

# System Verification and Validation Plan for Software Engineering

Team #12, Streamliners

Mahad Ahmed

Abyan Jaigirdar

Perna Prabhu

Farhan Rahman

Ali Zia

October 28, 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

<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 . . . . .	2
2.4	Relevant Documentation . . . . .	3
<b>3</b>	<b>Plan</b>	<b>3</b>
3.1	Verification and Validation Team . . . . .	3
3.2	SRS Verification . . . . .	3
3.3	Design Verification . . . . .	4
3.4	Verification and Validation Plan Verification . . . . .	6
3.5	Implementation Verification . . . . .	6
3.6	Automated Testing and Verification Tools . . . . .	6
3.7	Software Validation . . . . .	7
<b>4</b>	<b>System Tests</b>	<b>7</b>
4.1	Tests for Functional Requirements . . . . .	7
4.1.1	Area of Testing1 . . . . .	8
4.1.2	Area of Testing2 . . . . .	9
4.2	Tests for Nonfunctional Requirements . . . . .	9
4.2.1	Area of Testing1 . . . . .	9
4.2.2	Area of Testing2 . . . . .	10
4.3	Traceability Between Test Cases and Requirements . . . . .	10
<b>5</b>	<b>Unit Test Description</b>	<b>12</b>
5.1	Unit Testing Scope . . . . .	12
5.2	Tests for Functional Requirements . . . . .	12
5.2.1	Module 1 . . . . .	12
5.2.2	Module 2 . . . . .	13
5.3	Tests for Nonfunctional Requirements . . . . .	13
5.3.1	Module ? . . . . .	14
5.3.2	Module ? . . . . .	14
5.4	Traceability Between Test Cases and Modules . . . . .	14

<b>6</b>	<b>Appendix</b>	<b>15</b>
6.1	Symbolic Parameters . . . . .	15
6.2	Usability Survey Questions? . . . . .	15

## List of Tables

1	Traceability Between Test Cases and Requirements . . . . .	10
	[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

This Verification and validation plan intends to accomplish the objective of ensuring the Large Event Management Platform for the McMaster Engineering Society meets its mission critical qualities that are most important as defined by the Software Requirements Specification (SRS) and Hazard Analysis (HA).

The mission critical qualities intended to be accomplished as part of the objective are as follows:

- **Functional Correctness:** The core system components and functionalities outlined in the SRS such as the payment system, role-based and feature-based access control and RSVP/Bus/Table signups should operate correctly and reliably to support the event operation of the platform for both admins and users.
- **System Reliability:** The platform remains stable and responsive under expected and unexpected load conditions and supports users and admins to accomplish their tasks without hindrance.
- **Safety and Security:** The platform must comply with security requirements documented in the HA relating to protection of payment information and personal user data as well as proper enforcement of role and feature based access controls.

Out of scope objectives due to project scope and resource limitations include:

- **Third party service verification:** External services provided by payment providers will not be verified. This objective is left out as these services are assumed to be reliable and safe for integration within the platform due to their existing brand recognition and usage in industry making this exclusion justified.
- **Accessibility features:** Accessibility features are beyond current needs and will not be verified for quality of usage. This objective is left out because a general user-friendly design is assumed to encompass most quality needs. This exclusion is justified as the required resources needed are disproportionate to its impact.

## 2.3 Challenge Level and Extras

This project includes several approved extras designed to enhance the quality, usability, and long-term sustainability of the system beyond its core functional requirements. These activities aim to ensure that the final product is user-friendly, maintainable, and well-documented for future MES teams.

- **Usability Testing:** The team will conduct structured usability sessions with MES organizers and student participants to evaluate workflow clarity and overall user experience. Feedback from these sessions will guide interface refinements and ensure the system remains intuitive for both attendees and administrators.
- **User Documentation:** Comprehensive user and administrator documentation will be prepared to support onboarding, maintenance, and future development. This documentation will include setup instructions, feature overviews, and troubleshooting guidelines to help future MES teams effectively operate and extend the platform.

By including these extras, the project emphasizes usability, maintainability, and user-centered design. The planned activities will ensure that the system not only meets its core functional goals but also provides a reliable, accessible, and sustainable solution for event management within the MES environment.

## 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

The SRS verification will be done through a multi-stage review process involving peer reviews, team reviews, supervisor feedback and checklist based inspection to confirm alignment with project objectives.

The verification approach will involve the following steps:

- **Peer Review (Primary reviewers):** Members of the primary reviewing team will take part in a structured review of the SRS where issues they find will be logged as GitHub Issues and resolved through tracked commits. The deliverable would be a consolidated review log documenting each issue identified and the resolution implemented by the member whose part is associated with the issue.
- **Team Review:** Based on the review provided by the primary reviewers and the resolution implemented by the team member who was



responsible for the associated issue, a different team member will work on identifying if the resolution sufficiently addresses the feedback and implements a corrective solution. The solutions that address the feedback will then be pushed to create the finalized document.

- **Supervisor Review:** In this step, a meeting will be scheduled with the project supervisor in which a structured review walkthrough will take place to validate the SRS. The structured walkthrough will be a meeting in which the SRS will be presented. The agenda will cover the following: High level overviews of each aspect of the SRS to validate main functionalities, system processes and scope boundaries, critical requirements derived from problem statements and discussing the validity of assumptions made. The validation questions that will be asked will relate to understanding if all functional requirements are measurable and verifiable and if there are any gaps or ambiguities in assumptions or scope definitions. The supervisor feedback will be documented, addressed and resolved with traceability to the specific SRS sections.
- **SRS Checklist:** In this final step the team will gather and use the below verification checklist to ensure comprehensive evaluation of the SRS using a Pass/Needs Revision criterion

Category	Verification Criteria
Accuracy/Completeness	All requirements are included and clearly defined
Consistency	No conflicting/contradicting/overlapping requirements exist across sections
Clarity	Requirements are clear and leave no room for ambiguity
Feasibility	Requirements are achievable and realistically implementable

### 3.3 Design Verification

The design verification process will confirm that the system architecture, data flow, and component design correctly satisfy all functional and non-functional requirements defined in the SRS. Verification will occur before

implementation begins to ensure that the design is both complete and technically feasible.

**Verification Approach:** The following activities will be used to verify the design:

- **Peer Review:** Each design document will be reviewed by an assigned peer team within the course. Their feedback will focus on correctness, consistency, and alignment with the SRS. All comments will be logged and tracked through GitHub Issues until resolved.
- **Team Review:** Internal walkthroughs will validate subsystem responsibilities, data flows, and interface consistency across the mobile app, admin dashboard, and backend service.
- **Supervisor Review:** The supervisor will review key design decisions and architectural diagrams to ensure the proposed solution is feasible and satisfies project constraints.
- **Traceability Check:** A mapping between SRS functional and non-functional requirements and corresponding design components will be created to confirm full coverage.

### Design Verification Checklist

Category	Verification Criteria
Completeness	All SRS requirements are represented in the design model.
Consistency	No contradictory or missing data/control flows between modules.
Feasibility	Each design decision is realistic given project constraints.
Modularity	System is divided into clear, maintainable, and loosely coupled components.
Traceability	Every requirement in the SRS can be traced to one or more design elements.

**Summary:** Design verification will rely on peer reviews by classmates, team walkthroughs, and supervisor feedback to confirm that the proposed architecture is correct, consistent, and feasible before implementation begins. Review outcomes will be documented and used to refine the design prior to implementation.

### 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 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

The software validation process will mainly be focussed on confirming the system is correctly achieving its objectives and addresses stakeholder needs and aligns with functional requirements identified in the SRS. For instance, verifying features such as the payment system, role and feature based access control and RSVP/Bus/Table sign-ups are implemented, and perform correctly.

Since the Large Event Management Platform is a new undertaking of event organization, there are no external data sources that can be used for validating if results align with intended real-world use cases or behaviors. Therefore the need arises to engage stakeholders such as club executives and event organizers in a structured review session to check that the SRS captures the requirements needed for their needs and goals. The approach taken for this can be task based inspection in which the stakeholders are focused on verifying compliance of the requirements document to their needs and assessing risk by focusing on already defined functions. Additionally, following the Rev 0 demo, a dedicated validation meeting will be done with the external supervisor of the team in which the team will demonstrate each core feature through task based inspection. The aim of the inspection is to allow the supervisor to observe and provide critique on critical workflows and functions as defined in the SRS such as creating events, processing payments, verifying access permissions and managing sign-ups. Validating these workflows, will aim to confirm their operational and functional expectations align with the SRS defined requirements focused on delivering functional requirements to the stakeholders. The combination of both stakeholder and technical validation through MES executives and the external supervisor will ensure that both system alignment to needs and practical utility is achieved

## 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 Testing<sup>1</sup>

[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 Testing<sup>2</sup>

...

## 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 Testing<sup>1</sup>

#### 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

Table 1: Traceability Between Test Cases and Requirements

Requirement ID	Requirement Description (SRS Reference)	Associated Test Cases
FR1	Ticket Purchase and Payment System. Users can purchase event tickets, process payments, handle invalid inputs, and receive confirmations.	FR1-T01-1-4, FR1-T02-1-3, FR1-T03-1-3
FR2	Bus and Table Registration. Supports seat selection, waitlists, and capacity enforcement.	FR2-T01-1-3, FR2-T02-1-3, FR2-T03-1-3
FR3	Waivers and Preferences. Allows users to submit waivers and preferences and stores data securely for organizer access.	FR3-T01-1-2, FR3-T02-1-2, FR3-T03-1-2

**Table 1 (continued)**

<b>Requirement ID</b>	<b>Requirement Description (SRS Reference)</b>	<b>Associated Test Cases</b>
FR4	Role-Based Access Control (RBAC/FBAC). Verifies authentication, authorization, and correct privilege handling.	FR4-T01-1-3, FR4-T02-1-3, FR4-T03-1-3
FR5	Event Creation and Management. Enables organizers to create, edit, archive, and manage events.	FR5-T01-1-3, FR5-T02-1-3, FR5-T03-1-3
NFR1	Performance. Measures latency and throughput during registration, RSVP, and check-in.	NFR1-T01-1-2, NFR1-T02-1-2, NFR1-T03-1-2
NFR2	Data Integrity. Ensures no inconsistent or duplicate data appears under failure or concurrency.	NFR2-T01-1-2, NFR2-T02-1, NFR2-T03-1, NFR2-T04-1
NFR3	Usability. Evaluates ease of use, completion rate, and error recovery in user testing.	NFR3-T01-1-3
NFR4	Security Compliance. Checks RBAC enforcement, authentication, and protection of sensitive data.	NFR4-T01-1, NFR4-T02-1, NFR4-T03-1
NFR5	Reliability. Verifies uptime and system stability under continuous use.	NFR5-T01-1
NFR6	Auditability. Ensures all registration, payment, and event actions are logged for traceability.	NFR6-T01-1-2



## 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.

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?