System Verification and Validation Plan for MES-ERP

Team #26, Ethical Pals
Sufyan Motala
Rachid Khneisser
Housam Alamour
Omar Muhammad
Taaha Atif

January 17, 2025

Revision History

Date	Version	Notes
Date 1	1.0	Notes
Nov 4, 2024	1.1	Housam: Add parts to section 3 (plan)

[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

List of Figures

N/A

1 Symbols, Abbreviations, and Acronyms

Abbreviation	Description
MES	McMaster Engineering Society
V&V	Verification and Validation
SRS	Software Requirements Specification
UI	User Interface
API	Application Programming Interface
CI/CD	Continuous Integration / Continuous Deployment

This section defines the symbols, abbreviations, and acronyms used throughout the document to ensure clarity and consistency.

This document outlines the Verification and Validation (V&V) Plan for the McMaster Engineering Society Custom Financial Expense Reporting Platform. The V&V plan will ensure that the software performs as expected, meets user requirements, is secure, and reliable. The primary goals of this document are to define the objectives, methods, and scope of the V&V process, covering aspects such as requirements validation, design verification, and system testing.

2 General Information

2.1 Summary

The MES Custom Financial Expense Reporting Platform is a web-based application designed to streamline financial operations for McMaster Engineering Society and its associated student groups. It enables users to submit, review, and track reimbursement requests, create budgets, and manage expenses in a transparent, auditable, and efficient manner. This V&V Plan outlines the verification and validation processes for ensuring the accuracy, functionality, and security of the software.

2.2 Objectives

The primary objective of this V&V Plan is to confirm that the MES Custom Financial Expense Reporting Platform meets its requirements and operates as intended. The plan aims to:

- Build confidence in the software's functional correctness and ensure it meets all outlined requirements.
- Validate usability by demonstrating a clear, intuitive, and user-friendly interface for all key user roles.
- Ensure system security and data integrity, especially regarding sensitive financial data and authorization levels.

The plan does not prioritize nonfunctional aspects such as aesthetic design consistency or performance optimizations beyond standard system requirements, as these are not critical to initial functionality and budget constraints.

2.3 Challenge Level and Extras

1. Useability Testing
   Conduct formal usability testing with a variety of stakeholders (students, financial staff) to refine the interface and ensure a smooth user experience.

2. User Documentation
   Create comprehensive user documentation in the form of written documentation and video tutorials that guides end-users through every step of submitting expenses, reviewing budgets, and navigating the platform.

The challenge level for this project is classified as **General**, aligning with the expectations outlined in the problem statement. The team has chosen to address the critical needs of verification of functionality, security, and usability. No additional extras beyond the required project scope are included in this V&V Plan.

2.4 Relevant Documentation

The following documents are relevant to this V&V plan and provide a basis for verification activities:

- **Software Requirements Specification (SRS)** (Atif et al., 2024d): Defines the functional and nonfunctional requirements of the system, serving as a primary source for validation and test case development.
- **Hazard Analysis**(Atif et al., 2024b): Outlines potential system risks and serves as a guide for security testing and fault tolerance validation.
- **Problem Statement**(Atif et al., 2024c): Provides an overview of the project's purpose, goals, and core functionality requirements, detailing the needs of the McMaster Engineering Society for a streamlined financial reporting platform. It identifies key problems in the current system, justifying the need for an automated, digital solution to enhance efficiency and reliability in financial processes.
- **Development Plan**(Atif et al., 2024a): Outlines the project's development lifecycle, including phases for requirements gathering, design,

implementation, and testing. It establishes milestones, assigns roles, and defines the timeline, while ensuring that the team follows best practices in Agile development, with iterative check-ins for verification and validation of functional requirements.

These documents establish the foundation for testing activities, allowing each verification and validation activity to target specific functional areas, requirements, or potential risk zones.

3 Plan

This section outlines the Verification and Validation (V&V) process for the MES Custom Financial Expense Reporting Platform, which aims to provide reliable financial management for MES and its associated student groups. The plan is structured to cover team roles, verification of the Software Requirements Specification (SRS), design verification, and verification of the V&V plan itself. Each subsection provides an approach tailored to the critical functional and non-functional requirements outlined in the SRS.

3.1 Verification and Validation Team

The V&V team is composed of designated members, each assigned specific responsibilities to ensure thorough verification of functionality, usability, security, and data integrity. The table below describes each team member's role and responsibilities:

Name	Role	Responsibilities
Housam	V&V Lead	Oversees all V&V activities, ensures that testing is aligned with MES requirements, and coordinates communication between team members and stakeholders. Reviews testing documentation for completeness and accuracy.
Rachid	Functional Tester	Manages the testing of all functional requirements, such as reimbursement submission, payment tracking, and budget management. Ensures that each feature meets specified criteria and conducts integration tests for module interactions.
Taaha	Security Tester	Responsible for verifying data security and access controls. Conducts tests to validate role-based access, secure data handling, and compliance with MES privacy policies. Implements static and dynamic security tests.
Sufyan	Usability Tester	Focuses on the user experience by conducting accessibility testing, analyzing interface consistency, and evaluating navigation efficiency. Engages with end-users to gather feedback and adjust usability requirements.

Omar	Performance Tester	Conducts tests to evaluate system performance under various loads and conditions. Responsible for stress testing, measuring response times, and identifying any performance bottlenecks, ensuring the platform remains efficient and responsive even during peak usage.

Each team member will participate in weekly meetings to discuss V&V progress, address issues, and adapt the V&V approach as needed. The V&V Lead will compile findings from each member and provide consolidated feedback to the project supervisor and stakeholders for continuous improvement.

3.2 SRS Verification Plan

The SRS document is critical for establishing a clear and agreed-upon set of requirements for the MES Custom Financial Expense Reporting Platform. Our SRS verification plan will ensure the SRS's accuracy, completeness, and alignment with MES's objectives. The following approaches will be used for verifying the SRS:

- **Peer Review**: Team members and classmates will conduct an initial peer review of the SRS document. Reviewers will assess the clarity and completeness of requirements, noting ambiguities or missing details. Feedback will be documented, and revisions will be made to address identified issues.

- **Stakeholder Review**: The SRS will be presented to MES stakeholders in a structured review meeting. During this meeting, we will walk stakeholders through each requirement, prioritizing critical functionality such as reimbursement processing, budget management, and notification systems. Specific feedback will be sought on:

  - Completeness of functional requirements, such as the need for real-time updates on reimbursement statuses.
  - User accessibility and usability, focusing on ease of submission for student leaders.
  - Compliance with MES financial policies, including secure audit trails and access controls.

- **Supervisor Feedback**: The project supervisor will review the SRS as part of a task-based inspection. The supervisor will be provided with a checklist of essential aspects to verify, including:
  - Consistency in terminology and definitions.
  - Logical flow between sections, ensuring that each requirement aligns with MES's needs.
  - Identification of non-functional requirements related to performance, scalability, and security.

  Feedback from the supervisor will be reviewed, and adjustments will be made to enhance clarity and accuracy.

- **SRS Checklist**: A checklist will be created to ensure all requirements are accounted for. The checklist will include criteria such as:
  - Traceability: Each requirement must map back to a specific use case or MES need.
  - Testability: Each requirement must be written in a way that makes it possible to validate through specific tests.
  - Clarity and conciseness: Requirements should be free from ambiguous language and be concise enough for straightforward implementation.

3.3 Design Verification Plan

To confirm that the design aligns with the SRS and will fulfill MES's functional and non-functional requirements, a structured design verification process will be conducted. The design verification plan includes the following elements:

- **Design Walkthroughs**: The team will conduct regular design walkthroughs, reviewing key components and workflows. During these sessions, each module (e.g., reimbursement submission, audit logging, budget tracking) will be analyzed against the SRS to ensure alignment. Feedback from walkthroughs will be documented, and modifications will be made to address any identified issues.

- **Peer Reviews**: Team members will conduct detailed reviews of each other's design documentation, checking for adherence to design principles, completeness, and alignment with MES's goals. This review will include:

  - Checking that all use cases are addressed in the design.
  - Verifying that interactions between modules support seamless user experience and efficient functionality.
  - Ensuring that security measures are in place for handling sensitive financial data.

- **Checklists for Design Verification**: Each design element will be assessed using a checklist, covering:

  - Consistency with SRS requirements.
  - Scalability and extensibility of the design.
  - Feasibility of implementation within the project timeline and resources.
  - Compliance with MES standards, such as maintaining an intuitive user interface.

  Checklists will help maintain thoroughness and consistency across the team's design review efforts.

- **Supervisor Feedback**: After each design phase, the design documentation will be reviewed by the project supervisor. The supervisor's review will focus on verifying that the design meets project expectations for functionality, scalability, and security. A checklist will be provided to the supervisor to guide the review, focusing on aspects such as:

  - Alignment of the design with SRS requirements.

  - Feasibility of the design for expected usage levels by MES stakeholders.
  - Identification of any areas where design adjustments are needed to meet MES standards.

  Feedback from the supervisor will be integrated into the design documentation.

3.4 Verification and Validation Plan Verification Plan

The V&V plan itself requires verification to ensure that it is both feasible and comprehensive. This verification process will involve the following steps:

- **Peer Review**: Team members will review the V&V plan for logical flow, completeness, and feasibility. The reviewers will check that each component of the V&V plan aligns with the requirements in the SRS and that planned activities are feasible within the project timeline.
- **Mutation Testing**: Selected sections of the V&V plan will undergo mutation testing to identify weaknesses or gaps in test case descriptions. This will help us refine test cases to ensure they cover a wide range of possible inputs and edge cases, making the V&V plan more robust.
- **Checklist for V&V Plan Review**: A checklist will guide the review of the V&V plan, covering:
  - Presence of all required V&V components, such as unit, integration, and system tests.
  - Feasibility of planned activities, given project resources and constraints.
  - Traceability between test cases and SRS requirements, ensuring comprehensive coverage.
  - Alignment of verification techniques with the specific needs of the MES platform.
- **Feedback from Supervisor and Stakeholders**: A meeting will be held with the project supervisor and MES stakeholders to review the V&V plan. During this session, we will gather feedback on:

- Adequacy of planned testing for mission-critical functions (e.g., reimbursement submission, audit logging).
- Planned usability testing to ensure user satisfaction among student groups and administrators.
- Risk mitigation strategies for potential project challenges, such as data security concerns.

The feedback gathered will be used to refine the V&V plan further, ensuring it meets the expectations of all stakeholders.

By implementing the above V&V plan, we aim to build confidence in the MES Custom Financial Expense Reporting Platform's functionality, usability, and security, ensuring it meets the expectations of the MES and its users.

3.5 Implementation Verification Plan

The implementation of the verification plan will be accomplished through a multi faceted approach:

Code Reviews

- All code changes will undergo peer review through GitHub Pull Requests before being merged into the main branch.
- At least one team member has to approve the pull request before it can be merged.
- Code reviews will use a standardized checklist focusing on:
  - Implementation correctness against requirements.
  - Security considerations for financial data handling.
  - Test coverage verification.
  - Error handling completeness.

Test Execution

- System tests as detailed in Section 4, with particular focus on:
  - Functional tests for reimbursement submission (Section 4.1).
  - Security and data integrity tests (Section 4.2).
  - Performance and scalability tests (Section 4.2).
- Unit tests as outlined in Section 5, covering:
  - Individual module testing.
  - Integration points between components.
  - Edge cases and error conditions.

Static Analysis

- ESLint will be used for static code analysis.
- Type checking through TypeScript's compiler.
- SonarQube will be used for code quality and security analysis.

Review Sessions

- Weekly review sessions will be held with the team to discuss progress, issues, and verification results. This meeting will additionally be held with the project supervisor on a monthly basis.
- Review sessions will include a demonstration of new features and verification results.
- Occasionally review sessions will include a demonstration of the system to stakeholders.
- The checkpoint meetings with the TA will also serve as a review session.
- The final presentation will serve as a comprehensive code walkthrough.

3.6 Automated Testing and Verification Tools

The following tools will be utilized for automated testing and verification, aligned with our development technology stack:

Testing Frameworks

- Jest: Primary testing framework for unit and integration testing.
- React Testing Library: Component testing focusing on user interactions and UI behavior.
- Cypress: End-to-end testing for user flows and system integration.

Code Quality Tools

- ESLint: Configured with specific rules for TypeScript and Next.js development.
- Prettier: Automated code formatting on commit to maintain consistent style.
- TypeScript: Static type checking with strict mode enabled.
- Git pre-commit hooks: Automated formatting and linting checks before commits are allowed.

Continuous Integration

- GitHub Actions for:
  - Automated test execution on each commit.
  - Code quality checks through ESLint.
  - Type checking through TypeScript compiler.
  - Automated deployment to staging environment for pull requests.
- Test coverage reporting through Jest with:
  - Minimum 90% coverage requirement as specified in the Development Plan.
  - Coverage reports generated for each pull request.
  - Blocking of merges if coverage thresholds aren't met.

Performance Monitoring

- Lighthouse: Performance, accessibility, and best practices monitoring.
- Chrome DevTools: Memory usage and rendering performance analysis.
- Next.js Analytics: Built-in performance monitoring for production deployments.

Security Testing

- npm audit: Regular dependency vulnerability scanning.
- Automated vulnerability scanning integrated into CI pipeline.

All tools have been selected to integrate seamlessly with our Next.js and TypeScript stack while supporting our coding standards and quality requirements. The automation of these tools through our CI/CD pipeline ensures consistent verification across all stages of development.

3.7 Software Validation Plan

The validation plan ensures the software meets the needs of the McMaster Engineering Society and its 60 student groups through a comprehensive approach:

External Data Validation

- Historical reimbursement data from the MES's previous Excel-based system will be used to validate:
  - Accuracy of financial calculations.
  - Correct budget categorization.
  - Processing time improvements.
- Sample receipts provided by the MES will be used to validate the receipt processing system.

Stakeholder Requirements Review

- Task-based inspection sessions with MES VP Finance to verify SRS completeness:
  - Walkthrough of common reimbursement scenarios.
  - Review of financial approval workflows.
  - Verification of audit requirements.
- Requirements validation through prototype demonstrations:
  - Interactive sessions with student group leaders.
  - Feedback collection on user interface design.
  - Verification of notification preferences.

Rev 0 Demo Validation

- Scheduled demonstration with external supervisor (MES VP Finance).
- Focus areas for supervisor feedback:
  - Validation of implemented requirements from Section 9.1 of the SRS.
  - Verification of financial workflow accuracy.
  - Assessment of security measures.
  - User interface evaluation.
- This feedback will be documented and incorporated into subsequent development iterations.

User Acceptance Testing

- Beta testing program with selected student groups:
  - Testing with real reimbursement scenarios.
  - Validation of notification systems.
  - Performance under typical usage patterns.
- Specific validation metrics:

  - Reimbursement request completion time.
  - Budget categorization accuracy (target: 90%).
  - System response times under load.
  - User error rates during submission process.

Requirements Validation

- Regular requirement review sessions to ensure:
  - Alignment with MES financial policies.
  - Compliance with security requirements.
  - Meeting of performance targets.
- Validation of specific SRS requirements:
  - Custom budget creation functionality.
  - Real-time ledger tracking.
  - Multi-level approval workflows.
  - Automated notifications.

All validation activities will be documented and tracked through GitHub. Feedback will be incorporated into the development cycle. There will be particular attention to the core requirements of reducing wait times and preventing loss of reimbursement requests. Any changes to requirements identified through validation will be reviewed and updated in the SRS document accordingly.

4 System Tests

4.1 Tests for Functional Requirements

Test 1: Submission Confirmation

- **Test ID**: test-id1
- **Control**: Automatic

- **Initial State**: User logged in, valid reimbursement document available
- **Input**: A completed reimbursement form with attached receipt
- **Output**: Confirmation message and ledger update
- **Test Case Derivation**: Ensures the form submission logic updates the ledger correctly, as described in the SRS
- **How Test Will Be Performed**: Submit the form and verify through the UI and database that the ledger is updated

Test 2: Approval Workflow

- **Test ID**: test-id2
- **Control**: Manual
- **Initial State**: Admin logged in, pending request available
- **Input**: An admin approval action on a pending reimbursement request
- **Output**: Status changes to approved, and notifications are sent
- **Test Case Derivation**: Derived from multi-level approval requirements to ensure workflow consistency
- **How Test Will Be Performed**: Approve the request and check the status change, notification delivery, and database integrity

Test 3: Expense Addition

- **Test ID**: test-id3
- **Control**: Automatic
- **Initial State**: Initialized budget with no transactions
- **Input**: An expense entry into the system
- **Output**: Budget overview reflects the new expense
- **Test Case Derivation**: Ensures real-time updates to the budget, as required by the SRS

- **How Test Will Be Performed**: Add an expense and verify updates in the UI and database

Test 4: Budget Creation and Categorization

- **Test ID**: test-id4
- **Control**: Automatic
- **Initial State**: User logged in with access to budget management tools
- **Input**: Budget creation request specifying category and amount
- **Output**: Confirmation of successful budget creation, correctly categorized under specified category
- **Test Case Derivation**: Ensures budget creation and categorization align with SRS requirements for custom budget management
- **How Test Will Be Performed**: Create budget, categorize it, and verify correct display and categorization in dashboard

Test 5: Notification on Reimbursement Status Change

- **Test ID**: test-id5
- **Control**: Automatic
- **Initial State**: User has a pending reimbursement request
- **Input**: Admin updates status of reimbursement request to "approved"
- **Output**: Notification is sent to user confirming approval, with update in dashboard
- **Test Case Derivation**: Confirms notification system reliability as per SRS notification requirements
- **How Test Will Be Performed**: Change status in admin panel, observe notification and dashboard update for the user

Test 6: Audit Trail Logging

- **Test ID**: test-id6

- **Control**: Automatic
- **Initial State**: System is idle with no recent actions in audit log
- **Input**: User performs actions, such as budget creation and expense submission
- **Output**: Each action is logged in the audit trail with accurate timestamps and user IDs
- **Test Case Derivation**: Ensures all actions are traceable for compliance, as per SRS audit requirements
- **How Test Will Be Performed**: Perform actions and verify entries in audit log

4.2 Tests for Nonfunctional Requirements

Test 7: Response Time Under Load

- **Test ID**: test-id7
- **Type**: Dynamic
- **Initial State**: System with minimal load
- **Input/Condition**: Simultaneous budget submissions by multiple users
- **Output/Result**: Response time remains within acceptable limits, per performance specifications
- **Test Case Derivation**: Verifies system's performance under high load, per nonfunctional requirements for speed and latency
- **How Test Will Be Performed**: Use simulated users to submit requests, monitor response times

Test 8: Security and Access Control

- **Test ID**: test-id8
- **Control**: Manual

- **Initial State**: Unauthorized user attempting access
- **Input**: Unauthorized access attempt to restricted sections
- **Output**: Access denied, with secure handling of unauthorized access attempt
- **Test Case Derivation**: Confirms that unauthorized access is prevented as per security requirements in SRS
- **How Test Will Be Performed**: Attempt access and verify access denial and logging of the event

Test 9: Load Handling

- **Test ID**: test-id9
- **Type**: Dynamic
- **Initial State**: User dashboard loaded with multiple active sessions
- **Input/Condition**: Simultaneous form submissions from multiple users
- **Output/Result**: System responds within acceptable limits without degradation in performance
- **Test Case Derivation**: Verifies system's load handling capacity to ensure scalability, per nonfunctional requirements
- **How Test Will Be Performed**: Manually simulate multiple users by having several people submit forms at the same time from different devices or browsers. Each user will fill out and submit a form concurrently while monitoring the system's response time, performance, and any issues that arise, such as slowdowns or errors.

Test 10: Data Integrity and Security

- **Test ID**: test-id10
- **Control**: Automatic
- **Initial State**: Data integrity checks enabled
- **Input**: Multiple transactions and database updates

- **Output**: Data remains consistent, secure, and unaltered without authorization
- **Test Case Derivation**: Ensures data integrity and protection against unauthorized changes as per security and data integrity requirements
- **How Test Will Be Performed**: Perform series of database operations and validate data consistency and security logs

4.3 Traceability Between Test Cases and Requirements (above)

Requirement	**Test Case(s)**
Reimbursement Submission	test-id1, test-id2
Budget Management	test-id3, test-id4
Notification System	test-id5
Audit Logging	test-id6
Performance Under Load	test-id7, test-id9
Security and Access Control	test-id8
Data Integrity and Security	test-id10

Table 1: Traceability Between Test Cases and Requirements

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

Taaha Atif, Omar Muhammad, Housam Alamour, Sufyan Motala, and Rachid Khneisser. System requirements specification. https://github.com/Housam2020/MES-ERP/blob/main/docs/DevelopmentPlan/DevelopmentPlan.pdf, 2024a.

Taaha Atif, Omar Muhammad, Housam Alamour, Sufyan Motala, and Rachid Khneisser. System requirements specification. https://github.com/Housam2020/MES-ERP/blob/main/docs/HazardAnalysis/HazardAnalysis.pdf, 2024b.

Taaha Atif, Omar Muhammad, Housam Alamour, Sufyan Motala, and Rachid Khneisser. System requirements specification. https://github.com/Housam2020/MES-ERP/blob/main/docs/ProblemStatementAndGoals/ProblemStatement.pdf, 2024c.

Taaha Atif, Omar Muhammad, Housam Alamour, Sufyan Motala, and Rachid Khneisser. System requirements specification. https://github.com/Housam2020/MES-ERP/blob/main/docs/SRS/SRS.pdf, 2024d.

6 Appendix

N/A

6.1 Symbolic Parameters

N/A

6.2 Usability Survey Questions?

N/A

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?
   Sufyan: What went well for me this assignment was getting github actions to build the pdf on commit. I also enjoyed learning about all the tooling we could use for the project to ensure code quality, test coverage and security.
   Taaha What went well while writing this deliverable was the amount of support I had when asking teammates questions. Was extremely helpful whenever I came across an issue I was unsure of.
   Omar: What went well was the clarity of this document, we understand our project so we were able to devise this document with little to no questions asked to the TA.
   Housam: One aspect that went well for me was refining the plan structure to align with the project's goals and requirements. Working on the document also helped solidify my understanding of the V&V process and how it integrates into the larger project goals.
   Rachid: What went well for me this deliverable was that we discussed how we were going to work on this very early and this allowed me to spend more time thinking and working on this deliverable

2. What pain points did you experience during this deliverable, and how

did you resolve them?
Sufyan: The pain points I experienced were getting the github actions to work properly. I had to do a lot of research and trial and error to get it to work. I also had some issues with the formatting of the document. I resolved these issues by asking for help from my team members.
Taaha The pain points I experienced were mostly related to formatting the document correctly. In order to resolve this, I got help from teammates and looked online.
Omar: I got really sick during this deliverable which slowed down my progress a lot, in addition, it was harder to manage this deliverable since I had a bunch of other deliverables due near the same time.
Housam: The main challenge I faced was balancing detail with conciseness, as I wanted to cover each point in the plan without making it overly lengthy. At times, I felt I was going into too much detail, so I reviewed my work multiple times to find a clear and concise way to present information. Discussing with teammates helped refine certain sections and focus on essential content.
Rachid: This deliverable was extremely clear and as a result, I experienced no pain points.

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.

   What knowledge and skills will the team collectively need to acquire to successfully complete the verification and validation of your project?

   Key skills needed for V&V include:

   - **Dynamic Testing:** Using Jest and Cypress for unit and integration testing.
   - **Static Analysis:** Using ESLint and SonarQube to identify code issues.
   - **CI/CD Setup:** Configuring GitHub Actions to automate testing and deployment.

- **Security Testing:** Implementing npm audit for vulnerability scanning.
- **Usability Testing:** Conducting accessibility and usability tests using Lighthouse.

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?

    - **Dynamic Testing:** Team tutorials on Jest/Cypress and applying tests in the project, led by Housam.
    - **Static Analysis:** Reviewing ESLint/SonarQube documentation and integrating tools during development, led by Omar.
    - **CI/CD Setup:** Using GitHub documentation and incremental setup of GitHub Actions, led by Sufyan.
    - **Security Testing:** Researching secure coding and using npm audit, led by Taaha.
    - **Usability Testing:** Exploring Lighthouse and gathering user feedback, coordinated by Rachid.