Verification and Validation Report: Plutos

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

April 4, 2025

1 Revision History

Date	Version	Notes
03/10/2025	0.1	Rev0
...	...	...

2 Symbols, Abbreviations and Acronyms

Refer to the Software Requirements Specification (SRS) document for the list of abbreviations and acronyms (Section 1.3) and the list of symbolic constants (Section 10).

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

List of Figures

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	Pass	
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	Pass	
test-FT-2	Pass	
test-FT-3	Fail	Not yet implemented

The User Account Management tests evaluate the core functionalities related to user account creation, login, logout, updates, authorization access, and password reset. Inputs for these tests typically consist of user-provided data such as names, emails, passwords, and session information. The expected outputs include confirmation messages and access to the user dashboard, with a pass being defined by the successful execution of each function without errors, ensuring a seamless user experience.

The Image Processing tests assess the application's ability to handle image uploads, previews, and file size limitations. Inputs for these tests include various image files, while the outputs are the successful display of uploaded images and adherence to file size constraints. A pass is determined by the system's ability to process the images correctly and provide appropriate feedback to the user.

The Manual Expense Input test assesses the application's ability to handle manual input of expenses. The input for this test involves user-entered financial data, while the output is the successful recording and display of the entered expenses. A pass is determined by the system's ability to accurately capture and reflect the user's input without any discrepancies.

The Database Management tests evaluate the application's functionality in managing and processing user data. Inputs for these tests consist of various data entries related to user expenses and financial records. The expected outputs are the correct storage and retrieval of this data, with a pass being defined by the system's ability to manage data effectively and provide accurate results when queried.

The Item Recognition and Categorization tests focus on the application's capability to generate reports based on user data. Inputs for these tests include user financial data and parameters for report generation. The outputs are the generated reports that summarize spending trends and financial insights. A pass is achieved when the reports are accurately produced and reflect the correct data as per user specifications.

The Financial Tracking tests are designed to evaluate the application's capabilities in tracking spending history and creating budgets for users. Inputs for these tests consist of user-generated financial data, such as expenses and budget parameters. The outputs include visual representations of spending trends and budget trackers. A pass for these tests is achieved when the application accurately reflects the user's financial data ensuring effective financial management for users.

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 *CATEGORIZATION_ACCURACY*%. 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 *OCR_ACCURACY*%. • *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	Initial load testing has been conducted, but additional automated testing is necessary to achieve comprehensive performance coverage. Further load testing was deprioritized due to time constraints.
test-USAB-1	Pass	Testers successfully carried out tasks related to the app's functionality and account creation while adhering to the established guidelines. Manual errors were introduced to evaluate the system's ability to recover from errors.
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.

The Accuracy tests evaluate the application's ability to correctly process and categorize financial data. Inputs for these tests include various datasets, such as receipt images and manual expense entries. The expected outputs are accurate categorizations and data extractions that meet predefined accuracy thresholds. A pass is determined by the system's ability to achieve the required levels of accuracy in processing and categorizing expenses, ensuring reliable financial insights for users.

The Performance tests assess the application's responsiveness and stability under various conditions, including normal and peak usage scenarios. Inputs for these tests consist of simulated user interactions and load conditions. The outputs are performance metrics such as response times and system throughput. A pass is determined by the application meeting predefined performance benchmarks, ensuring it can handle expected user loads without degradation in performance.

The Usability tests focus on the overall user experience, evaluating how easily users can navigate and utilize the application. Inputs for these tests include user tasks and scenarios designed to assess the intuitiveness of the interface. The expected outputs are user feedback and task completion rates. A pass is achieved when users can complete tasks efficiently based on given instructions and report a positive experience, indicating that the application is user-friendly and intuitive. Further feedback should be provided by additional usability testing that may be done now or in future versions of the application.

The Security test evaluates the application's ability to protect user data and maintain confidentiality. Inputs for this test include various security scenarios, including but not limited to SQL injection attacks through input fields and session hijacking through cookies. The output is an assessment of the application's security measures, including data encryption and access controls. A pass is determined by the system's ability to effectively mitigate security risks and demonstrate compliance with relevant security standards.

The Maintainability tests assess the ease with which the application can

be updated and maintained over time. Inputs for these tests consist of code reviews and maintenance scenarios. The expected outputs are evaluations of the application's architecture and code quality. A pass is achieved when the application demonstrates a clear structure that facilitates easy updates and maintenance, minimizing the risk of introducing new issues during changes.

The Portability tests assess the application's ability to function across different platforms and environments. Inputs for these tests consist of various operating systems, devices, and configurations on which the application is expected to run. The outputs are evaluations of the application's performance and functionality in these different environments. A pass is achieved when the application and its required API's operates seamlessly across multiple platforms without significant issues, ensuring that users can access the application regardless of their device or operating system.

The Reusability tests evaluate the extent to which components of the application can be reused in other projects or contexts. Inputs for these tests include code modules and design elements. The outputs are assessments of the modularity and adaptability of the components. A pass is determined by the system's ability to demonstrate reusable components that can be easily integrated into other applications or systems, promoting efficient development practices. These can be assessed through code reviews by the development team and third parties.

The Understandability tests focus on the clarity and comprehensibility of the application's documentation and user interface. Inputs for these tests consist of user feedback and documentation reviews. The expected outputs are assessments of how easily users can understand and navigate the application. A pass is achieved when users report a clear understanding of the application's features and functionalities based on the provided documentation and interface design.

The Regulatory tests evaluate the application's compliance with relevant laws and regulations, particularly concerning data protection and user privacy. Inputs for these tests include legal requirements and guidelines that the application must adhere to. The expected outputs are assessments of the application's compliance status and any identified legal risks. A pass is determined by the system's ability to demonstrate adherence to legal standards and the implementation of necessary features to protect user data in accordance with applicable laws.

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: Test traceability Table

Test	**Testing plan**
test-UAM-1: Account creation	test_users.py
test-UAM-2: User login	Manual testing
test-UAM-3: User logout	Manual testing
test-UAM-4: Account update	test_users.py
test-UAM-5: Authorization access	test_users.py
test-UAM-6: Password reset	Manual testing
test-IP-1: Image upload	Manual testing
test-IP-2: Image preview	Manual testing
test-IP-3: Image upload file size limit	Manual testing
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

In accordance to the feedback provided through the Rev 0 demo and other users, the application will undergo small modifications to address these responses. A key component that was brought up during the Rev 0 demo was the importance for users to be able to modify the data entries within

the app with relative ease. Following the initial development schedule, the categorization model is constantly being refined and has a possibility of misclassification. The system should allow the users to make changes to these inputs manually to improve overall usability. A few other suggestions revolving around usability were small features such as autofilling item entries and providing adaptable user metrics.

One small change was made to the requirements regarding accuracy. The accuracy of the categorization model, which was initially targeted to reach around 90%, will now only require an 80% level of accuracy. This adjustment was made to enhance the model's generalization, which in turn reduces ambiguity during data classification. A lower accuracy threshold encourages a more flexible model that avoids overfitting to the training data and can better adapt to unseen, real-world inputs. This is especially important when dealing with noisy, inconsistent, or highly varied data such as scanned receipts or user-generated content, where exact matches are less common. Furthermore, striving for an extremely high accuracy could lead to longer development times and significantly increased computational complexity without proportionate gains in usability. For the scope of this project, the team decided that by focussing on balancing accuracy and generalization, the model can remain lightweight and responsive while still offering meaningful performance. Additionally, the ability for users to modify misclassified entries will further help address any discrepancies and improve the overall accuracy over time through user feedback and potential retraining mechanisms.

At the moment, the core functional components of the application have been tested by a select group of users, providing feedback on its performance and usability. However, we have yet to conduct larger-scale usability testing to assess the flow of the entire app. Once we finalize the implementation of the smaller features, such as autofill options and adaptable user metrics, we plan to conduct further tests to optimize usability. This will involve a broader range of users to ensure that every criteria set for the system at the beginning of the project was met. By focusing on the complete flow of the application, we aim to have testers evaluate all the smaller aspects of the app and how they work cohesively with the main functional components, ensuring that each part functions seamlessly together for an optimal user experience.

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

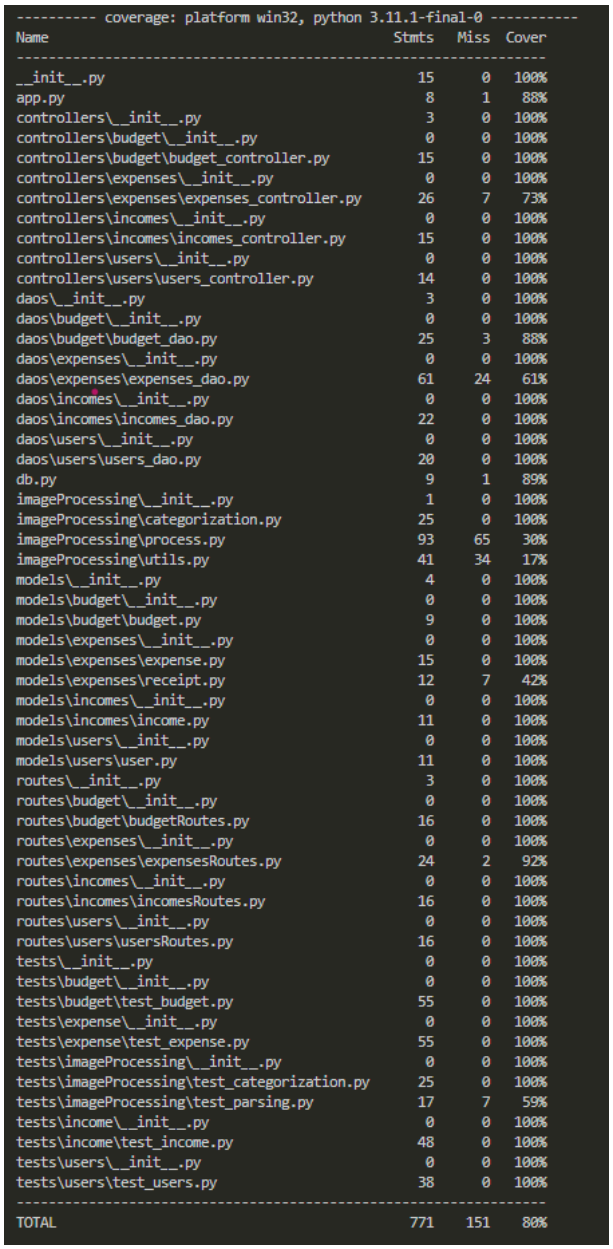
A traceability matrix between test cases and requirements can be found in this Excel sheet.

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?
   While writing the VnV report, it was noteworthy that the collaborative effort among team members was highly effective. As we assigned each member a different area of testing, we ensured that the results brought into our meetings were insightful and understood by all of the other group members. This collaboration not only enhanced the quality of the content but also ensured that all perspectives in response to the feedback were considered. Finally, the thorough testing conducted prior to writing the report provided a good foundation to plan out the future steps of development. This allowed us to present concrete evidence of our findings, validating the findings outlined in the report.

2. What pain points did you experience during this deliverable, and how did you resolve them?
   One challenge we faced was managing all of the feedback received from Rev 0 as well as the feedback provided from the users. There were some differing opinions as to how we should address these responses moving forward. We held a meeting to discuss the feedback together, allowing us to prioritize the most important points and agree on how to approach them within our implementation. Another challenge was ensuring the

technical details, especially in the testing and results sections, were accurate. Some team members had more knowledge of their assigned testing areas, which could create gaps. To mitigate these gaps we had several meetings where team members shared their expertise, helping improve our understanding and ensure the report was well written and that all group members had a solid understanding of all testing aspects.

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?
   Several sections of the VnV report were shaped by feedback from clients and peers. For example, the "Changes Due to Testing" section was influenced by the input provided by the instructor and TA during the Rev 0 demo, which highlighted the need for better usability features. This led us to adjust our focus to user-centered features. Peer reviews also helped refine our assessment of functional and nonfunctional requirements. On the contrary, some parts of the report, like the technical details of testing methods and code coverage metrics, were based on internal discussions and research. These sections relied on our team's expertise and the VnV Plan, as they required a deeper understanding of the system's technical aspects, while the feedback focused more on the project's overall direction.

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 content of the VnV report differed from the original plan due to continuous feedback, unforeseen technical challenges, and a greater emphasis on features revolving around usability. We had to modify our approach by expanding user-focused testing, introducing additional test cases, and addressing unexpected issues. These changes occurred as we gained new insights and encountered limitations in our initial assumptions. To improve future planning, we can allocate more buffer

time, conduct early individual pilot tests for core functional features, and allow for more flexibility in our initial VnV plan. Despite these deviations, our team effectively adapted to ensure a thorough validation process with sufficient coverage to guide further development.