

Verification and Validation Report: Measuring Microstructure Changes During Thermal Treatment

Team #30, ReSprint
Edwin Do
Joseph Braun
Timothy Chen
Abdul Nour Seddiki
Tyler Magarelli

March 12, 2023

1 Revision History

Date	Developer	Change
Mar 7, 2023	Abdul Nour	Updated template
Mar 8 2023	Edwin Do	Added usability test results
Mar 8 2023	Edwin Do	Added Traceability matrices
Mar 8, 2023	Timothy Chen	Added performance to the Nonfunctional Test Evaluation
Mar 8, 2023	Timothy Chen	Added unit test for input communication, current state and remote access
Mar 8, 2023	Joseph Braun	Added Sections 5, 7, 8
Mar 8, 2023	Joseph Braun	Added Reflection
Mar 8, 2023	Timothy Chen	Added Trace to Modules
Mar 8, 2023	Abdul Nour	Added Functional Requirements Evaluation & Trace to Requirements
Mar 8, 2023	Abdul Nour	Added Functional Req's Unit Tests

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test
MIN_USER_ACCEPT_RATE	90% - minimum acceptance rate
TARGET_TIME	60 seconds
INTERACT_TIME	5 seconds
MAX_MISTAKE	2
MAX_SIZE	8GB
MIN_UPTIME	30 minutes
MIN_SAMPLE_RATE	60 samples per second
TIME_ACCEPTED	1 second
ACCEPTED_SIGFIG	3 decimals

Contents

1 Revision History	i
2 Symbols, Abbreviations and Acronyms	ii
3 Functional Requirements Evaluation	1
4 Nonfunctional Requirements Evaluation	2
4.1 Usability	2
4.2 Performance	3
5 Comparison to Existing Implementation	4
6 Unit Testing	5
6.1 Input Communication Module	5
6.2 Remote Access Module	6
6.3 Current State Module	7
6.4 Calculation Module	11
6.5 User Input Validation Module	13
6.6 Hardware Input Validation Module	15
6.7 Functional Requirements Testing	16
7 Changes Due to Testing	17
8 Automated Testing	18
9 Trace to Requirements	18
10 Trace to Modules	19

List of Tables

1 Test Cases for Functional Requirements	1
2 Unit Tests for Functional Requirements	16
3 Requirements Traceability	19

List of Figures

1 Previous software user interface design	4
---	---

This document discusses the results from the test cases mentioned in the V&V plan and the changes the team made in response. Section 11 Code Coverage Metrics of this document has been removed.

3 Functional Requirements Evaluation

The following table suggests examples of system level tests that were executed in order to verify that the functional requirements of the system were met.

Table 1: Test Cases for Functional Requirements

ID	User Action	Expected Result	Result
ST1	Check "Current Source" radio button, type (1) in the "Current Supply" textbox, type (10) in the "Compliance" textbox and click "Set" button, click "Current ON" button, check "Nano-Voltmeter" radio button, click "Start Capture" button	Current Source is reset, current is set to 1mA and displayed, compliance voltage is set to 10V, current is turned on, nanovoltmeter is reset and values from the voltmeter are continuously displayed on the application along with real-time calculations	PASS
ST2	System is capturing values, thermal treatment is taking place	Critical changes in resistivity are noted in the capture log	FAIL*
ST3	Choose an "Integration Rate" of (1 PLC) from the drop-down menu	Voltmeter display shows faster reading than default	PASS
ST4	System is capturing values, thermal treatment is taking place	Changes in resistivity-temperature slopes are noted in capture log and displayed	FAIL*
ST5	Capture and current are on, click "Stop Experiment" button on remote interface	Capture and current supply are turned off	FAIL*
ST6	Start experiment and capture	Graph is displaying measurements and calculations	PASS
ST7	Enter experiment info & conduct an experiment	File output contains all experiment data	PASS

* This feature has not been implemented yet.

4 Nonfunctional Requirements Evaluation

In the section, the test results for the most important non functional requirements are shown. The team decided to focus testing primarily on Usability and Performance since they are key qualities to the success of this project.

4.1 Usability

The table below shows the results of our usability tests based on the tests in the V&V plan based on the requirements mentioned in the SRS. Each requirement can be traced to multiple unit tests and the usability survey used can be found in the Appendix.

Usability Tests				
Test Requirement	Related Unit Tests	Description	Expected Result	Result
NF-UT1	AF	Completing tasks without additional assistance.	User will be complete tasks successfully	PASS
NF-UT2	AX	Interact with interface to modify parameters.	User will be able to modify the parameters accurately and quickly.	PASS
NF-UT3	AL	Completing all tasks with limited number of mistakes	User will be able to complete all tasks with <i>MAX_MISTAKE</i>	PASS
NF-UT3	AL	Verifying if interacting with the application previously improves ease of use (learnability)	User will be able to complete the tasks more quickly and accurately the second time	PASS
NF-UT5	AS	Verifying how calculations are performed is hidden	User will not know how calculations are performed after doing the set of tasks	PASS
NF-UT6	AS	Verifying appropriate application size upon installation	User will install application onto computer and verify that the application size is less than or equal to <i>MAX_SIZE</i>	PASS

4.2 Performance

The following is the list of Non-functional tests performed on the application to evaluate the performance of the application with respect to the test requirement. Each test will be mapped to unit tests that are related to the corresponding requirement.

Performance Tests				
Test Requirement	Related Unit Tests	Description	Expected Result	Result
NF-PT1	-	Checking the minimum sampling rate of the application.	The sampling rate of the application will be equal or greater than <i>MIN_SAMPLE_RATE</i>	PASS
NF-PT2	-	Checking the time required for parameters to reflect in the application.	The parameters will reflect in the application by within <i>TIME_ACCEPTED</i>	PASS
NF-PT3	-	Checking the significant digits used for calculations and display in the application.	The significant digits seen and used in the application is accurate to <i>ACCEPTED_SIGFIG</i> .	PASS
NF-PT4	-	Checking the up-time of the application during and after usage.	The application will have a up-time equal to or more than <i>MIN_UPTIME</i> after the user completes a set of tasks.	PASS

5 Comparison to Existing Implementation

Below is an image of the existing implementation's GUI.

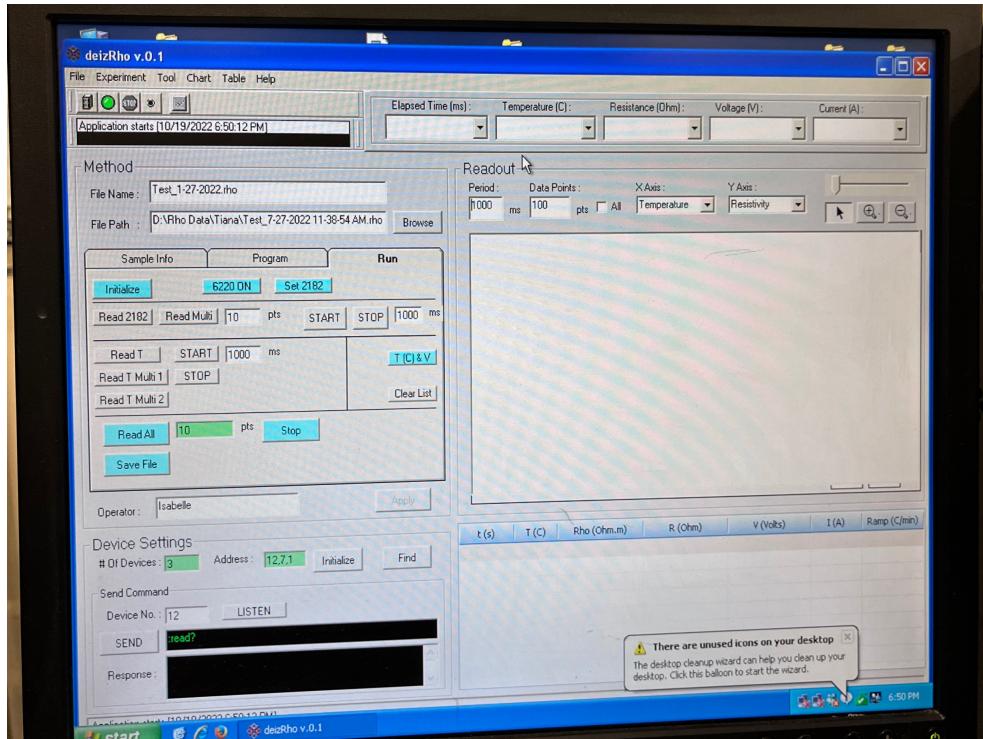


Figure 1: Previous software user interface design

The key elements of the existing implementation will also be included in our implementation. This is so that the application will be familiar to the user and as intuitive as possible. These elements are listed below:

- Method Panel: controls which device to read data from (temperature or voltage), and which file to save the data in
- Device Settings Panel: used to send SCPI commands directly to a device
- Readout: includes graphical output and listed output of relevant values (current, voltage, resistance, etc.)

The primary differences between the existing implementation and our implementation are the appearance of the GUI and the option of remote access. The existing implementation was developed for Windows XP whereas our implementation is developed for Windows 10, which gives it an updated look. One of the stretch goals for this project is to enable remote access to the application to be able to monitor and stop experiments remotely. The existing implementation does not have this functionality.

6 Unit Testing

In this section, the results of each unit test case is shown for each module. The module for graphical output utilizes a third party module so it will not be tested.

6.1 Input Communication Module

The table below shows the results of each unit test case in the Input Communication Module.

Unit Tests for Input Communication Module					
Unit Test ID	Description	Input	Expected Output	Output	Result
UT-IC1	Testing the get userInput Method	SamplingRate : 60, SampleLength : 4; SampleWidth : 2; Filename : sample1; Name : Bob; SampleName :sample1; Date : 03/01/2023	UserInput(SamplingRate : 60; SampleLength : 4; SampleWidth : 2; Filename : sample1; Name : bob; SampleName : sample1; Date : 03/01/2023)	UserInput(SamplingRate : 60; SampleLength : 4; SampleWidth : 2; Filename : sample1; Name : bob; SampleName : sample1; Date : 03/01/2023)	PASS
UT-IC2	Testing the get userInput Method		INVALID	INVALID	PASS
UT-IC3	Testing the getHardwareInput Method	Voltage : 5 ; Time : 10:00; Temperature : 50; Current: 1	HardwareInput(Voltage : 5 ; Time : 10:00; Temperature : 50; Current: 1)	HardwareInput(Voltage : 5 ; Time : 10:00; Temperature : 50; Current: 1)	PASS
UT-IC4	Testing the getHardwareInput Method		INVALID	INVALID	PASS

6.2 Remote Access Module

The table below shows the results of each unit test case in the Remote Access Module.

Unit Tests for Remote Access Module					
Unit Test ID	Description	Input	Expected Out-put	Output	Result
UT-RA1	Testing the connect Method	userName: Timothy, password 12341234	Observed connected		FAIL
UT-RA2	Testing the connect Method	userName: Bob, password 12341234	INVALID	INVALID	PASS
UT-RA2	Testing the connect Method	userName: Timothy, password 56785678	INVALID	INVALID	PASS
UT-RA2	Testing the connect Method	userName: Bob, password 56785678	INVALID	INVALID	PASS
UT-RA3	Testing the disconnect Method		FALSE		FAIL

6.3 Current State Module

The table below shows the results of each unit test case in the Current State Module.

Unit Tests for Current State Module					
Unit Test ID	Description	Input	Expected Output	Output	Result
UT-CS1	Testing the displayUserInfo Method	SamplingRate : 60, SampleLength : 4; SampleWidth : 2; Filename : sample1; Name : Bob; SampleName :sample1; Date : 03/01/2023	Observed data displayed	data displayed	PASS
UT-CS2	Testing the displayUserInfo Method	SamplingRate : ABC, SampleLength : 4; SampleWidth : 2; Filename : sample1; Name : Bob; SampleName :sample1; Date : 03/01/2023	INVALID	INVALID	PASS
UT-CS2	Testing the displayUserInfo Method	SamplingRate : -100, SampleLength : 4; SampleWidth : 2; Filename : sample1; Name : Bob; SampleName :sample1; Date : 03/01/2023	INVALID	INVALID	PASS

Unit Tests for Current State Module					
UT-CS2	Testing the displayUserInfo Method	SamplingRate : 60, SampleLength : ABC; SampleWidth : 2; Filename : sample1; Name : Bob; Sample-Name :sample1; Date : 03/01/2023	INVALID	INVALID	PASS
UT-CS2	Testing the displayUserInfo Method	SamplingRate : 60, SampleLength : -4; SampleWidth : 2; Filename : sample1; Name : Bob; Sample-Name :sample1; Date : 03/01/2023	INVALID	INVALID	PASS
UT-CS2	Testing the displayUserInfo Method	SamplingRate : 60, SampleLength : 4; SampleWidth : ABC; Filename : sample1; Name : Bob; Sample-Name :sample1; Date : 03/01/2023	INVALID	INVALID	PASS

Unit Tests for Current State Module					
UT-CS2	Testing the displayUserInfo Method	SamplingRate : 60, SampleLength : 4; SampleWidth : -2; Filename : sample1; Name : Bob; SampleName :sample1; Date : 03/01/2023	INVALID	INVALID	PASS
UT-CS3	Testing the displayHardwareState Method	Voltage : 5 ; Time : 10:00; Temperature : 50; Current: 1	Observed data displayed	data displayed	PASS
UT-CS4	Testing the displayHardwareState Method	Voltage : ABC ; Time : 10:00; Temperature : 50; Current: 1	INVALID	INVALID	PASS
UT-CS4	Testing the displayHardwareState Method	Voltage : -5 ; Time : 10:00; Temperature : 50; Current: 1	INVALID	INVALID	PASS
UT-CS4	Testing the displayHardwareState Method	Voltage : 5 ; Time : ABC; Temperature : 50; Current: 1	INVALID	INVALID	PASS
UT-CS4	Testing the displayHardwareState Method	Voltage : 5 ; Time : -10:00; Temperature : 50; Current: 1	INVALID	INVALID	PASS

Unit Tests for Current State Module					
UT-CS4	Testing the displayHardwareState Method	Voltage : 5 ; Time : 10:00; Temperature : ABC; Current: 1	INVALID	INVALID	PASS
UT-CS4	Testing the displayHardwareState Method	Voltage : 5 ; Time : 10:00; Temperature : 50; Current: ABC	INVALID	INVALID	PASS
UT-CS4	Testing the displayHardwareState Method	Voltage : 5 ; Time : 10:00; Temperature : 50; Current: -1	INVALID	INVALID	PASS

6.4 Calculation Module

The table below shows the results of each unit test case in the Calculation Module.

Unit Tests for Calculation Module					
Unit Test ID	Description	Input	Expected Output	Output	Result
UT-C1	Testing the getResistance Method	From HardwareInput ADT: Voltage = 5, Current = 1	5.000	5.000	PASS
UT-C1	Testing the getResistance Method	From HardwareInput ADT: Voltage = 4, Current = 0.6	6.667	6.667	PASS
UT-C2	Testing the getResistance Method	From HardwareInput ADT: Voltage = -5, Current = 1	Invalid	Invalid	PASS
UT-C2	Testing the getResistance Method	From HardwareInput ADT: Voltage = -4, Current = 0.6	Invalid	Invalid	PASS
UT-C3	Testing the getResistivity Method	Resistance = 3, Area = 2.5, Length = 2	3.750	3.750	PASS
UT-C3	Testing the getResistivity Method	Resistance = 2, Area = 2.8, Length = 1	5.600	5.600	PASS
UT-C4	Testing the getResistivity Method	Resistance = A, Area = 2.8, Length = 0	Invalid	Invalid	PASS
UT-C4	Testing the getResistivity Method	Resistance = 1, Area = 0, Length = BC	Invalid	Invalid	PASS
UT-C5	Testing the calcResistance Method	Voltage = 2.6, Current = 2	1.300	1.300	PASS
UT-C5	Testing the calcResistance Method	Voltage = 3.6, Current = 2	1.800	1.800	PASS

Unit Tests for Calculation Module (Continued)					
Unit Test ID	Description	Input	Expected Output	Output	Result
UT-C6	Testing the calcResistance Method	Voltage = AB, Current = 0	Invalid	Invalid	PASS
UT-C6	Testing the calcResistance Method	Voltage = 0, Current = DR	Invalid	Invalid	PASS
UT-C7	Testing the calcResistance Method	Resistance = 2, Area = 1.5, Length = 2	1.500	1.500	PASS
UT-C7	Testing the calcResistance Method	Resistance = 1.8, Area = 1.3, Length = 1.5	1.560	1.560	PASS
UT-C8	Testing the calcResistance Method	Resistance = AB, Area = 1, Length = 2	Invalid	Invalid	PASS
UT-C8	Testing the calcResistance Method	Resistance = 2, Area = 2, Length = NT	Invalid	Invalid	PASS

6.5 User Input Validation Module

The table below shows the results of each unit test case in the User Input Validation Module.

Unit Tests for User Input Validation Module					
Unit Test ID	Description	Input	Expected Output	Output	Result
UT-UI1	Testing the get userInput Method	N/A	UserInput (SamplingRate: 60, SampleLength: 4, SampleWidth:2)	UserInput (SamplingRate: 60, SampleLength: 4, SampleWidth:2)	PASS
UT-UI2	Testing the validate-FileData Method	FileName: Test, Date: 03/01/2023, Name: Test	TRUE	TRUE	PASS
UT-UI2	Testing the validate-FileData Method	FileName: Output, Date: 03/01/2023, Name: John	TRUE	TRUE	PASS
UT-UI3	Testing the validate-FileData Method	FileName: Output, Date: Some-day, Name: Tester	FALSE	FALSE	PASS
UT-UI3	Testing the validate-FileData Method	FileName: 0, Date: 03/01/2023, Name: 0	FALSE	FALSE	PASS

Unit Tests for User Input Validation Module (Continued)					
Unit Test ID	Description	Input	Expected Output	Output	Result
UT-UI4	Testing the validateSampleData Method	SamplingRate: 50, SampleLength:2, SampleWidth:5	TRUE	TRUE	PASS
UT-UI4	Testing the validateSampleData Method	SamplingRate: 60, SampleLength:1.5, SampleWidth:3	TRUE	TRUE	PASS
UT-UI5	Testing the validateSampleData Method	SamplingRate: -7, SampleLength:1.5, SampleWidth:3	FALSE	FALSE	PASS
UT-UI5	Testing the validateSampleData Method	SamplingRate: 60, SampleLength:-5, SampleWidth:3	FALSE	FALSE	PASS

6.6 Hardware Input Validation Module

The table below shows the results of each unit test case in the Hardware Input Validation Module.

Unit Tests for Hardware Input Validation Module						
Unit Test ID	Description	Input	Expected Output	Output	Result	
UT-HI1	Testing the getHardwareInput Method	-	HardwareInput (Voltage: 5, Current: 3)	HardwareInput (Voltage: 5, Current: 3)	PASS	
UT-HI2	Testing the validateParameters Method	Voltage: 3.0 , Time: 5:31 PM , Current: 1.0	TRUE	TRUE	PASS	
UT-HI2	Testing the validateFileData Method	Voltage: 3.2 , Time: 5:45 PM , Current: 1.2	TRUE	TRUE	PASS	
UT-HI3	Testing the validateFileData Method	Voltage: 10 , Time: 5:48 PM , Current: NY	FALSE	FALSE	PASS	
UT-HI3	Testing the validateFileData Method	Voltage: YT , Time: 5:51 PM , Current: 0.5	FALSE	FALSE	PASS	

6.7 Functional Requirements Testing

Table 2: Unit Tests for Functional Requirements

ID	User Action	Expected Result	Result
UT1	Check "Current Source" radio button	Current Source is reset	PASS
UT2	Type an integer (1) in the "Current Supply" textbox and click "Set" button	Current Supply is set to 1mA and displayed	PASS
UT3	Type an integer (10) in the "Compliance" textbox and click "Set" button	Compliance voltage is set to 10V	PASS
UT4	Click "Current ON" button	Current Source is supplying current	PASS
UT5	Click "Current OFF" button	Current Source stops supplying current	PASS
UT6	Check "Nano-Voltmeter" radio button	Nano-Voltmeter is reset	PASS
UT7	Click "Start Capture" button	Values from the voltmeter are continuously displayed on the application along with real-time calculations	PASS
UT8	Click "Stop Capture" button	Values and calcualtions stop generating, latest batch remains visible	PASS
UT9	Current supply is set up and capture is started	Accurate calculations of resistivity are displayed continuously	PASS

ID	User Action	Expected Result	Result
UT10	System is capturing values, thermal treatment is taking place	Critical changes in resistivity are noted in the capture log	FAIL*
UT11	System is capturing values, no thermal treatment	No critical changes in resistivity are noted	PASS
UT12	Choose an "Integration Rate" of (1 PLC) from the drop-down menu	Voltmeter display shows faster reading than default	PASS
UT13	Choose an "Integration Rate" of (5 PLC) from the drop-down menu	Voltmeter display shows no change than default rate	PASS
UT14	System is capturing values, thermal treatment is taking place	Changes in resistivity-temperature slopes are noted in caputre log and displayed	FAIL*
UT15	System is capturing values, no thermal treatment	No changes in resistivity-temperature slopes are noted in caputre log or displayed	PASS
UT16	Current supply is set up and capture is started, "Stop Experiment" button is clicked on remote interface	Capture and current supply are turned off	FAIL*
UT17	Current supply is set up and capture is started, remote interface is launched but not interacted with	Capture and current supply stay on (no change)	FAIL*

7 Changes Due to Testing

Based on the feedback from our Rev 0 Demo, we found that our application failed some requirements tests. The major failures were the lack of two main features: the graphical output (which still did not display real data as of Rev 0 Demo) and the ability to save output data to a chosen file. While we did not need to "test" our application to know this, we still consider these to be failed tests, as our application failed to meet requirements set out by the project supervisor. These two features have been implemented for the final demo.

Test ID	Failure Observed	Change(s) Made
ST6	Graph only displays dummy data	Bug with displaying real-time data was fixed
ST7	There is no method to output data to a file	File system browser and writing to output file was added

8 Automated Testing

We achieved automated unit testing through the use of the NUnit testing framework in Visual Studio. NUnit is one of the most popular test frameworks used for running tests on a .NET project.

NUnit tests are setup by first creating a new project file in Visual Studio and adding it to the solution file for your project (in our case, the application). In the new project file, a new class is created. Each unit test we want to carry out is written as a method of the test class. Since the project file for the test class is included in the same solution file as our application, we are able to call the test methods from our main application to run the tests.

9 Trace to Requirements

The table below shows the traceability between each non functional requirement from the SRS, and the test requirement in this report.

Traceability Matrix to Non Functional Requirements		
Requirement Type	Requirement(SRS)	Test Requirement
Non Functional	NFR-U1	NF-UT1
Non Functional	NFR-U2	NF-UT2
Non Functional	NFR-U3	NF-UT3
Non Functional	NFR-U4	NF-UT4
Non Functional	NFR-U5	NF-UT5
Non Functional	NFR-U6	NF-UT6
Non Functional	NFR-P1	NF-PT1
Non Functional	NFR-P2	NF-PT2
Non Functional	NFR-P3	NF-PT3
Non Functional	NFR-P4	NF-PT4

The table below shows the traceability between each functional requirement from the SRS, and the test requirement in this report.

Table 3: Requirements Traceability

System Test	Requirement	Plan
ST1	FR1	FR-T1
ST2	FR2	FR-T2
ST3	FR3	FR-T3
ST4	FR4	FR-T4
ST5	FR5	FR-T5
ST6	FR6	FR-T6
ST7	FR7	FR-T7

10 Trace to Modules

The table below shows the traceability between each module, and its respective unit tests in this report.

Traceability Matrix to Modules	
Module	Unit Tests
Input Communication	UT-IC1, UT-IC2, UT-IC3, UT-IC4
Remote Access	UT-RA1, UT-RA2, UT-RA3
Current State	UT-CS1, UT-CS2, UT-CS3, UT-CS4
File Output	-
Calculation	UT-C1, UT-C2, UT-C3, UT-C4, UT-C5, UT-C6, UT-C7, UT-C8
User Input Validation	UT-UI1, UT-UI2, UT-UI3, UT-UI4, UT-UI5
Hardware Input Validation	UT-HI1, UT-HI2, UT-HI3

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Reflection. Please answer the following question:

1. In what ways was the Verification and Validation (VnV) Plan different from the activities that were actually conducted for VnV? If there were differences, what changes required the modification in the plan? Why did these changes occur? Would you be able to anticipate these changes in future projects? If there weren't any differences, how was your team able to clearly predict a feasible amount of effort and the right tasks needed to build the evidence that demonstrates the required quality? (It is expected that most teams will have had to deviate from their original VnV Plan.)

There were several differences between what we had planned for VnV and what we ended up carrying out. When completing the initial revision of VnV plan, we had not yet completed the Design documents (MG, MIS, System Design) and so we did not have a plan for unit/module testing, only for system testing. Our initial VnV plan included three main sections: SRS Verification, Design Verification, and Implementation Verification.

For SRS Verification, we planned to meet every two weeks to discuss potential updates to the SRS doc. While we continued to meet frequently (weekly or bi-weekly), our team focussed instead on the next deliverable rather than revisiting old documents during meetings. Because of this, our SRS was not continually being updated during the project. This change occurred mainly due to time constraints, as we faced some technical issues leading up to the first demo which took priority over other tasks. In hindsight, frequently revisiting and updating the SRS certainly would have created less work for us in the long run. For future projects, though we can't predict any exact technical issues that would set us back on time, we should be able to anticipate issues arising which cause delays. We should have a plan to stay on schedule despite such issues.

For Design Verification, we planned to use the MIS checklist to ensure that requirements in the SRS are met and hazards in the Hazard Analysis are covered. Our team followed the MIS checklist when testing. We also planned to use feedback from the course instructor, teaching assistants, classmates, and our project supervisor. There were not many changes made in this section of the plan, except that our team decided to focus primarily on feedback from Dr. Zurob, as he is the end-user of the application.

For Implementation Verification, we planned to use GitHub issues and pull requests to maintain our code base. Any pull request made to the main branch requires at least two other team members to review and approve. We did not make any major changes to our Implementation Verification plan.