024d).

Guide Documentation Generation Test

1. test-FR3-1

   Control: Manual.

   Initial State: System is active and spot for generating guide documentation is open.

   Input: command for documentation presentation (button, enter, etc.).

   Output: System presents documentation in viewable form to user and remains active awaiting for further action directives.

   Test Case Derivation: system possesses guide documentation which it should be capable of transferring to user to provide guidance on use cases of system when prompted.

   How test will be performed: system will be opened in spot for generating guide documentation and command for guide generating documentation will be passed, all the while inspecting for errors or missing documentation (type: dynamic, functional).

4.1.4 Test Area 4

Applies to all FRs involving analysis report documentation generation for user. This currently covers FR-6 of subsection 9.1 of the SRS (Khan et al., 2024d).

Report Documentation Generation Test

1. test-FR6-1

   Control: Manual.

   Initial State: System is active, most recent plagiarism analysis has been completed since system activation and its results are available, and spot for report generation is open.

   Input: command for report documentation generation (button, enter, etc.).

   Output: System provides report documentation in viewable form to user and remains active awaiting for further action directives.

   Test Case Derivation: The system should possess the ability to aggregate the results of the plagiarism analysis which has most recently occurred and insert them into a template that can summarize all findings to a user, which will become the report document given.

   How test will be performed: Proceeding a successful analysis run from the system, the spot for generating a report will be opened and the command for generating a report will be passed, all the while inspecting for errors or missing documentation (type: dynamic, functional).

4.1.5 Test Area 5

Applies to all FRs involving creating an account within the system. This currently covers FR-7 of subsection 9.1 of the SRS (Khan et al., 2024d).

Account Creation Test

1. test-FR7-1

   Control: Manual.

Initial State: system is active, and spot for account creation is open.

Input: command for account creation (button, enter, etc.) alongside account user email and password, and the email is not associated with any existing account.

Output: System notifies user account has been successfully created for logging in with and remains active, awaiting further action directives.

Test Case Derivation: The system should be able to assess a set of account credentials is not yet within the system and proceed to add this set to the set of accounts that can be logged in with.

How test will be performed: Spot for account creation will be open, email and password not asssociated with any existing account will be given in the appropriate area, and command for account creation will be passed, all the while inspecting for any failure mode within the process (type: dynamic, functional).

2. test-FR7-2

Control: Manual

Initial State: system is active, and spot for account creation is open.

Input: command for account creation (button, enter, etc.) alongside account user email and password, and the email is associated with an existing account.

Output: System notifies user account was not possible to create for logging in with and remains active, awaiting further action directives.

Test Case Derivation: A set of pre-existing account credentials should not be possible to create an account with. Otherwise, account creation is arbitrary and not truly provided by the system as any set of account credentials are not tied to any particular account.

How test will be performed: Spot for account creation will be open, email and password asssociated with an existing account will be given in the appropriate area, and command for account creation will be passed, all the while inspecting for any failure mode within the process (type: dynamic, functional).

4.1.6 Test Area 6

Applies to all FRs involving logging into account within the system. This currently covers FR-8 of subsection 9.1 of the SRS (Khan et al., 2024d).

Account Login Test

1. test-FR8-1

   Control: Manual.

   Initial State: system is active, and spot for account login is open.

   Input: command for account login (button, enter, etc.) alongside account user email and password, and the email is associated with an existing account.

   Output: System notifies user account login was successful and and remains active, awaiting further action directives.

   Test Case Derivation: The system should be able to validate a set of pre-existing account credentials to facilitate login.

   How test will be performed: Spot for account login will be open, email and password asssociated with existing account will be given in the appropriate area, and command for account login will be passed, all the while inspecting for any failure mode within the process (type: dynamic, functional).

2. test-FR8-2

   Control: Manual.

   Initial State: system is active, and spot for account login is open.

   Input: command for account login (button, enter, etc.) alongside account user email and password, and the email is not associated with any existing account.

   Output: System notifies user account login was not successful and remains active, awaiting further action directives.

   Test Case Derivation: A set of account credentials not associated an existing account will fail to allow login as the system should determine there is no account to validate against.

How test will be performed: Spot for account login will be open, email and password not asssociated with existing account with will be given in the appropriate area, and command for account login will be passed, all the while inspecting for any failure mode within the process (type: dynamic, functional).

4.1.7 Test Area 7

Applies to all FRs involving emailing results of plagiarism analysis to users. This currently covers FR-9 of subsection 9.1 of the SRS (Khan et al., 2024d).

Result Email Test

1. test-FR9-1

   Control: Manual.

   Initial State: System is active, most recent plagiarism analysis has been completed since system activation and its results are available, and spot for emailing results is open.

   Input: command for emailing result (button, enter, etc.) and email to receive results.

   Output: System notifies email has been sent and remains active, awaiting further action directives. External to system, the specified email shall contain a .zip file possessing results of the recent analysis.

   Test Case Derivation: The system should possess the ability to condense results into a .zip file upon demand, and it should proceed to carry out emailing this file after it creates it.

   How test will be performed: Proceeding a successful analysis run from the system, the spot for emailing results will be opened and the command for sending email will be passed alongside an email for receiving results, all the while inspecting for errors in the process. The reception of the .zip file will be assessed at the very end once the system is awaiting action directives (type: dynamic, functional).

4.1.8 Test Area 8

Applies to all FRs involving visualizing plagiarism analysis results provided by user in a .zip file. This currently covers FR-10 of subsection 9.1 of the SRS (Khan et al., 2024d).

Result Visualization Test

1. test-FR10-1

   Control: Manual

   Initial State: System is active and spot for inserting .zip file containing results to produce visualization is open.

   Input: command for visualization (button, enter, etc.) and .zip file containing results.

   Output: System provides viewable visualization that corresponds to the results in the given .zip file.

   Test Case Derivation: System should possess ability to parse .zip file for relevant contents and pass it into an internal visualization template that can be filled out with what was found before proceeding to transfer it to the user.

   How test will be performed: spot for inserting .zip file to produce visualization is open and command for visualization is passed alongside a .zip file containing results, all the while inspecting for errors or missing parts of the visualziation in the process.

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 will not be filled in until after the MIS document has been completed.

References

Mohammad Mohsin Khan, Lucas Chen, Dennis Fong, Julian Cecchini, and Luigi Quattrociocchi. Hazard analysis. https://github.com/lilweege/SyntaxSentinels/blob/main/docs/HazardAnalysis/HazardAnalysis.pdf, November 2024a.

Mohammad Mohsin Khan, Lucas Chen, Dennis Fong, Julian Cecchini, and Luigi Quattrociocchi. Module guide. https://github.com/lilweege/SyntaxSentinels/blob/main/docs/Design/SoftArchitecture/MG.pdf, November 2024b.

Mohammad Mohsin Khan, Lucas Chen, Dennis Fong, Julian Cecchini, and Luigi Quattrociocchi. Module interface system. https://github.com/lilweege/SyntaxSentinels/blob/main/docs/Design/SoftDetailedDes/MIS.pdf, November 2024c.

Mohammad Mohsin Khan, Lucas Chen, Dennis Fong, Julian Cecchini, and Luigi Quattrociocchi. System requirements specification. https://github.com/lilweege/SyntaxSentinels/blob/main/docs/SRS-Volere/SRS.pdf, November 2024d.

Mohammad Mohsin Khan, Lucas Chen, Dennis Fong, Julian Cecchini, and Luigi Quattrociocchi. User guide. https://github.com/lilweege/SyntaxSentinels/blob/main/docs/UserGuide/UserGuide.pdf, November 2024e.

6 Appendix

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

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?

   While writing this deliverable, our team was able to collaboratively clarify and solidify our understanding of both the functional and non-functional requirements. This process helped us align on specific goals and create comprehensive test plans, which ensures that our testing will effectively verify that all key project requirements are met.

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

   A significant challenge we faced was the extensive amount of non-functional requirements (NFRs), with over 30 NFRs to address. Each NFR required us to develop a detailed test plan, specifying factors like type (e.g., functional, dynamic, manual), initial state, input/conditions, expected output/results, and test method. While the template helped streamline our process, the sheer volume of NFRs meant the documentation grew quickly, and managing this without sacrificing detail was challenging. We prioritized efficiency by dividing NFRs among team members and holding review sessions to ensure consistent quality and adherence to our testing criteria. This allowed us to maintain clarity

without being overwhelmed by the documentation demands. One of the main challenges we faced was the large amount of

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.

    - **Dynamic testing knowledge**:
    - **Static testing knowledge**:
    - **Automated testing tools**:

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?

    - **Lucas Chen:**
    - **Dennis Fong:**
    - **Julian Cecchini:**
    - **Mohammad Mohsin Khan:**
    - **Luigi Quattrociocchi:**