

# System Verification and Validation Plan for Software Engineering

Team 21, Alkalytics

Sumanya Gulati

Kate Min

Jennifer Ye

Jason Tran

November 2, 2024

## 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

<b>1</b>	<b>Symbols, Abbreviations, and Acronyms</b>	<b>iv</b>
<b>2</b>	<b>General Information</b>	<b>1</b>
2.1	Summary . . . . .	1
2.2	Objectives . . . . .	1
2.3	Challenge Level and Extras . . . . .	1
2.4	Relevant Documentation . . . . .	2
<b>3</b>	<b>Plan</b>	<b>2</b>
3.1	Verification and Validation Team . . . . .	2
3.2	SRS Verification Plan . . . . .	2
3.3	Design Verification Plan . . . . .	3
3.4	Verification and Validation Plan Verification Plan . . . . .	3
3.5	Implementation Verification Plan . . . . .	3
3.6	Automated Testing and Verification Tools . . . . .	3
3.7	Software Validation Plan . . . . .	4
<b>4</b>	<b>System Tests</b>	<b>4</b>
4.1	Tests for Functional Requirements . . . . .	4
4.1.1	Data Input and Storage . . . . .	5
4.1.2	Data Querying and Results . . . . .	5
4.1.3	Data analysis . . . . .	6
4.1.4	Data Hygiene . . . . .	7
4.1.5	User Access . . . . .	7
4.1.6	Data Export . . . . .	8
4.2	Tests for Nonfunctional Requirements . . . . .	8
4.2.1	Look and Feel Requirements . . . . .	9
4.2.2	Usability and Humanity Requirements . . . . .	10
4.2.3	Performance Requirements . . . . .	12
4.2.4	Operational and Environmental Requirements . . . . .	16
4.2.5	Maintainability and Support Requirements . . . . .	17
4.3	Traceability Between Test Cases and Requirements . . . . .	18
<b>5</b>	<b>Unit Test Description</b>	<b>20</b>
5.1	Unit Testing Scope . . . . .	20
5.2	Tests for Functional Requirements . . . . .	20

5.2.1	Module 1	20
5.2.2	Module 2	21
5.3	Tests for Nonfunctional Requirements	21
5.3.1	Module ?	22
5.3.2	Module ?	22
5.4	Traceability Between Test Cases and Modules	22
<b>6</b>	<b>Appendix</b>	<b>23</b>
6.1	Symbolic Parameters	23
6.2	Usability Survey Questions?	23

## 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

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

### 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]

### 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]

## 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]

Author (2019)

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

## 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 walk-throughs, code inspection, static analyzers, etc. —SS]  
[The final class presentation in CAS 741 could be used as a code walk-through. 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**

This section covers all tests for different areas of the system.

### **4.1 Tests for Functional Requirements**

The subsections below covers each major functional of the application, from uploading data to the various outputs of post query analysis. Many of the functions come from the same user flow. By testing the subsections below guarantees that all possible user flows involving the main application functionalities are working as expected.

#### 4.1.1 Data Input and Storage

The tests below provide a way to evaluate the correctness of data input and storage for the following functional requirements:

- FR-1
- FR-2
- FR-3
- FR-4

##### 1. FR-ST1

Control: Manual

Initial State: Database is running and ready to intake data

Input: Dataset to be stored, in .CSV format

Output: Data that is inputted is sent into the system, then labelled and stored successfully

Test Case Derivation: When the data is sent to the system, the data should be stored somewhere and labelled properly before it can be queried.

How test will be performed: The test can be performed by sending a test sample of varying sizes to the storage in the system.

#### 4.1.2 Data Querying and Results

Th tests below provide a way to evaluate the data querying and visualization process of the system for the following functional requirements:

- FR-5
- FR-6
- FR-7
- FR-8
- FR-9

#### 1. FR-ST2

Control: Manual

Initial State: The system is not running any jobs, and the user interface is cleared.

Input: A selection of different combinations of parameters and datasets to be queried on

Output: A human-readable and customizable visualization of correct results corresponding to the selected parameters from the input

Test Case Derivation: The expected output of the system is based on the query parameters selected. A user expects the data analysis to match with what they asked for, and the user is allowed to customize the visualized data.

How test will be performed: The database will be queried using multiple combinations of parameters, and the results will be compared against the expected output. The outputted visualization will then be tested for customizability.

### 4.1.3 Data analysis

The tests below provide a way to evaluate the data analysis in the application for the following functional requirements:

- FR-10
- FR-11

#### 1. FR-ST3

Control: Manual

Initial State: The application's cleared user interface which has not yet been used to query data, with no graph showing yet.

Input: A combination of parameters to query on for a selected dataset

Output: A small written human-readable paragraph explaining the input data.

Test Case Derivation: To be able to understand the returned data in ways other than through a graph, a written response gives the user a variety of choices.

How test will be performed: The website interface will allow the user to pick a written analysis response. The application will look for patterns and trends in the data and will output the findings.

#### **4.1.4 Data Hygiene**

The tests below provide a way to evaluate how the application maintains the data hygiene of the datasets related to the following functional requirements:

- FR-12
- FR-13

##### **1. FR-ST4**

Control: Manual

Initial State: The application's cleared user interface which has not yet been used to query data, with no graph showing yet.

Input: Dataset to be stored, in .CSV format

Output: A log file documenting errors found in the input data and/or removals of missing data.

Test Case Derivation: This is to ensure the efficiency of the querying and ensuring the database is only as big as it needs to be. This will also ensure that errors are dealt with by the application and are recorded to document any inconsistencies to increase traceability.

How test will be performed: After loading in a CSV file with some error in the data. A log file will be generated documenting the error after the application attempts to fix it.

#### **4.1.5 User Access**

This tests below provide a way to evaluate how the application allows for user login related to the following functional requirements:

- FR-14

##### **1. FR-ST5**

Control: Manual

Initial State: User interface shows a login page, with no login credentials currently used

Input: Sample user credentials

Output: The page redirects to the page designated after login

Test Case Derivation: The system must be able to authenticate users properly, and when authenticated, they should be given access to the application

How test will be performed: Sample credentials with different combinations of characters will be used to log in to ensure the fields handle credentials correctly.

#### **4.1.6 Data Export**

The tests below provide a way to evaluate the export of query reports after a session for the following functional requirement:

- FR-15

##### **1. FR-ST6 Control: Manual**

Initial State: User interface after multiple usages of data queries

Input: User clicking the button for saving or downloading

Output: Query report will be downloaded to the user's device

Test Case Derivation: The user needs to be able to get a system generated report of the queries from their session, and be able to save or download that report as needed.

How Test Will Be Performed: After making multiple queries on the data, the save/download button will be pressed to test functionality.

## **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]

Non-functional requirements will diverge from typical functional requirements testing. Listed below are different testing methods that will be used throughout the section.

**User Demo Assessment:** We will have users engage in a demo of the application, and observe how they interact with the application with respect to the requirements and criteria defined in SRS.

#### 4.2.1 Look and Feel Requirements

Look and feel testing is heavily subjective, and will require users to test. These requirements will be heavily tested with the user demo assessment.

##### 1. NFR-LF1 (User Interface)

NFR: LFR-1, LFR-3, LFR-4, LFR-5, LFR-6, LFR-7

Type: User Demo, Manual

Initial State: Fully functional application ready for user interaction, starting at the login page

Input/Condition: User engagement with application

Output/Result: Recorded observations of how the user was able to interact with the system

How test will be performed:

- User will be given temporary, working credentials
- User will use the app, starting from the login page, and navigate through its functionality

- An observer will record how long it takes for them to figure out functionality and how to navigate the application
- Users will also be given a survey about the interface after testing

## 2. NFR-LF2 (Device Compatibility)

NFR: LFR-2

Type: Manual

Initial State: Application running on multiple different devices

Input/Condition: Manual tester's engagement with application

Output/Result: A list of inconsistencies through multiple devices

How test will be performed:

- Each member of the team will use the application across multiple platforms, and try to identify any inconsistencies between them

### 4.2.2 Usability and Humanity Requirements

Usability is also a subjective area, and thus will also require the use of a user demo assessment.

## 1. NFR-UH1 (User Experience)

NFR: UHR-1, UHR-4, UHR-5

Type: User Demo, Manual

Initial State: Fully functional application ready for user interaction, starting at the login page

Input/Condition: User engagement with application

Output/Result: Recorded observations of how the user was able to interact with the system without clicking the help button or asking for outside help\*

How test will be performed:

- User will be given temporary, working credentials

- User will use the app, starting from the login page, and navigate through its functionality
- An observer will record how long it takes for them to figure out functionality and how to navigate the application

## 2. NFR-UH2 (Language and Localization)

NFR: UHR-2

Type: Manual

Initial State: Fully developed application, starting at the login page

Input/Condition: Manual tester using application

Output/Result: A list of all identified language discrepancies

How test will be performed:

- A manual tester will navigate through each web page to ensure that the only language being used is English (US)

## 3. NFR-UH3 (System Notation)

NFR: UHR-3

Type: Manual

Initial State: Application ready to take in data

Input/Condition: Sample input data

Output/Result: Error logs from unrecognized characters, or successful upload

How test will be performed:

- A tester will upload a file that has scientific and mathematical symbols
- They will note whether or not the file was uploaded successfully, and if the data was transferred correctly, with the symbols

## 4. NFR-UH4 (Accessibility)



NFR: UHR-6

Type: Manual

Initial State: Application ready for use

Input/Condition: Tester engagement using third party tools

Output/Result: List of accessibility issues in accordance to WCAG, and checklists for if page is screen-readable

How test will be performed:

- A tester launch third party tools (NVDA Reader, SiteImprove)
- Third party tools will examine web page and provide results

#### 4.2.3 Performance Requirements

These requirements can be tested in ways similar to functional requirements. Many of these requirements have thresholds defined, making testing a pass/fail basis.

Test Data Generation: Some tests will require a large, dummy set of data to be used as input in order to test how well the system can handle heavy payloads. Thus, large test sets of data will be automatically generated accordingly.

##### 1. NFR-P1 (Upload Speed)

NFR: PR-1

Type: Manual

Initial State: Application navigated to upload page, ready to upload file

Input/Condition: Sample .CSV files for input

Output/Result: Upload duration

How test will be performed:

- Different .CSV files of varying sizes will be uploaded to the system
- The upload functionality will include instructions for the system to time how long the upload took

- The total duration will be logged on the browser's console.

## 2. NFR-P2: (Query Response Time)

NFR: PR-2, PR-4

Type: Manual

Initial State: Application navigated to querying page, ready to make query

Input/Condition: Test queries

Output/Result: Query response durations

How test will be performed:

- A set of parameter/dataset combinations will be used as queries
- The duration of each query will be logged and compared to defined criteria

## 3. NFR-P3 (Website Response Time)

NFR: PR-3

Type: Manual

Initial State: Application ready to use

Input/Condition: Tester engagement

Output/Result: Average response time of buttons on website

How test will be performed:

- A tester will have a timer
- They will time the response time of different buttons, and record it

## 4. NFR-P4 (Data Visualization Speed)

NFR: PR-5

Type: Manual

Initial State: Application navigated to query page, ready to make query to generate graphs

Input/Condition: Query parameters

Output/Result: Time taken to generate graphs/visualizations

How test will be performed:

- A set of parameter/dataset combinations will be used as queries
- A tester will manually time how long it takes for a graph to be generated, or the system will log it in the console

#### 5. NFR-P5 (System Accuracy)

NFR: PR-6, PR-7, PR-8

Type: Manual

Initial State: Application navigated to query page, ready to make query

Input/Condition: Query parameters

Output/Result: A check for precision of numbers in different components of system

How test will be performed:

- A set of parameter/dataset combinations will be used as queries
- A tester will manually time how long it takes for a graph to be generated, or the system will log it in the console

#### 6. NFR-P6 (Robustness - Backend Disruption)

NFR: PR-9

Type: Manual

Initial State: Application running as normal

Input/Condition: Tester temporarily taking down back end

Output/Result: Error message displayed

How test will be performed:

- The tester will run the application as normal

- The tester will then disable back end services and ensure error messages are generated

#### 7. NFR-P7 (Robustness - Internet Disruption)

NFR: PR-10

Type: Manual

Initial State: Application running as normal

Input/Condition: Tester temporarily disconnects internet connection

Output/Result: Previously generated plots and previous queries still working

How test will be performed:

- The tester will run the application as normal
- The tester will then disconnect their device from their internet connection
- They will then try to load previous queries and previously generated plots

#### 8. NFR-P8 (User Capacity)

NFR: PR-11

Type: Manual

Initial State: Three devices ready to run application

Input/Condition: Multiple different queries run by the different devices

Output/Result: System response time while under load

How test will be performed:

- Multiple devices will make queries simultaneously
- Page responsiveness will be measured while system runs multiple queries

## 9. NFR-P9 (Storage Capacity)

NFR: PR-12

Type: Data Generation, Automated

Initial State: A database with a known amount of experiment data

Input/Condition: Large suite of dummy test data

Output/Result: Observations on system health after large payload

How test will be performed:

- Multiple, dummy sets of experiment data will be automatically generated
- Using a script, this data will be uploaded into the database
- System health will be monitored after upload is complete

### 4.2.4 Operational and Environmental Requirements

These tests are to ensure that the application works in the expected environment, for the expected users. They will be manual tests done by a person to check for an environment's compatibility with the system.

#### 1. NFR-OE1 (Operating Environment)

NFR: OER-2, OER-3, OER-4

Type: Manual

Initial State: Application running on a windows device as a web application on a Chromium based browser.

Input/Condition: Tester engagement

Output/Result: A list of all discovered issues with the application that arise due to environment compatibility

How test will be performed:

- A tester will run the web application on their Windows device, on Google Chrome

- They will use the application as normal and try to find any issues that are caused due to operating environment

## 2. NFR-OE2 (User Onboarding)

NFR: OER-5

Type: Manual, User Demo

Initial State: Application running and ready for use on home screen

Input/Condition: User engagement

Output/Result: Survey results depicting subjective complexity of onboarding process

How test will be performed:

- A user will go through the onboarding process
- A survey will be given after asking the user how complex they found the onboarding to be

### 4.2.5 Maintainability and Support Requirements

These tests aim to ensure that future users of the application will not have any issues using the application.

## 1. NFR-MS1 (Interface Intuitiveness)

NFR: MSR-4

Type: Manual, User Demo

Initial State: Application running, ready for use

Input/Condition: User engagement

Output/Result: Observations on user's ability to complete tasks without support

How test will be performed:

- A pre-defined list of tasks will be created
- A user will be given this set of tasks to perform on their own

- The user will be observed to see if they are able to perform these tasks without outside help from those conducting the test

## 2. NFR-MS2 (Cross-Browser Compatibility)

NFR: MSR-6

Type: Manual

Initial State: Multiple Chromium-based web browsers open

Input/Condition: Tester Engagement

Output/Results: All abnormal behaviour of web pages observed on each different web browser

How test will be performed:

- A tester will use the application as normal on multiple different browsers
- They will note down any issues that they find, especially if the issues are unique to the usage of a certain browser

### 4.3 Traceability Between Test Cases and Requirements

Req. ID	System Test ID
FR-1 FR-2 FR-3 FR-4	FR-SLN1
FR-5 FR-6 FR-7 FR-8 FR-9	FR-SLN2
FR-10 FR-11	FR-SL3

FR-12	FR-SL4
FR-13	
FR-14	FR-SL5
FR-15	FR-SL6
UHR-1	NFR-UH1
UHR-4	
UHR-5	
UHR-2	NFR-UH2
UHR-3	NFR-UH3
UHR-6	NFR-UH4
PR-1	NFR-P1
PR-2	NFR-P2
PR-4	
PR-3	NFR-P3
PR-5	NFR-P4
PR-6	NFR-P5
PR-7	
PR-8	
PR-9	NFR-P6
PR-10	NFR-P7
PR-11	NFR-P8
PR-12	NFR-P9
OER-2	NFR-OE1
OER-3	
OER-4	
OER-5	NFR-OE2
MSR-4	NFR-MSR1
MSR-6	NFR-MSR2



## 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, your 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.

## 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.

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?