System Verification and Validation Plan for Software Engineering

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

October 27, 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
  - The SRS is the formal documentation of the functional and non-functional requirements for the project. All requirements that must be verified and validated are specified and elaborated on in the SRS.
- Hazard Analysis Documentation

  - Due to the project's nature of processing and storing financial data, many non-functional requirements arise from hazards related to the security and integrity of data handling. Previously discovered hazards can also inform caution points during the verification and validation processes.

- Problem Statement and Goals Documentation

  - Validation of the requirements proposed in the SRS must relate to the overall goals of the project. The *Problem Statement and Goals* documentation includes these high-level goals of the project, and is the primary source of information when the client and/or supervisors cannot be reached for clarification.

- Development Plan Documentation

  - The development process is to achieve the goals and meet the requirements of the project. In the case that any requirements are later invalidated, the plans for development must be modified correspondingly.

- Design Documentation

  - The software must be designed such that it meets the verification methods outlined in the VnV. For requirements that can be unit-tested, specifics will be included in the unit testing documentation. Many qualitative requirements, however, must be manually evaluated according to the procedures set forth in this document.

- Unit Testing Plan Documentation

  - The *Unit Testing Plan* documentation will be required to reference the VnV Plan for outlined procedures for automatic tests, where applicable.

[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

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

[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]

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

[Plans for design verification —SS]
[The review will include reviews by your classmates —SS]
[Create a checklists? —SS]
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]

Objective: Ensure that the V&V Plan accurately defines all verification and validation activities, aligns with the SRS and design documentation, and includes measurable procedures for system and unit testing.
Individuals Involved: Team Members, Luke (Supervisor), Peer Reviewers
Inputs: Current version of the V&V Plan, SRS, Design, and Unit Testing documents
Procedure:

1. All reviewers read and annotate the plan, marking unclear, inconsistent, or incomplete sections.
2. The team presents the plan to peers and the supervisor, who assess completeness, coverage, and clarity.
3. Reviewers confirm that all required sections (team roles, verification methods, validation methods, test case traceability, and schedules) are properly included and justified.
4. Any missing or unclear items are documented, categorized (minor/major), and assigned for correction.
5. The plan is updated and re-evaluated until all checklist items are satisfied.

Exit Criteria: all major issues have been resolved, every section meets consistency and completeness standards, plan is reviewed by supervisor.
Outputs: Review log containing identified issues and resolutions and final version version of the V&V Plan.

Verification Checklist:

- All sections of the V&V Plan are present and clearly labeled.
- Objectives and scope are clearly defined and align with the SRS.
- Verification and validation methods are feasible and well justified.
- Roles and responsibilities of team members are specified.
- Traceability between requirements, design, and test cases is clear.
- Testing procedures and success criteria are measurable and realistic.
- Inputs, outputs, and review artifacts are well defined.
- Peer and supervisor review process is documented.
- Formatting, consistency, and terminology conform to standards.
- All identified issues or inconsistencies have been resolved.

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]

Objective: Ensure the correctness, quality, and maintainability of the codebase and its compliance with requirements through systematic testing and review.

Relation to Other Documents: The system-level tests described in this document validate the integrated behavior of the MES Finance Tracking Platform, while the unit-level tests verify the correctness of individual components and modules.

Verification Activities:

1. **Code Walkthroughs:** Conducted bi-weekly among developers to ensure code readability, adherence to the design specification, and appropriate documentation. Each walkthrough focuses on newly implemented modules or functions to confirm alignment with the Module Interface Specification (MIS).

2. **Code Inspections:** Performed at key development milestones. Each inspection evaluates code clarity, conformance to coding standards, naming conventions, and consistency across modules. Defects and deviations are logged and resolved prior to integration.

3. **Static Analysis:** Automated static analyzers (e.g., `ESLint` for JavaScript / TypeScript) will be used to detect syntax errors, potential vulnerabilities, and code smells. Additional tools such as `Prettier` will enforce formatting consistency. Code quality metrics, including cyclomatic complexity and unused dependency checks, will be collected to ensure maintainability.

4. **Unit Testing:** Each module will undergo automated testing with a minimum of 80% code coverage. Unit tests verify logical correctness, handle boundary conditions, and confirm expected output against defined input cases.

5. **Integration Verification:** Conducted after all modules pass their unit tests. Integration testing ensures modules interface correctly and data flows as defined in the MIS. Errors detected during integration will be logged and traced back to responsible modules.

6. **System and Acceptance Testing:** System tests validate the complete software against functional and nonfunctional requirements. Stakeholder User Acceptance Testing (UAT) will confirm that the application meets all acceptance criteria.

Inputs:

- Implementation codebase and supporting libraries
- Unit testing plan and results
- Static analysis reports and linting logs

- Integration and system test reports
- MIS and design documentation

Outputs:

- Verified implementation source code
- Code inspection and walkthrough reports
- Static analysis summaries and coverage metrics
- Approved unit, integration, and system test results

End Criteria:

- All modules pass unit and integration tests without critical defects.
- Static analysis yields no unresolved warnings or high-severity vulnerabilities.
- Implementation verified to meet all functional and nonfunctional requirements.
- Supervisor and stakeholders approve code quality and testing outcomes.

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]

To ensure efficiency and consistency in testing, automated testing and verification tools will be utilized where possible. As the project spans a wide

spread of technologies, languages, and frameworks, multiple domain-specific tools will be required. The following section outlines the portions of testing where automated testing tools will be employed; specific tools may not yet be specified at the current stage of the project, but potentially applicable tools will be included as examples.

- **Linting and Style:** Linting and style tools will be used to improve and preserve code readability and maintainability.
  - **Tools:** Pylint (Python), ESLint (Javascript)
- **Unit Testing:** Unit testing will be performed for each part of the project. Due to the project's large technical scope, multiple unit testing suits may be required across different languages/frameworks.
  - **Tools:** pytest (Python), Jest/Cypress/Playwright (Javascript), Postman (API)
- **Load Testing:** As the system is expected to be used for many MES-affiliated clubs, load testing on the APIs and database to ensure availability and scalability.
  - **Tools:** JMeter (Java-based), Locust (Python-based)
- **Test Scripts:** Any testing functionality not covered by existing tools but requires automatic processing may be faciliated through custom testing scripts. As an example, this may include scripts that routinely verify database integrity through queries.

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]

This section describes the planned system tests for verifying the MES Finance Tracking Platform. Each subsection corresponds to one or more functional requirements defined in the SRS. The goal is to ensure that all reimbursement-related workflows—submission, review, and status tracking—perform as intended.

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]

The following tests verify that the MES Finance Tracking Platform correctly implements the functional requirements defined in Section 9.1 of the SRS. These tests ensure that all features related to reimbursement submission, review, and audit logging behave as expected.

4.1.1 Area of Testing — Reimbursement Submission

This area covers **Functional Requirements FRQ-1** and **FRQ-4**, ensuring that club members can submit expense claims with attached digital receipts and that data is permanently stored.

Title for Test: Submit Expense Reimbursement Request
Test-id: FRQ-1-SUBMIT

- **Control:** Manual versus Automatic
  Automatic (performed using UI automation with manual verification)
- **Initial State:**
  The user is logged in as a valid club member with an existing club ID. No reimbursement requests are currently active for that user.
- **Input:**
  Receipt file (PDF or image $\leq$10 MB), amount = \$100, reimbursement form fields completed.
- **Output (Expected Result):**
  The request appears in the user's dashboard with status = "Submitted." The uploaded receipt is stored and accessible under that request.
- **Test Case Derivation:**
  Based on BUC-2 ("Submitting Reimbursement Request") and FRQ-1/FRQ-4 in the SRS. Confirms form submission, file upload, and database persistence.

- **How Test Will Be Performed:**
  Using Cypress or Playwright to automate UI submission, then verifying MongoDB entries and front-end confirmation.

4.1.2 Area of Testing — Reimbursement Review Workflow

This area verifies **FRQ-2**, **FRQ-3**, and **FRQ-5**, ensuring that MES reviewers can view submissions, approve/reject them, and that actions are recorded in the audit trail.

Title for Test: Review and Approve Reimbursement Request
Test-id: FRQ-2-REVIEW

- **Control:** Manual versus Automatic
  Semi-automatic (UI actions by MES reviewer, back-end verification automated)
- **Initial State:**
  A reimbursement request exists in "Submitted" status. MES reviewer is logged in with admin privileges.
- **Input:**
  Reviewer selects request → clicks "Approve" → enters approval note.
- **Output (Expected Result):**
  Request status = "Approved." Submitter is notified. Audit log records: {timestamp, reviewerID, action: Approved}.
- **Test Case Derivation:**
  Based on BUC-3 ("Reviewing Reimbursement Request") and FRQ-2/FRQ-5 in the SRS.
- **How Test Will Be Performed:**
  UI interaction test plus backend log verification using database queries to ensure audit trail persistence.

4.1.3 Area of Testing — Reimbursement Status Tracking

This test verifies **FRQ-3** and **FRQ-6**, ensuring that submitters and authorized users can view real-time reimbursement statuses.

Title for Test: View Reimbursement Status
Test-id: FRQ-3-STATUS

- **Control:** Manual versus Automatic
  Automatic (API and UI check)
- **Initial State:**
  Multiple reimbursement requests exist with different statuses (submitted, under review, approved).
- **Input:**
  User navigates to the "My Requests" page or queries the API endpoint `/api/reimbursements`.
- **Output (Expected Result):**
  The UI displays the correct status for each request. The API returns accurate data matching the database state.
- **Test Case Derivation:**
  Derived from FRQ-3 and FRQ-6, validating that access is role-based and statuses are synchronized with the database.
- **How Test Will Be Performed:**
  Automated API tests (Postman/pytest) and front-end UI verification to cross-validate consistency.

4.1.4 Area of Testing — Receipt Persistence and Security

This area validates the data integrity requirement from **FRQ-4** and **Security Requirements INT-1, PRI-1**.

Title for Test: Verify Receipt Storage Security
Test-id: FRQ-4-STORAGE

- **Control:** Automatic
- **Initial State:**
  One reimbursement submission with a receipt attached.
- **Input:**
  Query the database and storage bucket for the receipt file.
- **Output (Expected Result):**
  Receipt file is encrypted at rest (per SRS Section 16.3), accessible only to authorized users, and retrievable without corruption.

- **Test Case Derivation:**
  From FRQ-4 and SRS Section 16 (INT, PRI).

- **How Test Will Be Performed:**
  Check encryption metadata and permissions via the cloud storage API (AWS S3 or Vercel storage).

4.1.5 Area of Testing — Audit Logging

This area ensures **FRQ-5** is satisfied by verifying audit log entries for all major actions.

Title for Test: Audit Log Verification
Test-id: FRQ-5-AUDIT

- **Control:** Automatic

- **Initial State:**
  System contains multiple reimbursement records with various user actions.

- **Input:**
  Run log export query for "Approve" or "Reject" events.

- **Output (Expected Result):**
  Each event entry includes timestamp, userID, actionType, and status change. Logs persist for at least 3 years per SRS Section 16.4.

- **Test Case Derivation:**
  Derived from FRQ-5 and Audit Requirements [AUD-1, AUD-2].

- **How Test Will Be Performed:**
  Automated test script queries audit collection and validates schema and retention policies.

4.2 Tests for Nonfunctional Requirements

The following tests verify that the MES Finance Tracking Platform meets the nonfunctional requirements defined in the SRS. These include performance, usability, security, and maintainability tests to ensure the system is robust, efficient, and user-friendly.

4.2.1 Area of Testing — Performance and Latency

This area covers **Speed and Latency Requirements [SaL]** and partially overlaps with **Robustness or Fault-Tolerance Requirements [FLT]**.

Title for Test: API and Page Load Performance Test
Test-id: NFL-SAL-1

- **Control:** Automatic
- **Initial State:**
  System deployed to staging environment with seeded database (100 clubs, 1,000 reimbursement requests).
- **Input:**
  Execute 100 concurrent API requests for reimbursement data and record response times. Load homepage and dashboard in Chrome using Lighthouse.
- **Output (Expected Result):**
  API response times average below 1 second; page load times under 3 seconds; file uploads (< 10 MB) complete within 5 seconds.
- **Test Case Derivation:**
  Derived from SRS Section 13.1 [SaL-1, SaL-2, SaL-3].
- **How Test Will Be Performed:**
  Automated load testing with tools such as JMeter or Locust, combined with Lighthouse performance scoring. Results summarized in a table comparing observed times to SRS thresholds.

4.2.2 Area of Testing — Usability and Accessibility

This area validates **Ease of Use and Learning [EUL]**, **Accessibility [ABL]**, and **Style [STY]** requirements.

Title for Test: Usability and Accessibility Evaluation
Test-id: NFL-USAB-1

- **Control:** Manual (survey-based)
- **Initial State:**
  A functioning system prototype accessible by 5 test users representing different roles (club member, executive, MES reviewer).

- **Input:**
  Users perform typical tasks (submit reimbursement, review request, check status). Afterward, they complete a usability survey using a 5-point Likert scale.

- **Output (Expected Result):**
  Average ease-of-use rating $\geq 4/5$. All UI components are keyboard-navigable and meet WCAG 2.1 AA color contrast.

- **Test Case Derivation:**
  Based on SRS Sections 11 and 12, especially [EUL-1] and [ABL-1 – ABL-5].

- **How Test Will Be Performed:**
  Observation-based usability session followed by survey analysis. Accessibility verified with automated tools (axe-core, Lighthouse accessibility audits).

4.2.3 Area of Testing — Security and Privacy

This area ensures compliance with **Access [ACS]**, **Integrity [INT]**, and **Privacy [PRI]** requirements.

Title for Test: Role-Based Access and Data Protection Test
Test-id: NFL-SEC-1

- **Control:** Automatic with manual verification

- **Initial State:**
  Test users include a general member, club executive, and MES admin, each authenticated with proper roles.

- **Input:**
  Attempt unauthorized access to restricted endpoints (e.g., reviewing reimbursements as a non-admin). Submit tampered form payloads.

- **Output (Expected Result):**
  Unauthorized requests receive HTTP 403 responses; input validation prevents injection/XSS; all communication uses HTTPS (TLS 1.2+). Sensitive fields are encrypted in the database.

- **Test Case Derivation:**
  Derived from SRS Section 16: [ACS-1, INT-1, PRI-1].

- **How Test Will Be Performed:**
  Automated penetration testing with OWASP ZAP or Burp Suite; verification of encryption and RBAC through API inspection and code review.

4.2.4 Area of Testing — Maintainability and Supportability

This area validates **Maintenance [MNT]** and **Supportability [SUP]** requirements.

Title for Test: Code Quality and Documentation Compliance
Test-id: NFL-MNT-1

- **Control:** Static test (manual review and CI enforcement)

- **Initial State:**
  Project repository set up with ESLint and Prettier configuration as specified in the SRS.

- **Input:**
  Run code through linting and formatting checks; review developer documentation for completeness.

- **Output (Expected Result):**
  No critical linting errors; documentation covers setup, deployment, and troubleshooting. All dependencies are updated to current minor versions.

- **Test Case Derivation:**
  Derived from SRS Section 15 [MNT-1, SUP-1, SUP-2].

- **How Test Will Be Performed:**
  CI pipeline enforces linting and formatting. Manual inspection ensures documentation matches environment setup and maintenance requirements.

4.3 Traceability Between Test Cases and Requirements

[Provide a table that shows which test cases are supporting which requirements. —SS]

This table maps the high-level Functional Requirements to the supporting Non-Functional Requirements that must be considered during testing to ensure the system's overall quality, security, and performance.

Table 1: Traceability Matrix for MES Finance Tracking Platform

Functional Requirement (FRQ)	**Supporting Non-Functional Requirement IDs (NFR)**
FRQ-1: The system shall allow MES clubs and teams to submit expense claims.	EUL-1, UaP-2, ABL-1, ABL-3, ABL-5, SaL-1, SaL-2, FLT-2, CAP-1, ACS-1, ACS-2, INT-1, PRI-1, PRI-2, AUD-1, CUL-1, SEI-2, AUL-1, STD-1, STD-2
FRQ-2: The system shall provide MES reviewers with tools to efficiently review, approve, or reject the reimbursement requests.	APP-1, APP-2, STY-3, EUL-1, ABL-1, ABL-5, SaL-1, PRE-1, FLT-1, CAP-1, ACS-1, ACS-2, INT-1, PRI-1, PRI-2, AUD-1, AUD-2, CUL-1, SEI-2, AUL-1, AUL-2, STD-1, STD-2
FRQ-3: The system shall track the status of each expense claim (e.g., submitted, under review, approved, rejected, reimbursed).	APP-2, STY-3, ABL-1, SaL-1, SaL-2, FLT-3, CAP-2, EXT-2, LNG-2, ACS-1, ACS-2, PRI-1, AUD-1, AUD-2, SEI-2, AUL-2, STD-1
FRQ-4: The system shall permanently store and retain digital receipt submissions.	SaL-3, PRE-1, FLT-2, CAP-2, LNG-1, PRI-1, PRI-2, AUD-2, SEI-1, SEI-2, SEI-3, AUL-2, RDL-1, STD-1

Functional Requirement (FRQ)	Supporting Non-Functional Requirement IDs (NFR)
FRQ-5: The system shall maintain an audit trail that records who submitted, reviewed, approved, or denied each expense claim.	FLT-2, CAP-2, LNG-1, ACS-2, PRI-2, AUD-1, AUD-2, IMM-1, SEI-2, SEI-3, AUL-1, AUL-2, RDL-1, STD-1, STD-3
FRQ-6: The system shall enable access to club expense submissions to the submitters, and other club members with a role greater or equal to that of the submitter, or MES administrators and approvers.	APP-3, ABL-1, ABL-5, SaL-1, SaL-2, CAP-1, EXT-2, ACS-1, ACS-2, INT-1, PRI-1, PRI-2, AUD-1, SEI-2, AUL-1, AUL-2, STD-1, STD-2

Legend: Non-Functional Requirement Categories

- **APP / STY:** Look and Feel Requirements (Appearance, Style)
- **EUL / UaP / ABL:** Usability and Humanity Requirements (Ease of Use, Understandability, Accessibility)
- **SaL / PRE / FLT / CAP / EXT / LNG:** Performance Requirements (Speed, Precision, Fault-Tolerance, Capacity, Extensibility, Longevity)
- **ACS / INT / PRI / AUD / IMM:** Security Requirements (Access, Integrity, Privacy, Audit, Immunity)
- **CUL:** Cultural Requirements
- **SEI / AUL / RDL / STD:** Compliance Requirements (Storage/Encryption/Infrastructure, Access/Audit/Logging, Retention/Disposal, Standards)

This matrix ensures that for every function the system performs, the corresponding quality attributes are verified through specific test cases.

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, you 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?

Ease of Use

1. On a scale of 1–5, how easy was it to submit a reimbursement request with all required information and receipt uploads?
2. How intuitive did you find the navigation between different sections of the platform (dashboard, submission form, request status)?
3. Did you encounter any confusing terminology or unclear instructions while using the system?

Efficiency

1. Approximately how long did it take you to complete your first reimbursement submission?
2. Did the system provide adequate feedback during file uploads and form submission processes?
3. Were you able to quickly locate and check the status of your previous reimbursement requests?

Role-Specific Functionality

- **For Reviewers:** How easy was it to view all relevant details needed to approve or reject a reimbursement request?
- **For Club Members:** Did you feel confident that your submission was successful after completing the form?

- **For Admins:** How efficiently could you access and manage reimbursement data across multiple clubs?

Accessibility and Design

1. Were all buttons, links, and interactive elements clearly identifiable and easy to click/tap?
2. Did the color contrast and text sizing make the interface comfortable to read?
3. Were you able to complete all tasks using only your keyboard (without a mouse)?

Error Handling

1. If you encountered any errors, were the error messages clear and helpful in resolving the issue?
2. Did the system prevent you from making mistakes (e.g., uploading invalid file types, missing required fields)?
3. How satisfied were you with the system's response when attempting actions you didn't have permission for?

Overall Satisfaction

1. On a scale of 1–5, how likely are you to recommend this platform to other MES clubs?
2. What feature or aspect of the platform did you find most helpful or well-designed?
3. What improvements would you suggest to make the platform easier to use?

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?

Thomas's Reflection:

1. Something that went well for this deliverable was the way that we were all able to split up the work. We each individually assigned ourselves a section to write but all came together to discuss the overall structure

and flow of the document. We were able to complete the document on time and with good quality.

2. A pain point that was experienced in this deliverable was ensuring that we were able to complete the deliverable in a timely manner while balancing other course work. We weren't really able to solve it for this week due to the number of midterms and assignments from other courses, but we will try to plan better for future deliverables.

3. One knowledge area that I will need to acquire to successfully complete the verification and validation is how to perform dynamic testing for the front end UI framework and how to measure and report database access and API response times. For me personally, I believe this is a skill that will be vital in completing the testing portion of the project but it will also be a useful skill to have in my future career as a software engineer.

4. For both knowledge areas, there is a structured method and a practical method of learning. The structured method involves using online resources, taking courses, or reading documentation to learn the concepts and theory behind the topics. Meanwhile, the practical method involves hands-on experience by actually implementing tests and measuring performance on our project. The structured method can be used by all individuals within the group to build a foundation of understanding for how to perform the tests, while the practical method will be used by me to actually implement the tests and measure performance on our project. I believe that the practical method will be more effective for me as I learn best by trying and learning from my mistakes.

Justin's Reflection:

1. For this deliverable, something that went well was the we took initiative to take on work instead of handing out work deliberatly. This shows that we are all motivated to complete the deliverable and that we trust each other pick up work to share the workload. Its better than how we previously handled work delegation because theres more autonomy and less micromanagement.

2. A pain point was aligning out understanding of the deliverable structure and content. Even though we are relatively on the same page, specifics about the requirements and content were inconsistent amongst group members. This caused isseus when deciding how we wanted to create test cases and what should be included. The main resolution involved having more discussions and clarifying our understanding of the deliverable requirements.

3. To properly complete this project, we needed to learn different possible methods to perform dynamic testing for the front end UI framework and how to measure and report database access and API response times. For me, there was a lack of knowledge on different technologies that would allow me to create automated test cases, which is essential for the testing portion of the project.

4. To acquire the knowledge and skills needed, I want to take more time to find all the possible tools and technologies that can be applied to a situation, compared to just picking from the ones I already know. I can also try to learn from others who have more experience in testing and ask for their recommendations on what tools to use. I will most likely pursue the second option because I believe learning from others is more efficient and effective.

Zhenia's Reflection:

1. What went well was our team's ability to self-organize and take ownership of different sections without extensive coordination. This autonomous approach was more efficient than our previous deliverables and allowed us to complete the work on time.

2. The main challenge was balancing specificity in test cases with our current level of implementation knowledge. I addressed this by focusing on clear inputs and expected outputs while maintaining flexibility for later refinement.

3. I need to develop proficiency in end-to-end testing for our Next.js application, particularly testing workflows that span the frontend, API, and database layers.

4. I'll pursue hands-on experimentation with Cypress or Playwright on our existing codebase, as I learn best through practical application rather than theoretical study.

Michael's Reflection:

1. Similiar to some of our previous deliverables, I think that a lot of the information required to complete the deliverable was already collected before we started working on the deliverable. Although the documents were kind of designed to be completed in this order because of their dependencies on each other, it's good that the content we had previously written was meaningful enough to ease work on this deliverable.

2. There was some uncertainty regarding some of what information is required for each section of the document; the rubric provides a very high-level overview statement, while I personally was lost on some of the acronyms and expected information. I suspect the lectures would have answered some of my questions, but through cross-referencing the pink advice text, past works, and discussing with group members, I think we were able to not spend too much time blocked by a lack of information.

3. For sure a lot of testing knowledge across various subdomains and languages will need to be acquired. For myself, although I have some knowledge about setting up linters and custom scripts, I don't have any formal experience with running comprehensive unit testing suites. Developing the testing and software at the same time, or really any "test-driven" development is a skill I will have to acquire for this project.

4. I think for me, the most practical approach will be to kind of learn the systems as I go through trial and error and online resources, without focusing too much on perfecting my understanding based on ideal theory. There's still a lot of work we will have to do alongside verification and validation, so I anticipate that spending time to go out of my way and learn the testing systems from top to bottom will be impractical. Instead, maybe letting team members that are more experienced plan out the systems, and then contributing through adding regular unit tests, running custom tests, etc, will give me a better intuitive understanding of testing. Another approach might be to communicate more

with my team members and kind of learn from them in a tutor-like relationship. This approach would see me trying to bolster my understanding quickly early on, and then leverage the new knowledge and skills to be able to contribute broadly to the VnV process later on. To be honest, I think maybe I haven't laid two approaches that are selectable: I think that depending on the development situation and my teammates preferences, one of the approaches will naturally be the one I pursue.