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
03/10/2025	0.1	Rev0
...	...	...

2 Symbols, Abbreviations and Acronyms

Refer to Section 1.3 of the Software Requirements Specification (SRS) document for the list of symbols, abbreviations, and acronyms.

In addition, the following abbreviations are used in this document:

Table 1: Symbols, Abbreviations, and Acronyms

symbol	**description**
V&V	Verification and Validation
UI	User Interface
OCR	Optical Character Recognition
SQL	Structured Query Language
GDPR	General Data Protection Regulation

Contents

List of Tables

This document reports the results of the Verification and Validation (V&V) process for the Plutossoftware. The V&V plan is documented in the Verification and Validation Plan document.

3 Functional Requirements Evaluation

The functional system tests can be found in Section 4.1 of the Verification and Validation Plan document. These tests are all performed manually.

Table 2: Functional Requirements Evaluation

Test ID	Pass/Fail	Comments
test-UAM-1	Pass	
test-UAM-2	Pass	
test-UAM-3	Pass	
test-UAM-4	Fail	Not yet implemented
test-UAM-5	Pass	
test-UAM-6	Pass	
test-IP-1	Pass	
test-IP-2	Pass	
test-IP-3	Fail	Not yet implemented
test-MIS-1	Pass	
test-DM-1	Pass	
test-DM-2	Pass	
test-RS-1	Pass	
test-RS-2	Pass	
test-RS-3	Pass	
test-FT-1	Fail	Not yet implemented
test-FT-2	Pass	
test-FT-3	Fail	Not yet implemented

4 Nonfunctional Requirements Evaluation

The nonfunctional system tests can be found in Section 4.2 of the Verification and Validation Plan document. **Tests without comments are performed as described in the plan.**

Table 3: Nonfunctional Requirements Evaluation

Test ID	Pass/Fail	Comments
test-ACC-1	Pass	Actual output; accuracy is 47/57 = 82.46%, which meets the threshold of 80%. Test can be found here.
test-ACC-2	Pass	By manually comparing the input (set of receipt images) with the resulting output, and calculating the accuracy as descripted in the V&V Plan, current accuracy is 80%, which meets the threshold of 80%. • *foodbasics_1.jpg*: 13.5/15 = 90% • *foodbasics_2.jpg*: 7/9 = 77.78% • *walmart_1.jpg*: 7/10 = 70% • *costco_1.jpg*: 18.5/23 = 76.09% • Overall accuracy: 46/57 = 80.70%
test-ACC-3	Pass	
test-ACC-4	Pass	
test-ACC-5	Pass	
test-PERF-1	Pass	
test-PERF-2	Pass	
test-PERF-3	Fail	Load testing has not yet been performed

test-USAB-1	Pass	
test-USAB-2	Pass	
test-USAB-3	Pass	
test-USAB-4	Pass	
test-SEC-1	Pass	
test-MTB-1	Pass	System stability has been tested, but application is not backward compatible since it is still under active development.
test-MTB-2	Pass	
test-MTB-3	Pass	
test-PORT-1	Pass	
test-PORT-2	Pass	
test-PORT-3	Pass	
test-REUS-1	Pass	Code walkthrough/review was performed with the team. See meeting minutes.
test-REUS-2	Pass	See REUS-1
test-UND-1	Pass	See REUS-1
test-UND-2	Pass	
test-UND-3	Pass	
test-LEGAL-1	Fail	The application is still under active development, so it is still using the testing environment and not all security features are active.

5 Comparison to Existing Implementation

This section is not applicable.

6 Unit Testing

6.1 Front-end unit tests

All front-end unit tests can be found in the test directory
Refer to Table 4 for unit test traceability table.

Table 4: Unit Testing Table

Test	**Testing plan**
test-UAM-1: Account creation	
test-UAM-2: User login	
test-UAM-3: User logout	
test-UAM-4: Account update	
test-UAM-5: Authorization access	
test-UAM-6: Password reset	Manual testing
test-IP-1: Image upload	Manual testing
test-IP-2: Image preview	
test-IP-3: Image upload file size limit	
test-MIS-1: Manual input expense	AddExpenseView.test.tsx, AddExpenseModal.test.tsx
test-FT-1: View spending history and trends	ExpensesList.test.tsx, HomePageMetricsBox.test.tsx, SpendingDetails.test.tsx
test-FT-2: Set and track budget	BudgetBoxDetails.test.tsx, MyBudgetsBox.test.tsx, NewBudgetModal.test.tsx
test-FT-3: Notification when user approaching limit	Not implemented

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

Automated testing is performed using Pytest for the backend and Jest for the frontend. The tests are run automatically on each push to the repository, as part of our continuous integration pipeline. Frontend tests can be found here and backend tests can be found here.

9 Trace to Requirements

A traceability matrix between test cases and requirements can be found in this Excel sheet.

10 Trace to Modules

A traceability matrix between test cases and modules can be found in this Excel sheet.

11 Code Coverage Metrics

Backend coverage is 80% according to pytest coverage report. See Figure 1.

```
---------- coverage: platform win32, python 3.11.1-final-0 -----------
Name                                                Stmts   Miss  Cover
-----------------------------------------------------------------------
__init__.py                                            15      0   100%
app.py                                                  8      1    88%
controllers\__init__.py                                 3      0   100%
controllers\budget\__init__.py                          0      0   100%
controllers\budget\budget_controller.py                15      0   100%
controllers\expenses\__init__.py                        0      0   100%
controllers\expenses\expenses_controller.py            26      7    73%
controllers\incomes\__init__.py                         0      0   100%
controllers\incomes\incomes_controller.py              15      0   100%
controllers\users\__init__.py                           0      0   100%
controllers\users\users_controller.py                  14      0   100%
daos\__init__.py                                        3      0   100%
daos\budget\__init__.py                                 0      0   100%
daos\budget\budget_dao.py                              25      3    88%
daos\expenses\__init__.py                               0      0   100%
daos\expenses\expenses_dao.py                          61     24    61%
daos\incomes\__init__.py                                0      0   100%
daos\incomes\incomes_dao.py                            22      0   100%
daos\users\__init__.py                                  0      0   100%
daos\users\users_dao.py                                20      0   100%
db.py                                                   9      1    89%
imageProcessing\__init__.py                             1      0   100%
imageProcessing\categorization.py                      25      0   100%
imageProcessing\process.py                             93     65    30%
imageProcessing\utils.py                               41     34    17%
models\__init__.py                                      4      0   100%
models\budget\__init__.py                               0      0   100%
models\budget\budget.py                                 9      0   100%
models\expenses\__init__.py                             0      0   100%
models\expenses\expense.py                             15      0   100%
models\expenses\receipt.py                             12      7    42%
models\incomes\__init__.py                              0      0   100%
models\incomes\income.py                               11      0   100%
models\users\__init__.py                                0      0   100%
models\users\user.py                                   11      0   100%
routes\__init__.py                                      3      0   100%
routes\budget\__init__.py                               0      0   100%
routes\budget\budgetRoutes.py                          16      0   100%
routes\expenses\__init__.py                             0      0   100%
routes\expenses\expensesRoutes.py                      24      2    92%
routes\incomes\__init__.py                              0      0   100%
routes\incomes\incomesRoutes.py                        16      0   100%
routes\users\__init__.py                                0      0   100%
routes\users\usersRoutes.py                            16      0   100%
tests\__init__.py                                       0      0   100%
tests\budget\__init__.py                                0      0   100%
tests\budget\test_budget.py                            55      0   100%
tests\expense\__init__.py                               0      0   100%
tests\expense\test_expense.py                          55      0   100%
tests\imageProcessing\__init__.py                       0      0   100%
tests\imageProcessing\test_categorization.py           25      0   100%
tests\imageProcessing\test_parsing.py                  17      7    59%
tests\income\__init__.py                                0      0   100%
tests\income\test_income.py                            48      0   100%
tests\users\__init__.py                                 0      0   100%
tests\users\test_users.py                              38      0   100%
-----------------------------------------------------------------------
TOTAL                                                 771    151    80%
```

Figure 1: Backend Code Coverage Metrics

Frontend coverage metrics is calculated using Jest. See Figure 2.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	53.87	42.85	49.32	55.2	
client	100	100	100	100	
constants.ts	100	100	100	100	
client/api	100	100	100	100	
api.ts	100	100	100	100	
client/assets/icons	75	25	50	75	
AddCircleIcon.js	100	50	100	100	4
CameraIcon.js	100	50	100	100	4
CancelIcon.js	100	50	100	100	3
CartIcon.js	50	0	0	50	4
CoinIcon.js	100	66.66	100	100	3
ComputerIcon.js	50	0	0	50	4
HouseIcon.js	50	0	0	50	4
IncomeIcon.js	100	50	100	100	4
LaundryIcon.js	50	0	0	50	4
MoviesIcon.js	50	0	0	50	4
PhotoLibraryIcon.js	100	50	100	100	4
WifiIcon.js	50	0	0	50	4
client/components/common	61.81	55.43	53.26	63.48	
BudgetBox.tsx	100	100	100	100	
BudgetBoxDetails.tsx	91.66	96.42	83.33	91.66	32,159-160
DefaultLayout.tsx	100	100	100	100	
EntrySource.tsx	100	100	100	100	
ExpensesList.tsx	100	83.33	100	100	59
HomePageButton.tsx	100	100	100	100	
HomePageMetricsBox.tsx	100	100	100	100	
IncomeBox.tsx	46.66	100	33.33	50	24-26,34-37,53-66
MyBudgetsBox.tsx	43.75	100	22.22	46.66	24-26,34-37,56-81
NewBudgetModal.tsx	31.81	21.42	16.66	34.14	54-58,65,69-70,74,78-112,139
NewIncomeModal.tsx	33.92	33.33	26.66	35.84	58-66,71-75,79,83-84,88,92-128,194
SpendingDetails.tsx	61.53	0	36.36	64	34-38,43,48,55,59-62
client/components/view	60.24	25	57.14	64.47	
AddExpenseModal.tsx	94.11	100	66.66	94.11	72
AddExpenseView.tsx	42.85	6.25	42.85	46.93	52,67,73,82-131
DisplayExpenseItem.tsx	100	100	100	100	
client/contexts	57.14	100	33.33	50	
UserContext.tsx	57.14	100	33.33	50	31-34
client/services	7.35	0	7.69	7.35	
budgetService.js	4.76	100	0	4.76	5-44
expensesService.js	11.53	0	20	11.53	15-67
incomeService.js	4.76	100	0	4.76	5-40
client/utils	50	0	57.14	52.17	
util.ts	50	0	57.14	52.17	16-20,31-37,44-45

Figure 2: Frontend 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.)