System Verification and Validation Plan for Software Engineering

Team #5, Money Making Mauraders
Zhenia Sigayev
Justin Ho
Thomas Wang
Michael Shi
Johnny Qu

October 26, 2025

Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

[The intention of the VnV plan is to increase confidence in the software. However, this does not mean listing every verification and validation technique that has ever been devised. The VnV plan should also be a **feasible** plan. Execution of the plan should be possible with the time and team available. If the full plan cannot be completed during the time available, it can either be modified to "fake it", or a better solution is to add a section describing what work has been completed and what work is still planned for the future. —SS]

[The VnV plan is typically started after the requirements stage, but before the design stage. This means that the sections related to unit testing cannot initially be completed. The sections will be filled in after the design stage is complete. the final version of the VnV plan should have all sections filled in. —SS]

Contents

List of Tables

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

List of Figures

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

1 Symbols, Abbreviations, and Acronyms

symbol	description
BUC	Business Use Cases

[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]

The MES Finance Tracking Platform is a web application that centralizes club reimbursement submissions, reviews, and approvals for the MES with role based access and receipt storage. The front end user interface will be tested for usability and functionality to ensure a smooth user experience. The back end data storage will be tested for data integrity and security to ensure sensitive financial information is protected. The database will be tested for performance and reliability to ensure quick access to financial records.

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]

In Scope Objectives:

- Demonstrate software correctness of end-to-end user interface reimbursement workflow

- Ensure data integrity and security of sensitive financial information

- Meet all performance/capacity targets for database access times

- Ensure financial accuracy of reimbursement camera captures and calculations

Out of Scope Objectives:

- Comprehensive usability testing for all potential user personas due to limited resources.

2.3 Extras

[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]

N/A

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]

- SRS Documentation

- Hazard Analysis Documentation

- Problem Statement and Goals Documentation

- Design documents

- Module Guide Documentation

- Module Interface Specification Documentation

- Development Plan Documentation

- Unit Testing Plan Documentation

3 Plan

3.1 Verification and Validation Team

Name	Role	Responsibilities
Thomas, Justin, Zhenia, Michael, Johnny	Member of MES Finance Tracker Development Team	Development, correctness and usability testing.
Luke	Capstone Project Supervisor	Outline all requirements and ensure all requirements have been met and considered.
Member of MES Finance Team	Stakeholder	Provide feedback on usability and correctness of financial calculations.
Member of MES Clubs	Stakeholder	Provide feedback on usability and correctness of reimbursement workflow.

3.2 SRS Verification

Formal SRS Inspection
Individuals Involved: Team Members, Luke (Supervisor)
Inputs: SRS Document and Checklist of Requirements and BUC
Procedure:

1. Pre-read: all reviewers must read, annotate the SRS documentation, and prepare list of any questions and concerns.

2. Meeting Walkthrough: reviewers meet to discuss if all requirements listed within the SRS are *clear, verifiable, traceable* and have been met. They will also discuss if each BUC has been captured completely and accurately.

3. Log defects, classify (major/minor), assign owners.

4. Re-review and redo the process; sign-off when exit criteria are met.

End Criteria: All major defects resolved; each requirement marked *clear, verifiable, traceable.*
Outputs: Logged defects with resolutions; signed-off SRS document verification.

3.3 Design Verification

Task-Based Validation with Stakeholders
Individuals Involved: Team Members, MES Finance Team, Club Executives
Inputs: SRS Document and Checklist of Requirements and BUC, Each Stakeholder will provide list of desired functionality from the application
Procedure:

1. Pre-read: all team members must review stakeholder desired functionality list and list questions or concerns where applicable.

2. Meeting Walkthrough: reviewers meet to discuss with each individual stakeholder to see if requirements have been met. They will also discuss if each BUC has been captured completely, accurately, and is satisfactory compared to their initial list.

3. Log defects, classify (major/minor), assign owners.

4. Re-review and redo the process; sign-off when exit criteria are met.

End Criteria: All stakeholders are satisfied with the apps functionality and each requirement marked *clear, verifiable, traceable.*
Outputs: Logged defects with resolutions; signed-off SRS document verification.
Checklist to be Used for Design Verification

- Pages use consistent layout and design

- Buttons and links look clickable, disabled and hover states are differentiated

- Forms are easy to understand and fill out, with clear labels, units/currencies, and instructions

- Important actions are visually prominent

- Empty states explain what to do next

- Member can submit reimbursement with all required fields and receipt uploads

- File uploads only accept allowable types/sizes; oversized files return error message

- Reviewers can open request, see all details, and Approve/Reject with comments

- Notifications are sent/shown at the correct times (upon submission and decision)

- Members can see their own requests and current statuses easily

- Amounts always show 2 decimal points and use the correct currency symbol

- Dates and times are clearly outlined

- Each role can only access their permitted features, information, and actions

- Approved requests are unable to be edited or resubmitted

- Sensitive pages require login, and sessions time out after period of inactivity

- Admin features are simple and hard to misuse

- Backups/restore plan exists and receipts are included

3.4 Verification and Validation Plan Verification

[The verification and validation plan is an artifact that should also be verified. Techniques for this include review and mutation testing. —SS]
[The review will include reviews by your classmates —SS]
[Create a checklists? —SS]

3.5 Implementation Verification

[You should at least point to the tests listed in this document and the unit testing plan. —SS]
[In this section you would also give any details of any plans for static verification of the implementation. Potential techniques include code walkthroughs, code inspection, static analyzers, etc. —SS]
[The final class presentation in CAS 741 could be used as a code walkthrough. There is also a possibility of using the final presentation (in CAS741) for a partial usability survey. —SS]

3.6 Automated Testing and Verification Tools

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

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

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

3.7 Software Validation

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

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

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

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

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

Stakeholder UAT Against Acceptance Criteria

Individuals Involved: Team Members, MES Finance Team, Club Executives

Points of Clarification:

1. Ensure all MES software acceptance criteria have been met and incorporated into the final product.

2. Ensure application workflow meets the needs of club executives and MES finance team members.

3. Ensure all performance and capacity targets have been met in terms of concurrent users, API response times, and database access times.

4. Ensure all financial calculations and camera receipt captures are accurate and reliable.

5. Ensure application is usable and intuitive for all intended user personas during Rev 0 demo

4 System Tests

[There should be text between all headings, even if it is just a roadmap of the contents of the subsections. —SS]

4.1 Tests for Functional Requirements

[Subsets of the tests may be in related, so this section is divided into different areas. If there are no identifiable subsets for the tests, this level of document structure can be removed. —SS]

[Include a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good here. —SS]

4.1.1 Area of Testing1

[It would be nice to have a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good here. If a section covers tests for input constraints, you should reference the data constraints table in the SRS. —SS]

Title for Test

1. test-id1

   Control: Manual versus Automatic

   Initial State:

8

Input:

Output: [The expected result for the given inputs. Output is not how you are going to return the results of the test. The output is the expected result. —SS]

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

How test will be performed:

2. test-id2

   Control: Manual versus Automatic

   Initial State:

   Input:

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

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

   How test will be performed:

4.1.2   Area of Testing2

...

4.2   Tests for Nonfunctional Requirements

[The nonfunctional requirements for accuracy will likely just reference the appropriate functional tests from above. The test cases should mention reporting the relative error for these tests. Not all projects will necessarily have nonfunctional requirements related to accuracy. —SS]

[For some nonfunctional tests, you won't be setting a target threshold for passing the test, but rather describing the experiment you will do to measure the quality for different inputs. For instance, you could measure speed versus the problem size. The output of the test isn't pass/fail, but rather a summary table or graph. —SS]

[Tests related to usability could include conducting a usability test and survey. The survey will be in the Appendix. —SS]

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

5 Unit Test Description

5.1 Unit Testing Scope

5.2 Tests for Functional Requirements

5.2.1 Module 1

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

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?