Verification and Validation Report: Plutos

Team #10, Plutos
Payton Chan
Eric Chen
Fondson Lu
Jason Tan
Angela Wang

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

[symbols, abbreviations or acronyms – you can reference the SRS tables if needed —SS]

Contents

List of Tables

List of Figures

This document ...

3 Functional Requirements Evaluation

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

6.1 Front-end unit tests

Front-end unit tests can be found in the test directory

6.1.1 Component Rendering

- **Description:** Each component is rendered in their corresponding unit test to ensure it renders without error and with all corresponding mock information correctly (i.e. the component rendered with mock expense data is expected to display the mock expense data)
- **Inputs:** The component & component props
- **Expected Output:** Component renders without error and displays corresponding information
- **Result:** Pass

6.1.2 Metrics

This section describes the unit tests for calculating and displaying user metrics.

Expense and Budget

- **Description:** Unit tests to render expense and budget metrics to validate: total expense value, total budget value, percentage of budget spent, and how much value for each budget category is left
- **Inputs:** Expense array, budget array
- **Expected Output:** Corresponding metrics are correctly calculated and displayed to users
- **Result:** Pass

Expense categories

- **Description:** Unit test to render the expense metrics graph. This test validates that the graph correctly displays the top 5 expense categories by total cost and accurately shows the percentage of each category.
- **Inputs:** Expense array
- **Expected Output:** The metrics are correctly calculated and displayed to users
- **Result:** Pass

6.1.3 Manage resources

This section describes the unit tests for managing the CRUD functions of budgets, incomes and expenses.

Manage Expenses

- **Description:** Unit tests to display user expenses and allow users to add expense
- **Inputs:** Expenses array

- **Expected Output:** Expenses correctly rendered and displayed, users are able enter expense information and then save corresponding expense information
- **Result:** Pass

Manage Budget

- **Description:** Unit tests to verify left over budget (or over budget if users have expensed more than their budget), total expenses, and budget for each expense category. Also will validate actions for deleting, editing and adding new budget.
- **Inputs:** Expenses array, budgets array
- **Expected Output:** Corresponding remaining budget displayed along with total expenses. Budget for each expense category correctly rendered
- **Result:** Pass

Manage Income

- **Description:** Unit tests to validate user income(s) and how much more users are able to budget (if applicable). Also validates actions for deleting, editing and adding new income.
- **Inputs:** Incomes array, budgets array
- **Expected Output:** Corresponding income displayed along with the remaining budget. Users are able to enter income information and save it correctly. Actions for deleting, editing, and adding new income are validated.
- **Result:** Pass

7 Changes Due to Testing

[This section should highlight how feedback from the users and from the supervisor (when one exists) shaped the final product. In particular the feedback from the Rev 0 demo to the supervisor (or to potential users) should be highlighted. —SS]

8 Automated Testing

9 Trace to Requirements

10 Trace to Modules

11 Code Coverage Metrics

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