System Verification and Validation Plan for McMaster Engineering Society Custom Financial Expense Reporting Platform

Team #12, Reimbursement Rangers
Adam Podolak
Evan Sturmey
Christian Petricca
Austin Bennett
Jacob Kish

January 10, 2025

Revision History

Date	Version	Notes
11/4/2024	0	All sections, including reflections
12/29/2024	0	Syntax Fixes
1/10/2025	0	Explicit non-functional checklists

Contents

List of Tables

1 Symbols, Abbreviations, and Acronyms

symbol	description
HTTP	Hypertext Transfer Protocol
POST	HTTP request method to create new resource
PUT	HTTP request method to update or create new resource
GET	HTTP request method to retrieve data
PATCH	HTTP request method to update part of existing data
JSON	JavaScript Object Notation
Postman	Tool for developing and testing APIs
JPEG	Joint Photographic Experts Group (image file format)
PNG	Portable Network Graphic (image file format)
PDF	Portable Document Format (file format)
UI	User Interface
DB	Database
API	Application Programming Interface
SRS	Software Requirements Specification (Document)

This document provides a structured Verification and Validation (V&V) plan for the McMaster Engineering Society (MES) financial platform project. The purpose of this V&V plan is to outline a systematic approach to evaluate the platform's reliability, usability, and security. This document includes a summary of the software, objectives for testing, identified out-of-scope areas, and relevant references to the project's technical documents. Following this introduction, each section will elaborate on key elements necessary to achieve confidence in the platform's correctness and suitability for MES's needs.

2 General Information

2.1 Summary

The MES financial platform is primarily a CRUD (Create, Read, Update, Delete) application tailored to streamline accounting and intake responsibilities for MES. At its core, the software is responsible for managing and processing data related to reimbursement requests and integrates with external data sources for accounting functions. Key features include:

- **Accounting Integration:** Seamless integration with external data sources for transaction records, making financial management more efficient.
- **Data Processing:** Handling and tracking reimbursement requests from initiation through completion.
- **User Management:** Supports various user roles within MES to facilitate secure access and control over financial data.

The platform is designed to serve as a centralized tool for managing MES financial tasks, reducing the manual load on accounting personnel and ensuring reliable financial record-keeping.

2.2 Objectives

The primary objectives of this V&V process are to:

- **Build Confidence in Software Correctness:** Verify that the software functions as intended for all primary use cases, reducing errors in financial and intake records.

- **Demonstrate Adequate Usability:** Ensure the platform is intuitive, easy to adopt, and efficiently serves MES's accounting and reimbursement needs with minimal user training.
- **Establish Data Security:** Validate that the platform secures sensitive financial data, instilling confidence among users regarding data safety and privacy.
- **Ensure Maintainability:** Confirm that the codebase and system are designed to be easily understood, extended, and maintained by future MES teams or other developers.

2.3 Out-of-Scope Objectives

- **Performance Optimization:** While the platform's speed is important, performance tuning is considered a secondary priority for this V&V plan, as it does not impact the primary functionalities.
- **Reliability and Robustness Testing:** Long-term reliability under heavy load is out of scope, given MES's anticipated user volume is low. Robustness, including extensive error handling for edge cases, is beyond the available resources.
- **Scalability:** The platform is intended for MES's internal use, so high scalability to support a large user base or high transaction volume is not prioritized.
- **Portability and Reusability:** Adaptation for other organizations or platforms is not included, as this project is specifically tailored for MES.

In this project, the team recognizes the constraints on time and resources. Thus, while striving to meet MES's immediate requirements, this V&V plan will also ensure the platform lays a robust foundation for potential future improvements.

2.4 Challenge Level and Extras

This project is classified as a general-level project. Given the critical role of user adoption and ease of use, the team has opted to include **user doc-**

umentation as an extra deliverable. This documentation will provide essential guidance to new users, helping to ease the learning curve and ensure sustained adoption of the platform by MES personnel.

2.5 Relevant Documentation

The following project documents will inform the V&V process, providing necessary references for testing and validation efforts:

- **Software Requirements Specification (SRS):** Details the specific functional and non-functional requirements for the platform, serving as the baseline for validation efforts.
- **Software Design Document (SD):** Provides a comprehensive view of the platform's architecture and design decisions. This will help verify that the implemented software aligns with the intended structure and workflow.
- **Module Guide (MG):** Describes the organization of the software into modules, assisting in modular testing and helping developers isolate and validate individual components.
- **Module Interface Specification (MIS):** Defines the interface requirements for each module, supporting integration testing to confirm seamless interaction between components.

These documents collectively support the V&V process by specifying expected behaviors, design constraints, and modular structures, ensuring that the platform aligns with MES's goals.

3 Plan

This section outlines the roadmap for the Verification and Validation (V&V) process to ensure the reimbursement platform adheres to user requirements and system specifications. The subsections detail the responsibilities of the V&V team, verification plans for the SRS, design, and overall V&V plan, as well as the tools and approaches used.

3.1 Verification and Validation Team

The V&V team comprises the following members, each with designated roles to facilitate the V&V process:

3.2 SRS Verification Plan

The verification of the Software Requirements Specification (SRS) will include:

- **Peer Reviews**: Team members will perform thorough reviews to identify ambiguities, inconsistencies, and missing requirements.
- **Feedback Meetings**: Scheduled meetings with stakeholders will validate that the documented requirements align with their expectations.
- **Checklist-Based Inspections**: The team will use predefined checklists to verify that the SRS is complete, accurate, and adheres to industry standards.

These activities will be documented, and all feedback will be tracked using an issue tracker for consistent updates and resolution.

Checklist for SRS Verification Plan (3.2)

- **Completeness Check**
  - Verify that all user requirements are documented.
  - Confirm that all functional and non-functional requirements are included.
  - Ensure that all relevant use cases and user stories are covered.
- **Consistency Review**
  - Check for consistency in terminology and descriptions throughout the document.
  - Ensure that there are no conflicting requirements.
- **Clarity and Ambiguity**

Role	**Responsibilities**
Project Supervisor: **Adam Podolak**	Provides oversight and final approval for all V&V activities, ensuring that project requirements align with MES standards.
Quality Assurance Lead: **Evan Sturmey**	Conducts regular reviews and quality checks for documentation and code. Oversees test planning and testing processes.
Test Engineer: **Austin Bennett**	Responsible for writing and executing test cases, managing test data, and reporting defects during verification.
Documentation Specialist: **Christian Petricca**	Prepares detailed documentation of the V&V processes and results, and ensures all records are up to date and accessible for review.
Developer: **Jacob Kish**	Assists in integration testing and ensures that features are implemented according to specifications.

Table 1: Verification and Validation Team Roles and Responsibilities

- Identify any vague or ambiguous language.
- Ensure that all requirements are specific and measurable.

- **Traceability**
  - Confirm that each requirement can be traced back to a user need or stakeholder request.
  - Validate that the requirements can be mapped forward to the design and testing phases.
- **Format and Compliance**
  - Verify that the document follows the required format and style.
  - Ensure adherence to project documentation standards.
- **Stakeholder Review**
  - Include stakeholder feedback and address any concerns raised during review sessions.

3.3 Design Verification Plan

Verification of the system design will ensure alignment with the approved SRS. The following steps will be taken:

- **Design Reviews**: Formal reviews will be conducted with peers and the project supervisor to evaluate design decisions, data flow, and overall system architecture.
- **Checklist Inspections**: The design will be checked against a predefined checklist to verify that areas such as security, data integration, and scalability are thoroughly addressed.
- **Classmate Feedback**: External reviews by classmates will provide an unbiased assessment of the design's robustness and potential areas of improvement.

These reviews will be documented, and necessary modifications will be made to strengthen the design.

Checklist for Design Verification Plan (3.3)

- **Design Document Completeness**
  - Ensure that all components of the system are detailed in the design document.
  - Confirm the inclusion of diagrams (e.g., system architecture, data flow).
- **Alignment with SRS**
  - Verify that the design adheres to the requirements outlined in the SRS.
  - Check for traceability between design elements and SRS requirements.
- **Scalability**
  - Review the design for future scalability and possible extensions.
  - Confirm that the design supports anticipated system loads and additional features.
- **Security Considerations**
  - Validate that security measures are incorporated into the design.
  - Ensure that access controls and data protection protocols are part of the architecture.
- **Peer Review and Feedback**
  - Conduct a formal review with peers and collect feedback.
  - Address any concerns and implement necessary changes based on feedback.
- **Design Consistency**
  - Check that consistent design practices and naming conventions are used throughout.
  - Validate that all design components are coherent and properly integrated.

3.4 Verification and Validation Plan Verification Plan

The verification and validation plan itself is a crucial artifact that must be verified to ensure it meets the project requirements. The plan will undergo structured reviews and mutation testing for thorough analysis. This review will be conducted by team members and peers to ensure the plan's integrity and applicability.

- Conduct a peer review of the V&V plan, focusing on coverage, accuracy, and relevance.
- Utilize a checklist to verify that the plan includes all necessary components, such as strategies, methods, and validation steps.
- Implement mutation testing to identify the robustness of the validation methods.

Checklist for Verification and Validation Plan Verification Plan (3.4)

- **Plan Completeness**
  - Verify that all elements of the V&V process are documented in the plan.
  - Confirm that the plan outlines both verification and validation strategies.
- **Clarity of Methodology**
  - Ensure that the V&V methods (e.g., peer reviews, test cases, mutation testing) are clearly described.
  - Check that the plan includes details on how reviews will be conducted.
- **Checklist Inclusion**
  - Confirm the presence of checklists for reviewing the SRS, design, and other artifacts.
  - Ensure that the checklists cover relevant aspects and criteria for verification.

- **Feedback Mechanism**
  - Verify that there is a defined process for collecting and addressing feedback.
  - Check that the plan specifies how issues will be tracked and resolved.
- **Alignment with Project Goals**
  - Validate that the V&V plan aligns with overall project timelines and goals.
  - Ensure that resources are adequately allocated for V&V activities.
- **Review by Team Members**
  - Conduct a peer review of the V&V plan to ensure that it meets the project standards.
  - Address any feedback provided by team members or external reviewers.

3.5 Implementation Verification Plan

The implementation phase will involve comprehensive testing and code walkthroughs to verify that the system aligns with the specified requirements.

- Unit testing will be conducted for individual modules to verify functionality.
- Static code analysis tools will be used to check for coding standards and vulnerabilities.
- Code inspections and peer reviews will be carried out to ensure proper implementation and adherence to best practices.

The final class presentation can be utilized as a code walkthrough to further validate the implementation.

3.6 Automated Testing and Verification Tools

To ensure the robustness of the codebase, various automated testing and verification tools will be employed:

- **Unit Testing Frameworks:** Jest and Mocha will be used for writing and running unit tests.
- **Continuous Integration (CI):** GitHub Actions will be used for CI to automate the testing process.
- **Static Analysis:** ESLint and SonarQube for analyzing code quality and detecting potential issues.
- **Code Coverage:** Istanbul will be used to generate code coverage reports to summarize tested and untested portions.

These tools will help ensure code reliability and maintainability throughout the development process.

3.7 Software Validation Plan

Validation will ensure that the software meets the stakeholders' expectations and requirements. The following approaches will be used:

- Conduct review sessions with stakeholders to validate that the requirements have been correctly implemented.
- Organize a Rev 0 demo to present the software to the supervisor and gather feedback for potential improvements.
- Utilize user testing to simulate real-world interactions and identify any usability issues.

The feedback from these activities will be documented and used to refine the system.

4 System Tests

The following sections will cover system tests for functional and non-functional requirements outlined in the SRS.

4.1 Tests for Functional Requirements

4.1.1 Tests for Functional Requirements 9.1.1 - Reimbursement and Payment Request Feature

The subsections below cover tests relating to functional requirements for the reimbursement request feature. The tests below cover requirements FR1.1 - FR1.9 in section 9.1.1 of the SRS.

FR1.1 - Successful Submissions of reimbursement request

1. **FR1.1-TC1**

   Control: Manual

   Initial State: User is authenticated and logged into the system. User has navigated to the "Submit Reimbursement Request" page.

   Inputs:

   - Receipt images (JPEG or PNG format)
   - Supporting documents (PDFs)
   - Payment details - payment card number, expiry date, access code (through Stipe API)
   - Clicks "Submit" button

   Output:

   - System securely stores the submission in the database
   - Confirmation message is displayed - "Your reimbursement request has been submitted"
   - Request appears in the user's "My Submitted Requests" page
   - Request status reads "Submitted"

   Test Case Derivation: This test case verifies that the users can successfully submit a reimbursement request as per FR1.1 in the SRS. Successfully submitted requests will be given status "Submitted".

   How test will be performed:

- Log in as a user
- Navigate to ”Submit Reimbursement Request” page
- Provide specified sample inputs above
- Click ”Submit”
- Verify confirmation message is displayed
- Navigate to ”My Submitted Requests” page and confirm the newly created request appears with all the same information
- Postman is a tool that can be used to verify the POST/PUT and GET requests sent to the API and database. HTTP requests can also be verified using a web browser’s console to check response JSON objects

2. **FR1.1-TC2**

   Control: Manual

   Initial State: User is authenticated and logged into the system. User has navigated to the ”Submit Reimbursement Request” page.

   Input:

   - Receipt images (NOT in JPEG or PNG fomrat)
   - Supporting documents (PDFs)
   - Payment details - payment card number, expiry date, access code (through Stipe API)
   - Clicks ”Submit” button

   Output:

   - System prevents submission
   - Error message is displayed - ”Improper image format. Images must be JPEG or PNG)

   Test Case Derivation: This test case enforces proper input of required fields to allow for the successful submission of a reimbursement request as per FR1.1 in the SRS. Improper input file type should result in the system not allowing a user to submit their request.

   How test will be performed:

- Log in as a user
- Navigate to "Submit Reimbursement Request" page
- Provide specified sample inputs except receipt image provide an incorrect file type (.pdf, .tiff, etc.)
- Click "Submit"
- Verify the error message is displayed within the the UI
- Verify request is not successfully submitted into the database using Postman

FR1.2 - Admin Review and Approval of Submitted Request Tests

1. **FR1.2-TC1** Control: Manual

   Initial State:

   - A reimbursement request has been submitted and has "Submitted" status
   - An admin user is logged into the system

   Inputs:

   - Admin user navigates to the "Pending Requests" page
   - Admin user selects reimbursement request and reviews submission
   - Clicks "Approve" button

   Output:

   - Request status changed to "Approved". This change is reflected in both the UI and the DB
   - System records the approval action in the audit log
   - The user who submitted the reimbursement request receives a notification about the approval (Covered in FR1.5 tests)

   Test Case Derivation: This test case verifies that admins can review and approve submitted reimbursement requests as per FR1.2 in the SRS

   How test will be performed:

- Login as admin user
- Navigate to "Pending Requests" and select a reimbursement request
- Click "Approve"
- Confirm request status is updated to "Approved". Use Postman to validate PATCH request. Check if status changes in the UI.
- Check audit log to verify approval action was recorded, along with which admin user completed the approval.

2. **FR1.2-TC2**

   Control: Manual

   Initial State:

   - A reimbursement request has been submitted and has "Submitted" status
   - An admin user is logged into the system

   Inputs:

   - Admin user navigates to the "Pending Requests" page
   - Admin user selects reimbursement request and reviews submission
   - Clicks "Reject" button
   - Reason for rejection: "Receipt missing"

   Output:

   - Request status changed to "Rejected". This change is reflected in both the UI and the DB
   - System records the approval action in the audit log
   - The user who submitted the reimbursement request receives a notification about the rejection (Covered in FR1.5 tests)
   - reason for rejection is stored with the associated request

   Test Case Derivation: This test case verifies that admins can review and reject submitted reimbursement requests as per FR1.2 in the SRS

   How test will be performed:

- Login as admin user
- Navigate to "Pending Requests" and select a reimbursement request
- Click "Reject"
- Confirm request status is updated to "Rejected". Use Postman to validate PATCH request. Check if status changes in the UI.
- Check audit log to verify rejection action was recorded

FR1.3 - Tests for Editing Submitted Requests

1. **FR1.3-TC1**

   Control: Manual

   Initial State:

   - User is authenticated and logged into the system
   - User's submitted reimbursement request has status "Requires Attention" because of an unclear image of a purchase receipt

   Inputs:

   - Click "Edit" on the reimbursement request
   - Change image of receipt
   - Click "Save"
   - Click "Re-submit"

   Output:

   - Request status changed back to "Submitted"
   - Action is recorded in the audit log

   Test Case Derivation: This test verifies users and admins can edit a submitted reimbursement request as per FR1.3 in the SRS

   How test will be performed:

   - Log in as a user
   - Navigate to "My Requests"

- Click "Edit' on the reimbursement request that has status "Requires Attention"
- Make the necessary edits
- Click "Save" and "Re-submit"
- Confirm request status has been changed back to "Submitted" in the UI on the "My Requests" page
- Verify using Postman that the appropriate PATCH requests were successfully made (edited fields and status are reflected in the DB)
- Verify the edit action is recorded in the audit log in the DB

FR1.4 - Status Tracking Test

1. **FR1.4-TC1**

   Control: Manual

   Initial State:

   - A submitted reimbursement request has status "Submitted"
   - Logged in user is an admin

   Inputs:

   - Admin selects reimbursement requests
   - Click "Set status"
   - Select "Under Review"

   Output:

   - System updates and displays the current status of reimbursement request to "Under Review"
   - System reflects this change of status in DB

   Test Case Derivation: This test verifies that the system can track the status of each request as per FR1.4 in the SRS. If an admin sets a request's status to "Under Review" the status of the request should be changed to "Under Review".

   How test will be performed:

- Logged in as an admin, change a request's status from "Submitted" to "Under Review"
- Verify that the status is changed to "Under Review" in the UI
- Verify using Postman that the correct PATCH request is made and the status is reflected in the DB
- Repeat the above actions with various status values (Approved, Rejected).

FR1.5 - Notification Tests

1. **FR1.5-TC1**

   Control: Manual

   Initial State:

   - An admin is authenticated and logged into the system
   - A user's request has status "Submitted"
   - The user has a valid email and/or phone number associated with their account

   Inputs:

   - Admin changes the user's request from "Submitted" to "Approved"

   Output:

   - User receives an email or SMS notification updating them that their request status has changed from "Submitted" to "Approved"

   Test Case Derivation: This test verifies the system can notify users of any changes to their reimbursement request status as per FR1.5 in the SRS. If a request's status is changed to "Approved" the user should be notified that their request's status has been changed to "Approved"

   How test will be performed:

   - Log in as an admin
   - Change a user's request status to "Approved"
   - Check the user's email inbox or SMS message

- Verify that the notification contains the correct information. This includes which request's status was changed, and what the status was changed to.

FR1.6 - Data Storage and Categorization Tests

1. **FR1.6-TC1**

   Control: Manual

   Initial State:

   - A user is authenticated and logged into the system
   - User has navigated to the "Submit Reimbursement Request" page

   Inputs:

   - User selects "Office Supplies" as budget category from predefined list
   - User submits the request

   Output:

   - Request information is stored in the DB
   - Request object is given budget category attribute "Office supplies"

   Test Case Derivation: This test verifies that the system will store and categorize reimbursement requests based on budget categories for ledger tracking as per FR1.6 in the SRS.

   How test will be performed:

   - Logged in as a user, submit a reimbursement request with the budget category "Office Supplies"
   - Verify using Postman that the correct budget category was assigned to the request and reflected in the DB

FR1.7 - Access Control Tests

1. **FR1.7-TC1**

   Control: Manual

   Initial State:

   - User X and User Y have submitted reimbursement requests.

   Inputs:

   - User X attempts to access User Y's request details

   Output:

   - The system denies access and displays an error message - "You do not have access"

   Test Case Derivation: This verifies that the system can restrict access to view sensitive financial information submitted by users as per FR1.7 in the SRS. General users should only have access to view requests that they have submitted.

   How test will be performed:

   - Log in as User Y and navigate to "My Requests", and copy the URL.
   - Log in as User X and attempt to access the URL/page of User Y's requests page
   - Verify that access is denied and an error message is returned
   - Verify that User X can view their own requests

2. **FR1.7-TC2**

   Control: Manual

   Initial State:

   - User X and User Y have submitted reimbursement requests.

   Inputs:

   - Admin navigate to "Pending Requests"

Output:

- The system responds with all submitted reimbursement requests. Admin can view User X's requests and User Y's requests

Test Case Derivation: This verifies that the system can restrict access to view sensitive financial information submitted by users as per FR1.7 in the SRS. Admin users should have access to view all submitted requests.

How test will be performed:

- Log in as Admin and navigate to "Pending Requests" page
- Verify that all requests displayed in the UI match all requests stored in the DB
- Verify that User X's and User Y's requests can be selected, and viewed, and actions can be performed on them

FR1.8 - Admin Collaboration Tests

1. **FR1.8-TC1**

   Control: Manual

   Initial State:

   - A reimbursement request has "Submitted" Status
   - Admin 1 and Admin 2 are authenticated and logged into the system

   Inputs:

   - Both Admins access the same request
   - Admin 1 changes the status of the request to "Approved"
   - Admin 2 changes the status of the request to "Rejected"

   Output:

   - The system flags the request and sets its status to "Needs Review"
   - The system tracks both actions in the audit log

- The system adds a note associated with the requests - "Mismatched status update, please review and update"
- The system does not notify the user of this status change

Test Case Derivation: This test case ensures that multiple admins can collaborate and review submitted requests as per FR1.8 in the SRS. If two different admins set a request's status to conflicting values, that request must be reviewed and a singular status value must be selected.

How test will be performed:

- Log in as Admin 1 and Admin 2, access the same request, and make the respective status changes
- Verify the system changes the request status to "Needs Review" in the UI and in the DB
- Verify the system generated an associated 'note' field for the request that reads "Mismatched status update, please review and update." This field should be visible in the UI and stored in the DB
- Verify the system did not send an update notification to the user by either accessing the user's inbox or using Postman to check that no API call was made

FR1.9 - Action Tracking Tests

1. **FR1.9-TC1**

   Control: Manual

   Initial State:

   - A reimbursement request has been submitted and has status "Submitted"

   Inputs:

   - An admin changes the status of the request from "Submitted" to "Approved"

   Output:

- The system records the action with details: request ID, action type (status change), user (name, email), timestamp

Test Case Derivation: This test verifies that the system can track and store all actions performed concerning a single reimbursement request as per FR1.9 in the SRS.

How test will be performed:

- As a user, submit a request
- As an admin, change the status of the request to "Approved"
- Verify that the system stored the correct action details under the correct table in the DB.

4.1.2 Tests for Functional Requirements 9.1.2 - Custom Invoice Generation Feature

Note: These system tests have been omitted for this deliverable, as this feature may become a stretch goal depending on the progression of the project. If time permits, the system tests for this section will be included in the Revision 1 VnV Report

4.1.3 Tests for Functional Requirements 9.1.3 - General Functional Requirements

Note: the below tests for requirements FR3.1-FR3.3 are subject to change as a concrete authentication method has not yet been established. This will likely be established in the detailed design document. The system will either use a classic authentication method (username and password login/signup), or authentication using tokens sent to the user's McMaster email address.

FR3.1 - Secure Storage of User Account Information

1. **FR3.1-TC1**

   Control: Manual

   Initial State:

   - The system is operational with at least one registered user.

- The database contains user account information, including full name, email, password, and phone number

Inputs:

- legitimate user's credentials for log in

Output:

- Unauthorized access attempts are denied and an error message is displayed
- Passwords are securely stored (e.g., hashed and salted) and cannot be retrieved in plaintext
- User account information is not exposed to unauthorized users

Test Case Derivation: This test verifies that the system securely stores account information and prevents unauthorized access as per FR3.1 in the SRS

How test will be performed:

- Log in as User A
- Attempt to access User B's profile information by manipulating URLs or using application functionalities
- Confirm that access is denied and an appropriate error message is displayed
- Access the database (with proper authorization) to inspect the users table
- Verify that passwords are stored using secure hashing algorithms (e.g., bcrypt, Argon2) and are not in plaintext

FR3.2 - Authentication of Users as McMaster Students

1. **FR3.2-TC1**

   Control: Manual

   Initial State:

   - A user who is a registered student has valid university credentials

Inputs:

- User enters valid McMaster University email and credentials during registration/login

Output:

- The user is authenticated successfully

Test Case Derivation: This test ensures that only registered McMaster University students can access the system as per FR3.2 in the SRS

How test will be performed:

- Attempt to register/log in using a valid McMaster University email (e.g., student@mcmaster.ca) and correct credentials
- Verify that the system authenticates the user successfully and grants access to the landing page

2. **FR3.2-TC2**

   Control: Manual

   Initial State:

   - A user who is not a registered student attempts to register

   Inputs:

   - User enters a non-university email or invalid credentials during registration/login

   Output:

   - Authentication fails
   - The user receives an error message: "Authentication failed. You must be a registered McMaster University student to access this system."

   Test Case Derivation: This test ensures that only registered McMaster University students can access the system as per FR3.2 in the SRS

   How test will be performed:

- Attempt to register/log in using a non-university email (e.g., user@gmail.com) or invalid credentials
- Verify that authentication fails
- Confirm that an appropriate error message is displayed
- Ensure that no account is created or access granted for the invalid user

FR3.3 - Editing User Profile Information

1. **FR3.3-TC1**

   Control: Manual

   Initial State:

   - A user is authenticated and logged into the system
   - The user's profile contains existing information (phone number, name, email, etc.)

   Inputs:

   - User clicks "Edit Profile"
   - User edits their phone number from "905-2134-2930" to "905-2134-2940"
   - User clicks "Save"

   Output:

   - The system saves the updated profile information
   - The system displays the updated information in the UI
   - The updated data is successfully stored in the DB
   - A confirmation message is displayed: "Profile information updated successfully"

   Test Case Derivation: This verifies that users can update their profile information as per FR3.3 in the SRS

   How test will be performed:

- Log in as a user
- Navigate to "Profile" page, and select "Edit Profile"
- Edit phone number from "905-2134-2930" to "905-2134-2940" and click "Save"
- Verify the update is reflected in the system's UI
- Verify with Postman the the appropriate PATCH request is made and the update is stored in the DB

4.2 Tests for Nonfunctional Requirements

4.2.1 NFR10 - Look and Feel

Appearance

1. T1

   Type: Static

   Initial State: System is loaded on two different types of devices (desktop, tablet, mobile)

   Input/Condition: NA

   Output: Completed checklist, indicating whether the requirement is satisfied

   How Test Will Be Performed: The user will complete a checklist that determines whether the interface is responsive across different types of devices

Style

1. T1

   Type: Static

   Initial State: NA

   Input/Condition: NA

   Output: Completed checklist, indicating whether the user is familiar with the icon that appears in the program

How Test Will Be Performed: The user will complete a checklist filled with symbols from the system, and they are asked to note whether they know what the symbol means.

4.2.2 NFR11 - Usability and Humanity

Ease of Use

1. T1

   Type: Static

   Initial State: System is loaded on the home page

   Input/Condition: NA

   Output: Completed checklist, indicating the ease of use of the system

   How Test Will Be Performed: The user will complete a checklist documenting how easy it was to complete a specified task and how long it took them to locate the relevant information.

2. T2

   Type: Static

   Initial State: System is loaded on the home page

   Input/Condition: NA

   Output: Completed checklist, indicating whether the ease of use of the system

   How Test Will Be Performed: The user will be asked to complete a specific task while purposefully giving input errors. They will then fill out a checklist to determine whether the system includes relevant error messages that helps the user correct their mistake.

Personalization and Internationalization

1. T1

   Type: Static

   Initial State: System is loaded at the users account information

Input/Condition: NA

Output: Completed checklist, indicating the usability of the personalization features

How Test Will Be Performed: The user will be asked to edit their profile and complete a checklist that documents how many clicks it takes and how long it takes for the changes to be reflected.

Learning

1. T1

   Type: Static

   Initial State: System is loaded at the home page

   Input/Condition: NA

   Output: Completed checklist, indicating the learnability of the system

   How Test Will Be Performed: The user will be asked to complete a task with only the aid of the systems learning materials. They will then fill in a checklist of how accessible the information was and how much it helped them

Understandability and Politeness

1. T1

   Type: Static

   Initial State: NA

   Input/Condition: NA

   Output: Completed checklist, indicating the understandability of the system

   How Test Will Be Performed: The user will be given a list of text and instructions that appear when navigating the system. They will then fill out a checklist ensuring that they believe the instruction is written in plain language and uses no technical jargon.

Accessibility

1. T1

   Type: Static

   Initial State: System is loaded at the home page

   Input/Condition: NA

   Output: Completed checklist, indicating the accessibility of the system

   How Test Will Be Performed: The user will be asked to complete a task only using their keyboard and they will complete a checklist determining how feasible it was

2. T2

   Type: Automatic

   Initial State: System is loaded at the home page

   Input/Condition: The URL of the system

   Output: Pass Or Fail

   How Test Will Be Performed: Run a python script that locates all text elements on a page and determines the displayed font size. Test will pass if all text is at least 12 font size

4.2.3 NFR12 - Performance

Speed and Latency

1. T1

   Type: Automatic

   Initial state: System is ready to receive a new payment request

   Input/Condition: A dummy/QA payment request form

   Output: Success/Failure/Error message following processing of the request

   How test will be performed: Timer module will record time between sending of the input and receipt of the output to empirically determine processing time.

2. T2

   Type: Automatic

   Initial state: System is not processing and can accept a new invoice generation request

   Input/Condition: A dummy invoice generation request

   Output: Newly generated invoice or appropriate failure/error message

   How test will be performed: Timer module will record time between sending of the input and receipt of the output to empirically determine processing time.

3. T3

   Type: Automatic

   Initial state: System is ready to receive a new reimbursement request

   Input/Condition: A dummy/QA reimbursement request form

   Output: Success/Failure/Error message following processing of the request

   How test will be performed: Timer module will record time between sending of the input and receipt of the output to empirically determine processing time.

Safety-Critical

1. T1

   Type: Automatic

   Initial state: system is ready to receive a new reimbursement request

   Input: A dummy/QA reimbursement request form

   Output: Success/Failure/Error message

   How test will be performed: Verify the UI, upon completion of the request, displays the credit card number used for the request is masked, with * in place of all but the first 4 digits.

2. T2

   Type: Automatic

   Initial State: System is logged in with an account with access controls for student (non-MES admin)

   Input: Request for data from an MES admin account

   Output: Failure/Error message

   How test will be performed: System should not display information to non-authorized user but instead display an appropriate message explaining why the request was denied.

Precision and Accuracy

1. T1

   Type: Static/Manual

   Initial State: N/A

   Input: N/A

   Output: N/A

   How test will be performed: Static analysis of code modules to ensure all monetary calculations are rounded to 2 decimal places.

Robustness or Fault-Tolerance

1. T1

   Type: Automatic

   Initial State: System is about to upload daily backups to backup server

   Input: N/A

   Output: Backed up data is moved to the other database

   How test will be performed: Module will automatically compare the entries added to the backup database with those added to the main database on that day and raise an exception for any mismatch.

Capacity

1. T1

   Type: Automatic

   Initial State: System is not processing and is prepared to receive new requests

   Input: Large number of reimbursement, invoice, or payment requests (ie, from a load testing software to submit the requests to the API)

   Output: Success messages for all requests

   How test will be performed: System should be able to handle up to a certain number of requests per day as defined in the capacity requirements without failing.

2. T2

   Type: Automatic

   Initial State: System in not processing and is prepared to receive new requests

   Input: Large number of simultaneous reimbursement, invoice, or payment requests

   Output: Success messages for all requests

   How test will be performed: System should be able to handle up to a certain number of simultaneous requests as defined in the capacity requirements without failing.

3. T3

   Type: Automatic

   Initial State: System in not processing and is prepared to receive new requests

   Input: 10 years' worth of dummy data

   Output: Successful data storage

How test will be performed: 10 years' worth of dummy data will be added to the backup database to ensure that it can store all data for 10 years.

Scalability or Extensibility

1. T1

   Type: Automatic/Static Review

   Initial State: System in not processing and is prepared to receive new requests

   Input: Increasingly large numbers of payment requests to stress test (ie, from a load testing application)

   Output: Eventual system failure; system metrics provided by testing application or profiling tools

   How test will be performed: System will undergo stress tests until failure. After, developers will review code and use profiling tools to identify bottlenecks and assess current scalability.

2. T2

   Type: Automatic

   Initial State: System is not processing and is prepared to receive new requests

   Input: Dummy API requests (eg, from Postman) that mimic real third party requests

   Output: 200 Success and success body with proper format

   How test will be performed: Endpoints that will be open to third-party integrations provide success messages for basic requests (GET, POST) and return the message body in JSON format.

Longevity

1. T1

Type: Static

Initial State: N/A

Input: N/A

Output: Checklist of recommendations, and comments

How test will be performed: Developers and outside reviewers will perform static analysis of major code components to assess design choices and adaptability. They will also review documentation to assess if future developers will have the necessary support to maintain the system without difficulties.

4.2.4 NFR13 - Operational and Environmental

Expected Physical Environment

1. T1

   Type: Automatic

   Initial State: System is not processing and is prepared to receive new requests

   Input: Several payment requests

   Output: Success messages for all requests

   How test will be performed: This test will be performed on a machine with Windows 10 and matching or similar specs to Intuit Quickbooks' minimum requirements. The test is successful if no requests fail and are delivered within maximum time requirements (as outlined in 12.1 of our SRS document).

Wider environment

1. T1
   Type: Static

   Initial State: N/A

   Input: N/A

   Output: Checklist

How test will be performed: Static analysis of code, using a checklist, to ensure that the implemented processes conform to the industry standard for Canadian financial institutions

Interfacing with Adjacent Systems

1. T1

   Type: Automatic

   Initial State: System is not processing and is prepared to receive new requests

   Input: Mock payment request forms for users of all major Canadian banks (BMO, CIBC, RBC, Scotiabank, TD)

   Output: Success message

   How test will be performed: Valid request forms should be processed with a success message for all major Canadian financial institutions.

Productization

1. T1

   Type: Static

   Initial State: N/A

   Input: N/A

   Output: Peer-review checklist

   How test will be performed: Users will be given training materials and fill out a checklist review of how easy they and the system are to understand.

Release

1. T1

   Type: Static

Initial State: N/A

Input: N/A

Output: Checklist of recommendations, and comments

How test will be performed: Developers and outside reviewers will perform static analysis of major code components to assess design choices and advise if 2 releases per year is doable and appropriate.

4.2.5 NFR14 - Maintainability and Support

Maintenance

- T1

  Type: Static

  Initial State: N/A

  Input: N/A

  Output: Checklist of recommendations, and comments

  How test will be performed: Developers and outside reviewers will perform static analysis of major code components to assess design choices and advise if 10

Supportability

- T1

  Type: Manual

  Initial State: System has just undergone functional changes/updates

  Input: N/A

  Output: Checklist

  How test will be performed: Reviewer will review user and team documentation upon each development release and provide feedback (in the form of a checklist) if the update to documentation sufficiently keeps track of the changes to software. :

Adaptability

- T1

  Type: Automatic

  Initial State: System is not processing and is prepared to receive new requests

  Input: Valid dummy payment requests

  Output: Success message

  How test will be performed: Automated regression testing, using previously valid input data, will be performed regularly on modules that integrate with external sources (eg, financial institutions) to check for external changes that affect our software.

4.2.6 NFR15 - Security

Access

1. T1

   Type: Static

   Initial State: System is loaded on the home page

   Input/Condition: NA

   Output: Completed checklist, indicating whether the user has access to the correct features

   How Test Will Be Performed: The user will complete a checklist determining which features they have access to, test will be done across a student level and an admin level.

2. T2

   Type: Static

   Initial State: System is loaded at the login portal

   Input/Condition: NA

Output: Completed checklist, indicating whether the login portal meets all requirements.

How Test Will Be Performed: The user will complete a checklist to determine the security of the login portal. They will be asked to rate the performance of multi-factor authentication, password strengths and how inactive users are handled.

Integrity/Audit

1. T1

   Type: Static

   Initial State: System is showing transaction logs

   Input/Condition: NA

   Output: Completed checklist, indicating whether the system has data integrity

   How Test Will Be Performed: The user will be given a list of financial transaction that should be logged, and will verify that they are correctly logged

Privacy

1. T1

   Type: Static

   Initial State: System is showing financial data that is meant to be anonymous

   Input/Condition: NA

   Output: Completed checklist, indicating whether the privacy requirements are met

   How Test Will Be Performed: The user will complete a checklist determining whether the sensitive information is encrypted and anonymous

Immunity

1. T1

   Type: Manual

   Initial State: System is operational with available data stored in the database

   Input/Condition: Corrupting a file and initiating recovery procedures

   Output: Time it took the successfully recover the data

   How Test Will Be Performed: Simulate a data corruption event and time how long it takes to successfully recover the data, ensure that it is under 4 hours

4.2.7 NFR16 - Cultural

Cultural

1. T1

   Type: Static

   Initial State: System is loaded on the home page

   Input/Condition: NA

   Output: Completed checklist, indicating whether the user can understand the system

   How Test Will Be Performed: The user will complete a checklist determining whether they can understand the system instructions and how the system performs. This test will be done primarily by international students or students who do not speak English as their first language.

4.2.8 NFR17 - Compliance

Legal

1. T1

   Type: Static

Initial State: System is loaded on the home page

Input/Condition: NA

Output: Completed checklist, indicating whether the system meets legal requirements

How Test Will Be Performed: The user will be given a list of relevant laws and regulations and will complete a checklist determining whether the system complies

Standard Compliance

1. T1

   Type: Static

   Initial State: System is loaded on the home page

   Input/Condition: NA

   Output: Completed checklist, indicating whether the system meets web security standards

   How Test Will Be Performed: The user will be given a list of relevant compliance standards and will complete a checklist determining whether the system adheres to the guidelines

4.2.9 Non-Functional Checklist Summary

NFR Test Case	Check List
NFR10	Look and Feel Checklist
NFR11	Usability and Humanity Checklist
NFR12	Performance Checklist
NFR14	Maintainability and Support Checklist
NFR15	Security Checklist
NFR17	Compliance Checklist

Table 2: Mapping between non-functional test cases and their respective checklist

4.3 Traceability Between Test Cases and Requirements

4.3.1 Traceability Between Test Cases and Functional Requirements

Test Case	**Functional Requirement**
FR1.1-TC1	FR1.1
FR1.1-TC2	FR1.1
FR1.2-TC1	FR1.2
FR1.2-TC2	FR1.2
FR1.3-TC1	FR1.3
FR1.4-TC1	FR1.4
FR1.5-TC1	FR1.5
FR1.6-TC1	FR1.6
FR1.7-TC1	FR1.7
FR1.7-TC2	FR1.7
FR1.8-TC1	FR1.8
FR1.9-TC1	FR1.9
FR3.1-TC1	FR3.1
FR3.2-TC1	FR3.2
FR3.2-TC2	FR3.2
FR3.3-TC1	FR3.3

Table 3: Mapping between test cases and functional requirements

Test Case	Non-Functional Requirement
NFR10 - Appearance - T1	APR1, APR2
NFR10 - Style - T1	STYR1, STYR2
NFR11 - Ease of Use - T1	EUR1, EUR3
NFR11 - Ease of Use - T2	EUR2
NFR11 - Personalization and Internationalization - T1	PIR1, PIR2
NFR11 - Learning - T1	LER1, LER2
NFR11 - Understandability and Politeness - T1	UAPR1
NFR11 - Accessibility - T1	ACSR1
NFR11 - Accessibility - T2	ACSR2
NFR12 - Speed and Latency - T1	SPLR1
NFR12 - Speed and Latency - T2	SPLR2
NFR12 - Speed and Latency - T3	SPLR3
NFR12 - Safety Critical - T1	SFCR 1, PRD2
NFR12 - Safety Critical - T2	SFCR 2, RFT 2, PRD3
NFR12 - Precision and Accuracy - T1	POAR1
NFR12 - Robustness Or Fault-Tolerance - T1	RFT 1, MR1
NFR12 - Capacity - T1	CPR1
NFR12 - Capacity - T2	CPR2
NFR12 - Capacity - T3	CPR3
NFR12 - Scalability or Extensibility - T1	SER2, PRD1, PRD7
NFR12 - Scalability or Extensibility - T2	SER1, IAR2
NFR12 - Longevity - T1	LOR1

Table 4: Mapping between test cases and non-functional requirements

4.3.2 Traceability Between Test Cases and Non-Functional Requirements

Test Case	Non-Functional Requirement
NFR13 - Operational and Environmental - Expected Physical Environment - T1	OR1, OR2
NFR13 - Operational and Environmental - Wider Environment - T1	WR1
NFR13 - Operational and Environmental - Interfacing with Adjacent Systems - T1	IAR1, PRD4
NFR13 - Operational and Environmental - Productization - T1	PRD5, PRD6
NFR13 - Operational and Environmental - Release - T1	RLR1
NFR14 - Maintainability and Support - Maintenance - T1	MR2, MR3
NFR14 - Maintainability and Support - Supportability - T1	SR1
NFR14 - Maintainability and Support - Adaptability - T1	AR1
NFR15 - Access - T1	SCR1
NFR15 - Access - T2	SCR2, SCR3, SCR4, SCR5, SCR6
NFR15 - Integrity/Audit - T1	INR1, ADR1, ADR2
NFR15 - Privacy - T1	PVR1, PVR2
NFR15 - Immunity - T1	IMM1
NFR16 - Cultural - T1	CLTR1
NFR17 - Legal - T1	LR1, LR2
NFR17 - Standards Compliance	STR1

5 Unit Test Description

This section cannot be completed until after the detailed design document has been completed, therefore it is omitted.

6 Appendix

6.1 Checklists

6.1.1 Look and Feel

- ☐ Does the UI render properly on desktops (minimum resolution 1024x768)
- ☐ Does the UI render properly on mobile devices
- ☐ Are all elements correctly aligned on all devices?
- ☐ Are fonts, colors, and spacing consistent acorss all pages?
- ☐ Are the branding elements (logos, colors) consistent with existing MES branding?
- ☐ Are all icons easily recognizable and intuitive to their function?
- ☐ Is there a tooltip or label for icons without obvious meaning?
- ☐ Is the most important information on each page displayed clearly and prominently?

6.1.2 Usability and Humanity

- ☐ Can users navigate to primary function pages (i.e reimbursement request, generate invoice, admin page) within 3 clicks from the home page?
- ☐ Does it take the user 2 minutes to complete desired task?
- ☐ Are form validation messages correct? (i.e "Enter valid email address", "Enter valid McMaster Student Number")?
- ☐ Can user edit personal profile information within 3 clicks?
- ☐ Are personalization changes reflected immediately in the UI? (i.e Phone number)
- ☐ Is the user tutorial video easily accessible from the home page?

- ☐ Are all instructions free of technical jargon, written in clear and simple language?
- ☐ Are button labels self-explanatory? (i.e "Submit Request")

6.1.3 Performance

- ☐ Are responses to users actions completed to under 1 second for 95% of tasks?
- ☐ Does the system process large files (i.e invoices) within 30 seconds?
- ☐ Does the system recover from a simulated crash without data loss?
- ☐ Are backups stored and verified successfully?
- ☐ Can the system handle 20 simultaneous users without performance degradation?
- ☐ Can it process 10 years of dummy data without failure?

6.1.4 Maintainability and Support

- ☐ Are coding standards followed? (i.e Consistent naming and indentation conventions, approriate comments)
- ☐ Are there no critical errors identified by build tools?
- ☐ Is documentation up to date?
- ☐ Are software updates and patches tested and documented?

6.1.5 Security

- ☐ Are unauthorized users prevented from accessing sensitive data?
- ☐ Is multi-factor authentication enforced for all admin accounts?
- ☐ Is all senitive data encrypted in transit and at rest?
- ☐ Are audit logs maintained for all access and modifications?
- ☐ Are all fincancial transactions correctly logged and traceable?
- ☐ Can the system recover from simulated attacks within 4 hours?

6.1.6 Compliance

- ☐ Does the system adhere to relevant data protection policy?
- ☐ Are all financial transactions compliant with Canadian financial regulations?
- ☐ Are web security standards followed?

6.2 Symbolic Parameters

Not applicable for this delierable

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?

   Christian: There was a high degree of certainty with this document that made the final product come out well. Knowing that there must be tests for all requirements and that plans must be made to verify and validate all aspects of the project thus far as well as future elements, there is almost no ambiguity in what should be included, only with making sure everything relevant is included, as, to an extent, you can never be too sure when validating.

   Austin: The availability of resources to facilitate the creation of this document.

   Jacob: Being able to go back to prior documentation was useful for this deliverable, and provided a reference for keeping consistent while writing this deliverable.

   Adam: One thing that went well while writing this deliverable was having the SRS already written and being able to reference back to it when coming up with the system tests for the functional requirements. Having requirements organized by feature and labelled streamlined the process of formulating system requirements.

Evan: One thing that went well when writing this deliverable was the time we were given. Allowing us to get feedback on our SRS before this deliverable was due made a huge impact on the quality of our submission. Some times it can feel rushed, especially when deliverables are due on Wednesday and we have our TA meeting on Monday, but this assignment was spread out enough that we were able to feel the benefits of the feedback.

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

   Christian: The biggest pain point was simply making sure that all relevant information was present in the document. With so much to potentially validate and verify, it becomes a worry that you're not being rigorous enough, or you haven't anticipated enough. To mitigate this we engaged in group brainstorming as well as constantly supporting each other remotely. Another fact that helped was knowing that many of the specifics for this document are not imminently necessary to nail down.

   Austin: Trying to forecast what documents the VnV will use that have not been made yet

   Jacob: Trying to cover all relevant information regarding the requirements we came up with a couple deliverables ago was difficult as there was a lot to cover and it was easy to miss things. I alleviated it by just being meticulous and cross-referencing the old documents.

   Adam: The biggest pain point experienced while writing this deliverable was simply ensuring all test cases were correctly mapped to all requirements. Since the document is very long and there are numerous test cases and requirements, it was quite tedious to ensure the mapping was correct in the traceability section. This was resolved by simply taking the time to double-check the mappings

   Evan: The biggest pain point of this deliverable was the amount of tests that were needed for NFRs. We used the Volere template for our SRS which added a lot of NFRS and making test cases that covered each one was tedious.

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.

Austin: One knowledge I need to aquire is a better traceability of design requirements and decisisons into their respective documentation, to ensure nothing has been overlooked.

Christian: Personally, I will need to develop my knowledge of existing testing tools. I have a fair understanding of testing methods and what methods apply when, however I admittedly have little experience with actual testing tools, and should familiarize myself with any potential tools that can be useful for us going forward, whether it be for static or dynamic testing.

Jacob: For me it would be helpful to learn more about static code analysis and performance metrics. I have experience with testing tools for functionality but non-functional testing via other methods is relatively new to me.

Adam: One skill that I have to acquire is knowledge of how to use Postman for testing and developing APIs. This will come in handy for testing a lot of the functional requirements and making sure the correct requests are being made between frontend, backend and database.

Evan: I would need to learn more about automated testing. Familiarizing myself with the tools that we will be using is important for this project to be succesful.

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?

   Christian: One approach to mastering verification ot validation tools would be to simply read up on many different kinds and learn their strengths or weaknesses. While many testing platforms may explicitly offer the functionality we're looking for, some are lkely to be more geared to our application than othes. Another approach could be to find the most popular platform and get familiar with that one specifically, as a way to better prepare myself not only for this project but also what may be expected of me in industry for similar types of projects.

Personally, I do prefer the first option however because I value versatility in my development abilities and would rather not be pidgeonholed into a specific niche too early.

Austin: Perhaps some visualization technique, to see the ”flow” of requirements or design decisions reflected in a checklist

Jacob: Two approaches would be to gain firsthand experience by analyzing software or reading through educational materials to gain a better understanding of industry-standard techniques and tools. The latter is easier and more viable, so I would carry through with that.

Adam: Postman is a very popular tool so there is loads of documentation out there to learn from. Other than this, there are definitely numerous Youtube tutorials teaching the use of Postman in web development projects. Out of the two approaches, I think I will watch a few Youtube videos to gain a better understanding of this tool. This approach is slightly better seeing as the information is usually condensed and presented in a logical format, which is much more effective and time-efficient than just reading through documentation.

Evan: To familiarize myself with automated testing tools there are two ways to go about it. Doing research online or attending classes or seminars about it. Online research might be easier to do but in-person hands-on experience might prove to be more valuable across the length of this project. Personally, I will choose to do online research since it is less time-consuming and there are plenty of thorough resources that I can use.