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

March 10, 2025

1 Revision History

Date	**Version**	**Notes**
Date 1	1.0	Notes
Date 2	1.1	Notes

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test

Contents

List of Tables

List of Figures

3 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 staisfied 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.

- 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.

- 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.

- 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.

- 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.

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

4 Nonfunctional Requirements Evaluation

4.1 Usability

4.2 Performance

4.3 etc.

5 Comparison to Existing Implementation

This section will not be appropriate for every project.

6 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 12.1.

7 Changes Due to Testing

8 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.

9 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 check-list)	APR1, APR2, STYR1, STYR2, EUR1-3, PIR1, LER2, UAPR1, ACSR1, SCR1-5, CLTR1
Static analysis (via check-list)	POAR1, LOR1, WR1, RLR1, ACSR2, MR2, SR1, LR1, LR2, STR1
Load testing	CPR1-3, SER2

Table 2: Mapping between test suites and requirements

10 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

11 Code Coverage Metrics

12 Appendix

12.1 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 — 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.