Verification and Validation Report: Grocery Spending Tracker

Team 1, JARS
Jason Nam
Allan Fang
Ryan Yeh
Sawyer Tang

March 6, 2024

1 Revision History

Date	Version	Notes
06/03/2024	1.0	Initial version of the VnV Report

2 Symbols, Abbreviations and Acronyms

symbol	**description**
T	Test
SRS	Software Requirements Specification
MIS	Module Interface Specification
VnV	Verfication and Validation
UI	User Interface
UX	User Experience
ms	milliseconds
GB	gigabyte
MB	megabyte
OCR	Optical Character Recognition
CRUD	Create, Read, Update, Delete
JWT	JSON Web Token

Contents

List of Tables

List of Figures

This document will primarily focus on the test results of the verification and validation process for the Grocery Spending Tracker application. In particular, the results of the Functional Requirement System Tests, Nonfunctional Requirement System Tests, and Unit Tests will be covered. Detailed information regarding the Functional Requirements and Nonfunctional Requirements can be found in the SRS. Furthermore, unit tests will focus on the individual modules of the system which can be found in the MIS. Finally, detailed information regarding the overall test plan can be found in the VnV Plan.

In addition to test results, this document will also go over feedback from stakeholders and changes due to testing, the team's approach for automated testing, traceability between tests and requirements/modules, as well as overall code coverage.

3 Functional Requirements Evaluation

This section will cover the test results from the Functional Requirement System Tests performed for the verification and validation of the Grocery Spending Tracker application.

3.1 User Authentication

The following sections pertain to all tests that have to do with the user profiles, and access to the system.

Table 1: User Authentication Testing Summary

Test ID	**Result**	**Details**
FRT-FR1-1	✓	Takes 3 steps to reach landing page (enter email, enter password, submit)
FRT-FR1-2	✓	100% positive response (graded 4 or above out of 5) from survey (Section 11.1.1 Appendix)
FRT-FR2-1	✓	

FRT-FR2-2	✓	
FRT-FR2-4	✓	
FRT-FR2-5	✓	
FRT-FR2-6	✓	
FRT-FR2-7	✓	

FRT-FR1-1 to FRT-FR2-7 were all manually performed tests to verify the login and authentication process. FRT-FR1-1 to FRT-FR1-2 were to verify the actual login process from an actual use case scenario. The remaining tests verify different input cases for authentication such as correct email but incorrect password and new user scenarios.

3.2 Receipt Scanning

These tests pertain to all the functional requirements related to the scanning of receipts and inputting grocery data.

Table 2: Receipt Scanning Testing Summary

Test ID	**Result**	**Details**
FRT-FR3-1	✓	95% accuracy from manual testing
FRT-FR11-1	✓	Allows retake without prompting
FRT-FR11-2	✓	
FRT-FR11-3	✓	

The above tests were all manually performed tests primarily pertaining to the receipt scanning process. In terms of the tests themselves, FRT-FR3-1 involved measuring accuracy of the OCR and FRT-FR11-1 to FRT-FR11-3 verify different use cases that can occur while trying to input a receipt, such as handling camera permissions and manual entry of receipt data.

3.3 Account Goals and Preferences

These tests pertain to all the functional requirements related to the user's account goals and preferences.

Table 3: Account Goals and Preferences Testing Summary

Test ID	**Result**	**Details**
FRT-FR5-1	✓	
FRT-FR5-2	X	To be completed for Revision 1

FRT-FR5-1 and FRT-FR5-2 are manual tests verifying the User Goals functionality of the app. FRT-FR5-1 tests the ability to add new spending goals and FRT-FR5-2 tests the ability to update existing spending goals.

3.4 System Behaviour and Recommendations

These tests pertain to all the functional requirements related to how the system responds to the data that is provided to it. This includes receiving purchase data, grocery recommendations, and grocery item interfaces.

Table 4: System Behaviour and Recommendations Testing Summary

Test ID	**Result**	**Details**
FRT-FR7-1	✓	
FRT-FR7-2	✓	
FRT-FR7-3	✓	
FRT-FR8-1	✓	

FRT-FR8-2	X	50% positive response (graded 4 or above out of 5) from survey (Section 11.1.2 Appendix), users would like to see higher accuracy on relevance to user
FRT-FR9-1	✓	
FRT-FR9-2	✓	
FRT-FR10-1	✓	
FRT-FR10-2	✓	

The above tests were all manually performed and verifies how the system responds to data input, such as with the Classification Engine and Recommendations. FRT-FR7-1 to FRT-FR7-3 test the system's ability to take an item from the receipt and match it to an actual grocery product (including full product name, an image, etc.) before placing it into the database. FRT-FR8-1 to FRT-FR9-2 test the Recommendation functionality of the application in different use cases, such as suggesting cheaper products related to ones the user has purchased in the past. Finally, FRT-FR10-1 and FRT-FR10-2 test that the user is able to verify all data that has been submitted into the system, such as review receipts before submission and viewing previously submitted receipts.

4 Nonfunctional Requirements Evaluation

This section goes over the test results from the Nonfunctional Requirement System Tests as specified in the VnV Plan.

4.1 Look and Feel

This section provides a brief overview of the Look and Feel test results from the VnV Plan.

Table 5: Look and Feel Testing Summary

Test ID	**Result**

NFRT-LF1	✓
NFRT-LF2	✓
NFRT-LF4	✓

All the above tests were manually performed by external users under the supervision of the team. NFRT-LF1 involved the team walking users through the application and giving the users some time to go through the pages on their own. Afterwards, users were given surveys where results were positive. All ratings regarding the UI and UX were an 8 or higher on average and general feedback was positive with some minor complaints regarding blandness in terms of UI design. Altogether, users reacted positively to the general UI/UX of the application which is within the team's 80% overall acceptance rate for various aspects of the Look and Feel as documented in the SRS. More detailed information regarding the survey can be found in Section 11.2 of this document. NFRT-LF2 involved users being given a list of features and a list of icons. They would be tasked with matching the features to the icons which all users were able to do without difficulty. Finally, NFRT-LF4 was testing that the user is notified if they lose internet connection while using the application.

4.2 Usability

This section provides a brief overview of the Usability test results from the VnV Plan.

Table 6: Usability Testing Summary

Test ID	**Result**
NFRT-UH1	✓
NFRT-UH2	✓
NFRT-UH3	✓
NFRT-UH5	✓

NFRT-UH6	✓

All the above tests were by external users with supervision and instruction from members of the team. NFRT-UH1 was a test to verify how easy the application is to learn. This was done by having them go through the application on their own and they were then asked to navigate to different pages. NFRT-UH2 had users testing whether processes on the application were reversible/cancellable in terms of the receipt submission process. NFRT-UH3 and NFRT-UH5 were accessibility tests in regards to adjustable text size of the application. Finally, NFRT-UH6 was a survey where users were asked several questions related to the application's usability. This survey and its results can be found in Section 11.3 of this document. In general though, survey results were positive with some issues that have been noted down and the team has plans to work on, for example, better instructions for receipt capture.

4.3 Performance

This section covers the test results from Performance tests, as described in the VnV Plan.

Table 7: Performance Testing Summary

Test ID	**Result**
NFRT-PR1	✓
NFRT-PR2	✓
NFRT-PR3	✓
NFRT-PR4	✓
NFRT-PR5	✓
NFRT-PR6	✓

All above tests were performed by the team using various tools like Flutter Performance Profiling and Postman as described in the VnV Plan. NFRT-

PR1 focused on testing render performance of the application. Overall, render performance was consistently below 17 ms and at worst, never exceeded 60 ms. These results fall well below the team's upper-limit of render times which was set at 1 second according to the SRS. Below is a snapshot of the Flutter Profiling graph that shows render times for each frame, with 52 ms being the highest in this instance:

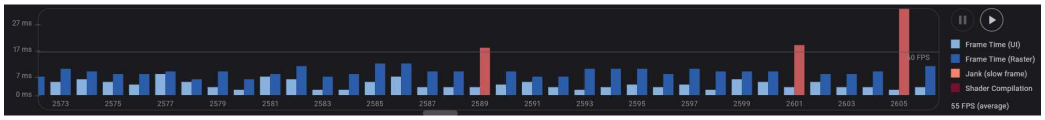

Figure 1: Frontend Render Performance Graph

Similarly, NFRT-PR2 focused on general performance on the backend, in particular, response times from the different endpoints. Various endpoints were accessed about 10 times using Postman and results showed that backend performance was within expectation for the project. Most of the responses were within 200 ms with the worst being below 600 ms. The team's expectations for backend performance were for responses to be below 2 seconds on average according to the SRS, which results from this test show. Below is a graphical representation of the trials performed and the endpoints tested:

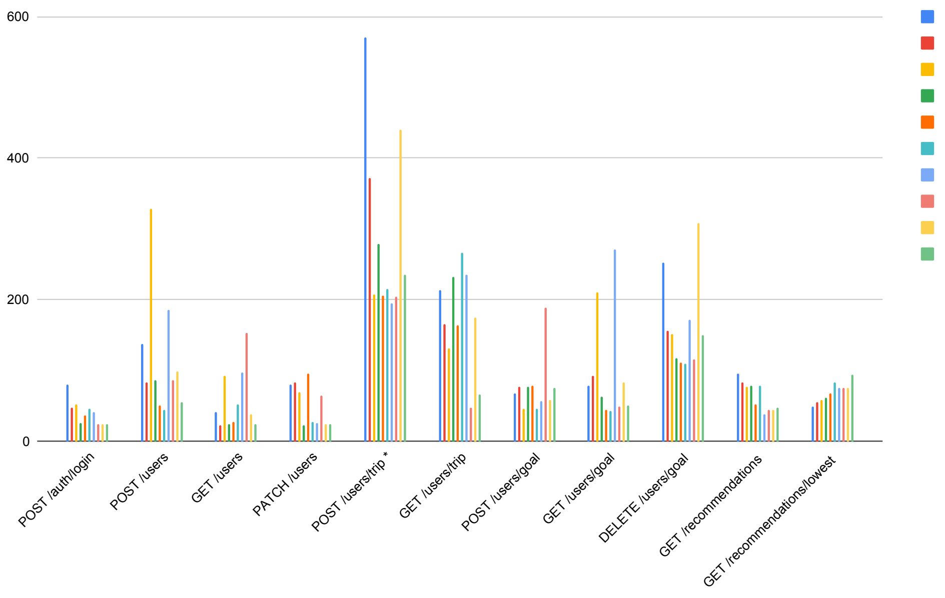

Figure 2: Backend Endpoint Performance Graph

NFRT-PR3 tested the robustness of the backend and involved passing faulty data to see if anything would cause the system to produce unwanted results. NFRT-PR4 and NFRT-PR5 tested the correctness of the system by verifying how often users need to correct the OCR and the accuracy of the spending data being returned. Both of these results were above the expected values, with user correction rate being 5% and 0.1% relative error on spending data. Finally, NFRT-PR6 verifies that the application is within reasonable size, taking up less than 1GB of the user's device storage. Actual storage consumption was between 300 to 400 MB depending on application usage.

4.4 Cultural

This section describes test results for Cultural tests performed as part of the verification and validation process.

Table 8: Cultural Testing Summary

Test ID	Result
NFRT-CUR2	✓

This test was performed internally by team members. NFRT-CUR2 verifies the Classification Engine and checks that if a new item is submitted into the database, the item's product information can be properly retrieved.

4.5 Operation and Environment

This section describes test results for Operation and Environment tests performed as documented in the VnV Plan.

Table 9: Operation and Environment Testing Summary

Test ID	Results
NFRT-OER1	✓
NFRT-OER2	✓
NFRT-OER3	✓

These tests were performed internally by team members. NFRT-OER1 verifies the compatibility of the application on devices using Android 13 or higher. NFRT-OER2 tests that the application interfaces with peripherals properly such as the device camera. Finally, NFRT-OER3 verifies the CI/CD process and ensures proper unit testing is performed for updates.

4.6 Security

This section covers the test results for Security tests performed as documented in the VnV Plan.

Table 10: Security Testing Summary

Test ID	Results

NFRT-SR2	✓

This test was performed internally by team members. NFRT-SR2 is a test that verifies protections are properly put in place such that a user cannot modify the database directly (i.e., without using the endpoint).

5 Unit Testing

This section will go over the test results from the unit tests made as part of the verification and validation process.

5.1 Receipt Extraction Module

The section provides a brief overview of the unit tests performed and their results for the Receipt Extraction Module.

Table 11: Receipt Extraction Unit Testing Summary

Test ID	**Result**	**Test ID**	**Result**	**Test ID**	**Result**
FRT-M3-1	✓	FRT-M3-2	✓	FRT-M3-3	✓
FRT-M3-4	✓	FRT-M3-5	✓	FRT-M3-6	✓
FRT-M3-7	✓	FRT-M3-8	✓	FRT-M3-9	✓
FRT-M3-10	✓	FRT-M3-11	✓	FRT-M3-12	✓
FRT-M3-13	✓	FRT-M3-14	✓	FRT-M3-15	✓
FRT-M3-16	✓	FRT-M3-17	✓	FRT-M3-18	✓
FRT-M3-19	✓	FRT-M3-20	✓	FRT-M3-21	✓
FRT-M3-22	✓	FRT-M3-23	✓	FRT-M3-24	✓
FRT-M3-25	✓	FRT-M3-26	✓		

To summarize, tests FRT-M3-1 to FRT-M3-26 are all frontend tests. These tests focus on the helper functions used to extract specific information when receiving a chunk of receipt text. FRT-M3-1 to FRT-M3-6 focus on date extraction with different formats, FRT-M3-7 to FRT-M3-13 test address extraction, FRT-M3-14 to FRT-M3-16 target grocery item extraction, FRT-M3-17 to FRT-M3-19 focus on subtotal cost extraction, and FRT-M3-20 to FRT-M3-22 tests total cost extraction. Finally, FRT-M3-23 to FRT-M3-24 and FRT-M3-25 to FRT-M3-26 test updating and JSON conversion for the Item and Grocery Trip objects respectively. In general, these tests consist of ideal use cases as well as some edge cases that may appear during use. For more detailed information regarding the Receipt Extraction Module unit tests, the testing file can be found here: receipt extraction unit tests.

5.2 User Analytics Module

The section provides a brief overview of the unit tests performed and their results for the User Analytics Module.

Table 12: User Analytics Unit Testing Summary

Test ID	**Result**	**Test ID**	**Result**	**Test ID**	**Result**
FRT-M5-1	✓	FRT-M5-2	✓	FRT-M5-3	✓
FRT-M5-4	✓	FRT-M5-5	✓	FRT-M5-6	✓
FRT-M5-7	✓	FRT-M5-8	✓		

Tests FRT-M5-1 to FRT-M5-5 are the front-end unit tests for the User Analytics Module. These tests focus on user budgeting goals and user purchase history. Details on these tests can be found in the following files: Front-end User Analytics Module Spending Goals Tests, Front-end User Analytics Module Spending History Tests.

Tests FRT-M5-6 to FRT-M5-8 are all back-end tests and can be found in the following file: Back-end User Analytics Module Controller Tests. These tests primarily focus on the endpoints responsible for CRUD operations related to user budget goals. Each of these tests have several sub-tests (a, b, c, etc.) that test the function's primary and edge-cases.

5.3 Users Module

The section provides a brief overview of the unit tests performed and their results for the Users Module.

Table 13: Authentication Unit Testing Summary

Test ID	Result	Test ID	Result	Test ID	Result
FRT-M6-1	✓	FRT-M6-2	✓	FRT-M6-3	✓
FRT-M6-4	✓	FRT-M6-5	✓	FRT-M6-6	✓

Tests FRT-M6-1 to FRT-M6-6 are all back-end tests and can be found in the following file: Back-end User Module Controller Tests. These tests focus on the endpoints related to CRUD operations for users on the application. Each test has several sub-tests (a, b, c, etc.) that test the function's primary and edge-cases.

5.4 Authentication Module

The section provides a brief overview of the unit tests performed and their results for the Authentication Module.

Table 14: Authentication Unit Testing Summary

Test ID	Result	Test ID	Result	Test ID	Result
FRT-M7-1	✓	FRT-M7-2	✓	FRT-M7-3	✓
FRT-M7-4	✓	FRT-M7-5	✓	FRT-M7-6	✓
FRT-M7-7	✓	FRT-M7-8	✓		

Tests FRT-M7-1 to FRT-M7-5 are the front-end unit tests for the Authentication Module. These tests focus on the user flows for authenticating user identity on the mobile app. Details on these tests can be found in the following file: Front-end Authentication Module Tests.

Tests FRT-M7-6 to FRT-M7-8 are all back-end tests and can be found in the following files: Back-end Authentication Module Controller Tests and

Back-end Authentication Module Utility Tests. FRT-M7-6 tests the login endpoint and FRT-M7-7 to FRT-M7-8 test the login utilities used such as returning a signed JWT token upon login. Each test has several sub-tests (a, b, c, etc.) that test the function's primary and edge-cases.

5.5 Recommendation Engine

The section provides a brief overview of the unit tests performed and their results for the Recommendation Module.

Table 15: Recommendation Unit Testing Summary

Test ID	**Result**	**Test ID**	**Result**	**Test ID**	**Result**
FRT-M8-1	✓	FRT-M8-2	✓		

Tests FRT-M8-1 and FRT-M8-2 are both back-end tests and can be found in the following file: Back-end Recommendation Module Controller Tests. FRT-M8-1 focuses on the endpoint for returning recommendations based on a user's frequency of purchase and FRT-M8-2 focuses on the endpoint for returning recommendations based on low product prices. Each test has several sub-tests (a, b, c, etc) associated that test the function's primary and edge-cases.

5.6 Classification Engine

The section provides a brief overview of the unit tests performed and their results for the Classification Module.

Table 16: Classification Unit Testing Summary

Test ID	**Result**	**Test ID**	**Result**	**Test ID**	**Result**
FRT-M9-1	✓	FRT-M9-2	✓	FRT-M9-3	✓
FRT-M9-4	✓	FRT-M9-5	✓	FRT-M9-6	✓
FRT-M9-7	✓	FRT-M9-8	✓	FRT-M9-9	✓

FRT-M9-10	✓	FRT-M9-11	✓		

Tests FRT-M9-1 to FRT-M9-11 are all back-end tests and can be found in the following file: Back-end Classification Module Web Scraping Tests, Back-end Classification Module Item Matching Tests, Back-end Classification Module Main Test. FRT-M9-1 to FRT-M9-6 tests the ability to fetch/scrape for different details regarding existing and new grocery items such the product name and price. FRT-M9-7 to FRT-M9-9 verifies the ability to match and classify an item from the receipt to existing/new web scraped items. Finally, FRT-M9-10 to FRT-M9-11 test the system's ability to cache items to make the classification process faster. Each test has several sub-tests (a, b, c, etc) that tests the function's primary and edge-cases.

6 Changes Due to Testing

Prior to formal usability testing, user impressions of the Revision 0 version of the app generally criticized the usability of the application. Specifically, better communication with the user in terms of loading. The previous version of the application lacked animated loading indicators so for forms that had a server request attached to them, they believed the application to be frozen after submission. This resulted in cases where users would try submitting again producing a duplicate server request. This has since been resolved by adding various animated loading indicators and overlays to prevent double clicking a submission and better communicate to users when server requests were being processed. Some intuitiveness features were also added as a result of user feedback such as tap focus on the receipt capture page. Additionally, some feedback received from the Revision 0 demo was addressed in the current version of the application. This included more detailed information on the Goals page and matching address to a list of known addresses to improve usability/reduce user input.

During formal usability testing of the current application version, much of the feedback had to do with robustness of the application as opposed to functionality. In particular, the request for submitting a trip in some cases ended up taking close to 10 seconds due to the current Classification Engine implementation. As a result, users believed that there was something wrong with the application or that there was an error. In response to this, a backend fix is being worked on to reduce the response time which will be done prior to Revision 1. Additionally, some users had some issues with the receipt capture, not knowing that they had to get the receipt to occupy as much of the camera frame as possible for the best results, so an information screen on this page is being worked on to better help users with receipt capture which will also be finished prior to Revision 1. Outside of the UI/UX, users found that recommendations could be improved in terms of relating more to the user's purchase history which the team will try to do before Revision 1 as well. Finally, during Functional Requirement Testing, it was found that the application was missing the ability to update user budget goals which is also now scheduled to be done prior to Revision 1.

7 Automated Testing

On the frontend, all automated tests are carried out using Flutter's built-in testing package. All tests are located in the "test" folder which is organized by components being tested (found here). For example, the *profile_provider_test* file executes tests related to the provider for profile functionality. These tests are executed twice during the development workflow: the first is locally while working on individual feature branches and the second is on merges/pushes to the "dev" branch of the frontend repository. The first execution is done manually using *flutter test* in the command line and the second is done automatically using *Github Actions*.

On the backend, automated tests are done using Mocha, Chai, and Sinon. Tests are located in the "tests" folder which is then organized by component and the module being tested (found here). As an example, the *userController.analytics.test* file contains unit tests related to the Analytics Module functionality in the user controller. These tests are automatically executed when code is pushed to the "master" branch of the backend repository as part of the deployment process. This process is handled by *Github Actions*.

8 Trace to Requirements

Table 17: Traceability Between Functional System Test Cases and Functional Requirements

Functional Requirement ID	**Test ID**								
	FR1	**FR2**	**FR3**	**FR5**	**FR7**	**FR8**	**FR9**	**FR10**	**FR11**
FRT-FR1-1	X								
FRT-FR1-2	X								
FRT-FR2-1		X							
FRT-FR2-2		X							
FRT-FR2-4		X							
FRT-FR2-5		X							
FRT-FR2-6		X							
FRT-FR2-7		X							
FRT-FR3-1			X						
FRT-FR11-1									X
FRT-FR11-2									X
FRT FR11-3									X
FRT-FR5-1				X					
FRT-FR5-2				X					
FRT-FR7-1					X				

FRT-FR7-2	X			
FRT-FR7-3	X			
FRT-FR8-1		X	X	
FRT-FR8-2		X		
FRT-FR9-1			X	
FRT-FR9-2			X	
FRT-FR10-1				X
FRT-FR10-2				X

Table 18: Traceability Between Nonfunctional System Test Cases and Look and Feel Requirements

NFR ID	**Test ID**				
	AR1	**AR2**	**SR1**	**SR2**	**SR3**
NFRT-LF1	X		X	X	X
NFRT-LF2		X			
NFRT-LF4					X

Table 19: Traceability Between Nonfunctional System Test Cases and Usability and Humanity Requirements

NFR ID	Test ID								
	EUR1	**EUR2**	**EUR3**	**PIR1**	**LR1**	**LR3**	**UPR1**	**UPR2**	**ACR1**
NFRT-UH1						X			
NFRT-UH2		X							
NFRT-UH3									X
NFRT-UH5	X								
NFRT-UH6			X	X	X		X	X	X

Table 20: Traceability Between Nonfunctional System Test Cases and Performance Requirements

NFR ID	Test ID					
	SLR1	**SLR2**	**PAR1**	**PAR2**	**RFR1**	**CR1**
NFRT-PR1		X				
NFRT-PR2	X					
NFRT-PR3					X	
NFRT-PR4			X			
NFRT-PR5				X		
NFRT-PR6						X

Table 21: Traceability Between Nonfunctional System Test Cases and Cultural Requirements

NFR ID	**Test ID**
	CUR2
NFRT-CUR2	X

Table 22: Traceability Between Nonfunctional System Test Cases and Operational and Environmental Requirements

NFR ID	**Test ID**							
	EPER1	**EPER3**	**WER1**	**WER2**	**IASR2**	**RER1**	**RER2**	**RER3**
NFRT-OER1	X	X			X			
NFRT-OER2			X	X				
NFRT-OER3						X	X	X

Table 23: Traceability Between Nonfunctional System Test Cases and Security Requirements

NFR ID	**Test ID**
	INR1
NFRT-SR2	X

9 Trace to Modules

Table 24: Traceability Between Test Cases and Modules

Test IDs	**Module IDs**					
	M3	**M5**	**M6**	**M7**	**M8**	**M9**
FRT-M3-1	X					
FRT-M3-2	X					
FRT-M3-3	X					
FRT-M3-4	X					
FRT-M3-5	X					
FRT-M3-6	X					
FRT-M3-7	X					
FRT-M3-8	X					
FRT-M3-9	X					
FRT-M3-10	X					
FRT-M3-11	X					
FRT-M3-12	X					
FRT-M3-13	X					
FRT-M3-14	X					
FRT-M3-15	X					
FRT-M3-16	X					
FRT-M3-17	X					
FRT-M3-18	X					
FRT-M3-19	X					
FRT-M3-20	X					
FRT-M3-21	X					
FRT-M3-22	X					
FRT-M3-23	X					
FRT-M3-24	X					
FRT-M3-25	X					
FRT-M3-26	X					

FRT-M5-1	X				
FRT-M5-2	X				
FRT-M5-3	X				
FRT-M5-4	X				
FRT-M5-5	X				
FRT-M5-6	X				
FRT-M5-7	X				
FRT-M5-8	X				
FRT-M6-1		X			
FRT-M6-2		X			
FRT-M6-3		X			
FRT-M6-4		X			
FRT-M6-5		X			
FRT-M6-6		X			
FRT-M7-1			X		
FRT-M7-2			X		
FRT-M7-3			X		
FRT-M7-4			X		
FRT-M7-5			X		
FRT-M7-6			X		
FRT-M7-7			X		
FRT-M7-8			X		
FRT-M8-1				X	
FRT-M8-2				X	
FRT-M9-1					X
FRT-M9-2					X
FRT-M9-3					X
FRT-M9-4					X
FRT-M9-5					X
FRT-M9-6					X
FRT-M9-7					X

FRT-M9-8	X
FRT-M9-9	X
FRT-M9-10	X
FRT-M9-11	X

10 Code Coverage Metrics

This section goes over the code coverage of the unit tests created for the Grocery Spending Tracker. For the .dart files, code coverage was calculated using Flutter's built-in testing package. For the .js files, code coverage was calculated using a Node.js package called c8. A summary of the code coverage achieved for each tested file can be found in the table below. All files not listed below were considered either out of scope for unit testing or is being tested as part of the System Tests.

Table 25: Code Coverage Summary

File Name	**Line Coverage**
extract_data.dart	75.9%
capture_item.dart	100%
grocery_trip.dart	100%
goals_repository.dart	86.4%
profile_repository.dart	87.9%
history_repository.dart	73.7%
db.js	100%
authController.js	100%
recommendationController.js	88.82%
usersController.js	92.2%
authentication.js	96.49%
main.js	81.39%
classifyItem.js	100%
fuzzyMatching.js	90.97%
src/classification/util.js	63.82%
brand.js	89.47%
fetchProductDetails.js	94.87%
image.js	100%
name.js	89.47%
price.js	89.47%
productNumber.js	62.96%
src/scraper/util.js	100%

11 Appendix — Survey Results

11.1 Functional Requirement Survey Questions

11.1.1 Survey questions for FRT-FR1-2.

1. How easy was it to log into the application? Responses: {(not easy) 1, 2, 3, 4, 5 (very easy)}

    - On average, a 5. The login process was as expected for users.

11.1.2 Survey questions for FRT-FR8-2.

1. How useful would you say that the application recommendations are to you? Responses: {(not useful) 1, 2, 3, 4, 5 (very useful)}

    - On average, a 3.5. Some recommendations were not 100% accurate for the user but about at the expected value for fit criterion.

2. How likely are you to act on the recommendations provided to you by the application? Responses: {(not likely) 1, 2, 3, 4, 5 (very likely)}

    - On average, a 3. Users would need to see some more accuracy improvements before acting on a recommendation.

11.2 Look and Feel Survey Questions

1. Did you feel the colour scheme was cohesive and consistent throughout the application?

    - General consensus was that colour scheme consistency and cohesiveness was not an issue.
    - General theme had purples and whites, which was consistent overall.

2. On a scale of 1-10, how would you rate the general layouts of each of the pages?

- On average, an 8. Users found the layouts easy to understand and navigate.
- Pages were not found to be too overwhelming in terms of features.
- Layouts/components on the page could be more creative.
- Similar to existing applications such as Instagram.

3. Did the general layouts of the application pages appear consistent throughout navigation?
    - The consensus was that layouts were pretty consistent across the different pages.
    - Layouts were similar enough where users were generally able to figure out the different features on the pages without issue.
4. On a scale of 1-10, how would you rate how features were organized?
    - On average, a 9. All features were where users expected them.
    - Features generally fit the main idea of the pages that they were present on.
5. Were there any places where you expected loading indicators but there weren't any or vice versa?
    - Loading indicators/communication was well received for this version of the application.
    - Many of the users who had seen the previous version of the application with less loading screens said the application was much improved in that aspect.
    - Users felt that the application properly communicated when processes were occurring throughout the application.
    - With general usage, users found that application never felt "frozen" or "stuck".
6. Do you have any feedback on the overall look and feel of the application?
    - In terms of how functional the appearance of the application was, there were no complaints.

- Some users voiced that in terms of visual appeal, the application was lacking (i.e. boring/lacking in creativity compared to other apps).
- Application is very minimal so some minor decorations could help make app more engaging, but generally a minor complaint.

11.3 Usability Survey Questions

1. What parts of the application were easiest for you to understand/learn?
    - Receipt confirmation form was easy for users to fill out for the first time
    - History page was also very easy for users to use and understand
    - Goals page took users slightly longer than the other pages but were able to figure features out without intervention and in reasonable time
2. What parts of the application do you feel additional explanation or clarity could be provided?
    - The receipt capture functionality could use some more clarification as it is not intuitively clear that filling the receipt in the camera frame produces the best results
3. Was the language and wording used in the application understandable and inoffensive? If not, please provide where and how the wording could be improved?
    - Language was generally understood and nothing was offensive to the users.
4. Were there any issues you encountered while using the application?
    - For some users, the submission of their grocery trip took a very long time, close to 10 seconds. This caused them to believe something was wrong with the application due to the length of time it was processing for.
5. Did you have any issues with navigating the application? If you made a mistake, were you able to navigate back to where you started?

- No issues with navigation. Navigation bar was very responsive and navigating back to previous pages was always available.
- Users liked how navigation was always available making it easy to find their way back to where they started.

6. On a scale of 1-10, how would you rate the usability and intuitiveness of the application?

    - On average, the score was an 8. Besides the hiccups with receipt submission and better communication for how the receipt should be captured, users found the app to be easy to navigate.
    - Navigation and layouts were similar to other apps they had used so learning curve was not too difficult.
    - Very little intervention from members of the team was needed to help users.

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Reflection. Please answer the following question:

1. 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.)

During the testing process, we found that the scope of our actual implementation had changed somewhat compared to what we had planned the application to be initially. Given the VnV Plan was written prior to any implementation, we did not have the clearest vision of what the project was going to be. One big example of this was with the Recommendation Engine and User Goals. When writing the VnV Plan, we had imagined them to be more coupled such that we could have recommendations that would specifically help users stay within goals and goals would be available for specific products. The actual implementation we ended up creating ended up being more general, where User Goals are more related to overall spending and recommendations currently exclusively consider a user's purchase history. Similar scope/requirement changes appeared in other parts of the development process resulting in various changes overall being made to the VnV Plan such that it aligned better with our implementation.

In order to better anticipate this kind of change in future projects, we believe that better planning would be needed. The reason these changes had to be made was because we had not fully realized our ideas for the project. Additionally, as development occurred, we realized certain aspects of the application were less feasible than we thought given our development period and knowledge of the technology. If more research and planning had been done prior to the creation of the VnV Plan, it is possible that the tests would have been closer to our current product. In conclusion, changes to the VnV

Plan had to be made due to scope changes throughout development. These changes could be better anticipated if more research and planning had been done at an earlier point.