# Verification and Validation Report: McMaster Engineering Society Custom Financial Expense Reporting Platform

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

April 4, 2025

# 1 Revision History

| Date | Version | Notes |
|---|---|---|
| 3/11/2025 | 1.0 | Initial revision, including all sections and reflection |
| 3/24/2025 | 2.0 | Updating for Authentication Module, see commit: 4cef88d |

# Contents

# List of Tables

# 2 Functional Requirements Evaluation

The functional requirements can be referenced from the SRS. The modules can be referenced below and from the Module Guide:

- M1: Hardware-Hiding Module
- M2: Account Management Module
- M3: Requests Module
- M4: Notification Module
- M5: User Dashboard Module
- M6: ~~Authentication Module~~
- M7: Email Module
- M8: Account Management Controller Module
- M9: Requests Controller Module
- M10: Clubs Database
- M11: Users Database
- M12: Requests Database
- M13: Graphical User Interface

Table 1 summarizes the functional requirements checklist for each functional requirement from the SRS, linking to each module from the MG and test case from the VnV Plan. If a functional requirement was determined to be satisfied or not satisfied by the tests with a "*" flag, then more explanation is given below:

- ~~FR1.3: UI components and pages are implemented to allow users and admins to edit submitted requests, however, work must now be done to connect backend to frontend UI to allow for changes to be reflected in the database.~~ Changes now reflected in database

- ~~FR1.5: The system does not currently notify users of changes to the status of their reimbursement request, however, the email notification service/module is set up and can be executed in an isolated environment. Work must now be done ensure the email service can dynamically email users with updates for their submitted requests.~~Email service is now functional for request updates.

- ~~FR1.6: UI is implemented to allow users to select a budget category. Budget categories need to be dynamically pulled from a database collection. Work must be done to connect backend service (Plaid API) to frontend UI to allow budget categories to be linked to requests and reflected in the database.~~Budget categories were replaced with item escriptions. Plaid has been setup.

- ~~FR1.7: Due to the fact that authentication is not completely implemented, this requirement cannot be satisfied. We do have page routing and UI implemented to distinguish admin page views and user page views.~~ *Note: it has been decided we will integrate into the existing MES authentication, which shall satisfy this requirement.*

- ~~FR3.2: Authentication is not fully set up. Currently, we are manually setting user browser tokens when creating a user and logging in, work must be done to either dynamically assign user browser token, or adjust the authentication process to user vanilla username (email) and password for the login process.~~ *See note above.*

Table 1: Functional Requirements Evaluation

| FR | Modules | Test Suite | Test Cases | Satisfied (Y/N) |
|---|---|---|---|---|
| FR1.1 | M3, M5, M9, M10, M11, M12, M13 | reconciler.test.js | FR1.1-TC1, FR1.1-TC2 | Yes |
| FR1.2 | M3, M5, M9, M10, M11, M12, M13 | requests.test.js | FR1.2-TC1, FR1.2-TC2 | Yes |
| FR1.3 | M3, M5, M9, M10, M11, M12, M13 | requests.test.js | FR1.3-TC1 | Yes* |
| FR1.4 | M3, M5, M9, M10, M11, M12, M13 | reconciler.test.js | FR1.4-TC1 | Yes |
| FR1.5 | M4, M7 | emailer.test.js | FR1.5-TC1 | No* |
| FR1.6 | M12 | reconciler.test.js | FR1.6-TC1 | No* |
| FR1.7 | M6 | Acct. mgmt. TS (TBD) | FR1.7-TC1, FR1.7-TC2 | No* |
| FR1.8 | M5, M3, M9, M13 | requests.test.js | FR1.8-TC1 | No |
| FR1.9 | M12 | reconciler.test.js | FR1.9-TC1 | No |
| FR3.1 | M10, M11 | Acct. mgmt. TS (TBD) | FR3.1-TC1 | Yes |
| FR3.2 | M6 | emailer.test.js, Acct. mgmt. TS (TBD) | FR3.2-TC1, FR3.2-TC2 | No* |
| FR3.3 | M2, M8 | | FR3.3-TC1 | Yes |

# 3 Nonfunctional Requirements Evaluation

## 3.1 Look and Feel (NFR10)

- Checklist

- **Appearance (Test T1):**
  - **Checklist 6.1.1.1:** UI renders properly on desktops (1024x768) – **Passed**.
  - **Checklist 6.1.1.2:** UI renders properly on mobile devices ~~**Failed.**~~**Passed - updated for final presentation**
  - **Checklist 6.1.1.3:** All elements aligned on all devices – ~~**Failed.**~~**Passed**

  - **Checklist 6.1.1.4:** Fonts, colors, spacing consistent – **Passed**.
  - **Checklist 6.1.1.5:** Branding aligns with MES guidelines – **Passed**.
  - **Checklist 6.1.1.8:** Key information displayed prominently – **Passed**.

- **Style (Test T1):**
  - **Checklist 6.1.1.6:** Icons intuitive – **Passed**.
  - **Checklist 6.1.1.7:** Tooltips for unclear icons – **Passed**.

**Notes**

- **Failures in Appearance T1:**
  - ~~Mobile rendering exhibited layout inconsistencies on screens smaller than 5.5 inches (Checklist 6.1.1.2).~~
  - ~~Elements misaligned on mobile devices during form submissions (Checklist 6.1.1.3).~~
- **Successes in Style T1:** Users recognized all icons intuitively, and tooltips were provided for ambiguous symbols (Checklist 6.1.1.6–7). After revisions, everything now displays properly on mobile (Checklist 6.1.1.2,6.1.1.3)

## 3.2 Usability and Humanity (NFR11)

- Checklist

- **Ease of Use (Tests T1, T2):**
  - **Checklist 6.1.2.1:** Navigate to primary functions in $\leq 3$ clicks – **Passed**.
  - **Checklist 6.1.2.2:** Form validation messages correct – **Passed**.
  - **Checklist 6.1.2.3:** Complete tasks in $\leq 2$ minutes – **Passed**.
  - **Checklist 6.1.2.4:** Edit profile in $\leq 3$ clicks – **Passed**.
  - **Checklist 6.1.2.5:** Personalization changes reflected immediately – **Passed**.

  - **Checklist 6.1.2.6:** Tutorial video accessible – ~~**Failed.**~~**Passed**
  - **Checklist 6.1.2.7:** Instructions free of jargon – **Passed**.
  - **Checklist 6.1.2.8:** Button labels clear – **Passed**.

- **Accessibility (Test T1, T2):**
  - **Checklist 6.1.2.9:** Keyboard navigation equivalent to mouse – **Passed**.
  - **Checklist 6.1.2.10:** Text font size $\geq$ 12pt – **Passed**.

**Notes**

- ~~**Failure in Ease of Use (Checklist 6.1.2.6):** The tutorial video was not easily accessible from the home page.~~Following the final demo, we created a tutorial video which is now accessible.
- **Testing Context:** Results for NFR10 and NFR11 were conducted within the group, supervisor, and external user.

## 3.3 Performance (NFR12)

- Checklist

- **Speed and Latency (Tests T1, T2, T3):**
  - **Checklist 6.1.3.1:** 95% of tasks completed in $\leq$ 1s – **Passed**.
  - **Checklist 6.1.3.2:** Large files processed in $\leq$ 30s – **Failed**.
  - **Checklist 6.1.3.3:** System recovers from crash without data loss – **Passed**.
  - **Checklist 6.1.3.4:** Backups stored and verified – **Failed**.
  - **Checklist 6.1.3.5:** Handles 20 simultaneous users – **Failed**.
  - **Checklist 6.1.3.6:** Processes 10 years of dummy data – **Failed**.
- **Safety-Critical (Test T1, T2):**
  - **Checklist 6.1.3.7:** Credit card numbers masked – **Failed**.
  - **Checklist 6.1.3.8:** Unauthorized access prevented – ~~**Failed.**~~**Passed following revision**

- **Precision and Accuracy (Test T1):**
  - **Checklist 6.1.3.9:** Monetary calculations rounded to 2 decimal places – **Passed**.
- **Robustness or Fault-Tolerance (Test T1):**
  - **Checklist 6.1.3.10:** Daily backups verified – **Failed**.
- **Capacity (Tests T1, T2, T3):**
  - **Checklist 6.1.3.11:** Handles large request volume – **Failed**.
  - **Checklist 6.1.3.12:** Handles simultaneous requests – **Failed**.
  - **Checklist 6.1.3.13:** Stores 10 years of dummy data – **Failed**.
- **Scalability or Extensibility (Tests T1, T2):**
  - **Checklist 6.1.3.14:** Handles stress tests until failure – **Failed**.
  - **Checklist 6.1.3.15:** Third-party endpoints return JSON – **Passed**.
- **Longevity (Test T1):**
  - **Checklist 6.1.3.16:** Code components adaptable for future maintenance – **Passed**.

**Notes**

- **Failures in Development Environment:** Many failed test cases (e.g., capacity, simultaneous users, backups) are attributed to the system being in development rather than production.

## 3.4 Operational and Environmental (NFR13)

- **Expected Physical Environment (Test T1):**
  - **Checklist 6.1.4.1:** Compatibility with Windows 10 and Intuit QuickBooks specs – **Deferred**.
- **Wider Environment (Test T1):**

  - **Checklist 6.1.4.2:** Compliance with Canadian financial institution standards – **Deferred**.

- **Interfacing with Adjacent Systems (Test T1):**
  - **Checklist 6.1.4.3:** Integration with major Canadian banks – **Deferred**.

- **Production (Test T1):**
  - **Checklist 6.1.4.4:** User training material clarity – ~~**Deferred.**~~**Passed**

- **Release (Test T1):**
  - **Checklist 6.1.4.5:** Feasibility of biannual releases – **Deferred**.

**Notes**

- **User manual:** It was decided as part of our extras to upload a user manual, see: https://github.com/ausbennett/mes-finance-platform/tree/main/docs/Extras/UserManual
- **Deferred Testing:** Tests for NFR13 will align with the system's progression to production readiness. Current development constraints (e.g., incomplete integrations, non-finalized environments) prevent accurate assessment.
- **Next Steps:** These tests will be prioritized during later development phases, including post-deployment monitoring and stakeholder reviews.

## 3.5 Maintainability and Support (NFR14)

- Checklist

- **Maintenance (Test T1):**
  - **Checklist 6.1.4.1:** Coding standards followed – **Passed**.
  - **Checklist 6.1.4.2:** No critical errors in build tools – **Passed**.
  - **Checklist 6.1.4.3:** Documentation up to date – **Passed**.
  - **Checklist 6.1.4.4:** Updates tested and documented – **Passed**.

- **Supportability (Test T1):**
  - **Checklist 6.1.4.5:** Documentation updated with releases – **Passed**.
- **Adaptability (Test T1):**
  - **Checklist 6.1.4.6:** Regression tests performed regularly – **Failed**.

## 3.6 Security (NFR15)

- Checklist

- **Access (Tests T1, T2):**
  - **Checklist 6.1.5.1:** Unauthorized access prevented – **Passed**.
  - **Checklist 6.1.5.2:** Multi-factor authentication enforced – **Failed**.
- **Integrity/Audit (Test T1):**
  - **Checklist 6.1.5.3:** Audit logs maintained – ~~**Failed.**~~**Passed**
- **Privacy (Test T1):**
  - **Checklist 6.1.5.4:** Sensitive data encrypted – **Failed**.
- **Immunity (Test T1):**
  - **Checklist 6.1.5.5:** System recovers from attacks within 4 hours – **Failed**.

### Notes

- **Audit logs:** Audit logging and reconciliation with Plaid integration was done based on stakeholder feedback.

## 3.7 Cultural (NFR16)

- **Testing Status:** Cultural compliance testing (e.g., internationalization, accessibility for non-English speakers) will be conducted during future development phases.

### 3.8 Compliance (NFR17)

- Checklist

- **Legal (Test T1):**
  - **Checklist 6.1.6.1:** Adheres to data protection policies – **Failed**.
  - **Checklist 6.1.6.2:** Compliant with Canadian financial regulations – **Failed**.
- **Standard Compliance (Test T1):**
  - **Checklist 6.1.6.3:** Follows web security standards – **Failed**.

**Notes**

- **Deferred Fixes:** Compliance failures (data protection, financial regulations, web security) will be prioritized during future development.

## 4 Unit Testing

Most files, excluding the 3rd-party library, top-level and GUI modules had corresponding unit tests. All tests were performed using jest for testing javascript, and each commit to the main branch must pass all the tests. All the tests passed as seen in the Test Report 10.1.

## 5 Changes Due to Testing

During Rev0 presentations, as well as meetings with the our supervisor, it was made apparent that authentication will be an "assumed" feature and not necessary to be showcased during final demos or expo. With that being said, we've decided as a team to put any further authentication changes on the backlog, as MES currently has an express.js authentication system that may seamlessly integrate into our current project. A change that was implemented is our use of the plaid API to aid in reconciliation and audit logging; this was feedback from our supervisor when showcasing the plaid API integration. We also conducted usability testing with a fellow engineering student. The documentation for this testing can be found at: https://github.com/ausbennett/

mes-finance-platform/tree/main/docs/Extras/UsabilityTest. Three key pieces of feedback were presented after. Firstly, the student found clicking the MES logo button to return to the landing page was unintuitive and initially struggled with this navigation. In response, we added a new "Dashboard" button in the top right to explicitly tell the user how to return (note that the MES button is still present and navigating to the dashboard). Secondly, the student expressed that our red buttons on the new request page were too bright and did not contrast well with the white page background. We updated the buttons to a be darker maroon with white lettering, which provides much better contrast and conforms to our McMaster colours. Finally, the user did not like our edit account button, which was simply the word "Account". We agreed that it was unappealing and changed it to an icon of a person, which is intuitive and removed unnecessary text.

During our final presentation, we also received feedback. It was pointed out that the "total amount" field in our new request forms is entered manually, instead of being autofilled by adding up the individual amounts. This creates opportunities for user error in mistakenly adding or approving incorrect amounts. Furthermore, it was expressed that on our audit page, our click-and-drag reconciliation system may be difficult to use if there are a large number of entries to scroll through. In response to these comments, we intend to autofill the total amount field to prevent mistakes. We also would like to add an option for presenting the audit page in different views, and adding a "Compact" or "List" view to make reconciling large amounts of data easier. However, these changes will have to be implemented in the future as we have already given our final presentation and are now focused on cleaning our documentation and preparing for the expo.

# 6 Automated Testing

The tests were set up to automatically run in GitHub Actions whenever a commit was pushed to the `main` branch. The configuration for the CI/CD automation can be found at https://github.com/ausbennett/mes-finance-platform/blob/main/.github/workflows/test.yml and the test pipeline can be viewed at https://github.com/ausbennett/mes-finance-platform/actions/workflows/test.yml.

# 7 Trace to Requirements

| **Test Suite** | **Functional Requirement** |
|---|---|
| emailer.test.js | FR1.5, FR3.2 |
| reconciler.test.js | FR1.1, FR1.4, FR1.6, FR1.9, INR1, SPLR1-3 |
| requests.test.js | FR1.2, FR1.3, FR1.8, PVR1 |
| Account management test suite (TBD) | FR1.7, FR3.1, FR3.2 |
| Usability testing (via checklist) | APR1, APR2, STYR1, STYR2, EUR1-3, PIR1, LER2, UAPR1, ACSR1, SCR1-5, CLTR1 |
| Static analysis (via checklist) | POAR1, LOR1, WR1, RLR1, ACSR2, MR2, SR1, LR1, LR2, STR1 |
| Load testing | CPR1-3, SER2 |

Table 2: Mapping between test suites and requirements

# 8 Trace to Modules

| **Test Suite** | **Module** |
|---|---|
| emailer.test.js | Notification Module (M4), Authentication Module (M6), Emailer API (M7) |
| reconciler.test.js | Clubs, User, Request Database(M10-12) |
| requests.test.js | Requests Module(M3), Requests Controller (M9) |
| Account management test suite (TBD) | Account Management Module (M2), User Dashboard Module (M5), Account Management Controller (M8) |
| Usability testing (via checklist) | Graphical User Interface (M13) |

Table 3: Mapping between test suites and modules

# 9 Code Coverage Metrics

Code coverage is shown in the Test Report in section 10.1.

# 10 Appendix

## 10.1 Test Report

```
 PASS  tests/requests.test.js
  Request Endpoints and Model Validations
    Model Validations
      should require necessary reimbursement fields (3 ms)
    Reimbursement Endpoints
      should create reimbursement with file upload (34 ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        0.927 s, estimated 1 s
Ran all test suites.
```

# Appendix — Checklists

### 10.1.1 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?

### 10.1.2 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")

### 10.1.3 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?

### 10.1.4 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?

### 10.1.5 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?

### 10.1.6 Compliance

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

# Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Reflection.

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?

   In this deliverable, leveraging tools like GitHub Actions for automated testing helped us streamline validation. Additionally, having already incorporated feedback from the V&V Plan, it made the process of creating the V&V Report much easier. From delegating roles to having pre-made checklists for usability testing, using the V&V Plan made our lives much easier.

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

   A significant challenge was designing a test suite that remains relevant as the project evolves. Since some core features are still in development, early tests risked becoming obsolete as APIs and interfaces changed. To address this, we prioritized testing isolated modules (e.g., database schemas, utility functions) with stable contracts and used mocking for incomplete subsystems. We also adopted a CI/CD-driven iterative approach, updating tests incrementally alongside implementation sprints

to ensure alignment while preserving test value.

3. Which parts of this document stemmed from speaking to your client(s) or a proxy (e.g. your peers)? Which ones were not, and why?

   The usability testing checklist was shaped by stakeholders as our supervisor has been vocal about which features should have a bigger focus on usability rather than expanding the functionality. Other things, such as our automated testing strategy, were internally designed. From the testing we did before the Rev0 demo, we are already aware of which areas require more rigorous testing compared to other features.

4. In what ways was the Verification and Validation (VnV) Plan different from the activities that were actually conducted for VnV? If there were differences, what changes required the modification in the plan? Why did these changes occur? Would you be able to anticipate these changes in future projects? If there weren't any differences, how was your team able to clearly predict a feasible amount of effort and the right tasks needed to build the evidence that demonstrates the required quality? (It is expected that most teams will have had to deviate from their original VnV Plan.)

   The main difference between the plan and the report was the scale of the testing. The plan was more ambitious than what was probably doable in the time frame, and we had pages and pages of tests for NFRs that would have required external stakeholders to execute, some of whom may be hard to get a hold of. As we moved forward it became clear we needed to focus our energy toward a smaller suite that makes more use of automated testing and usability testing with Luke to cover our bases. It is definitely possible in future projects to foresee this kind of an issue. However, timing can be a difficult thing to predict and one often doesn't know how things will pan out until they actually happen.