System Verification and Validation Plan for Grocery Spending Tracker

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

April 4, 2024

Revision History

Date	Version	Notes
03/11/23	1.0	Created initial version of the VnV Plan
04/03/24	1.5	Modified surveys to better algin with current version of the application
05/03/24	2.0	Removed/updated the following sections with outdated features to reflect the current state of the app: 4.1.3: Location-based Recommendations, FRT-FR2-3, FRT-FR6-1, FRT-FR6-1, FRT-FR9-2, NFRT-CUR1, NFRT-CUR3, FRT-FR11-1, FRT-FR8-1, NFRT-LF3, NFRT-UH2, NFRT-UH4, NFRT-OER1, NFRT-SR1, NFRT-SR3, NFRT-SR4, NFRT-SR5, NFRT-SR6
06/03/24	2.5	Added unit tests and unit test traceability
31/03/24	3.0	Updated symbolic constants, added testing details for {NFRT-UH1, FRT-FR1-2}, updated implementation plan for code review and survey plan details
02/04/24	3.5	Alphabetized symbols, updated {design verification plan, FRT-FR2-1, FRT-FR8-2, NFRT-LF1, NFRT-LF2, NFRT-UH6, NFRT-PR1, NFRT-PR2, NFRT-PR4, NFRT-PR5} for feedback

Contents

List of Tables

1 Symbols, Abbreviations, and Acronyms

This section will include any symbols, abbreviations and acronyms used in this document and their definitions.

symbol	**description**
DP	Development Plan
FR	Functional Requirement
HA	Hazard Analysis
MG	Module Guide
MIS	Module Interface Specification
NFR	Non-Functional Requirement
OCR	Optical Character Recognition
SRS	Software Requirements Specification
VnV	Verification and Validation

This document outlines the verification and validation plan for the Grocery Spending Tracker. The document will cover some basic information about the application as well as go over the plans the team has for testing its requirements and features throughout development. It will also cover the different system tests and unit tests that the team will employ to go about validating the software.

2 General Information

This section will give a general overview of the application as well as the objectives the team has for performing validation and verification. This section will also include any relevant document related to the overall testing plan.

2.1 Summary

The Grocery Spending Tracker is a mobile application being developed to help users save money on groceries. The main feature is allowing users to input photos of receipts which the application will then parse. Once parsed, the items and their corresponding prices will then be recorded. This data will then be used to show analytics regarding their purchase history over time. Additionally, the application will also allow users to get budgeting suggestions based on personal spending preferences as well as recommendations for alternative nearby grocery stores where recently purchased items can be obtained for cheaper.

2.2 Objectives

In regards to the application's primary functionality, testing will be done to build confidence in the correctness and accuracy of all main features such as the OCR for receipt input as well as analytic data being returned. This also holds for any budgeting suggestions and alternative grocery store suggestions. Due to the nature of this application focusing on user finances, the accuracy and correctness of all main features are paramount to this application's success. Additionally, qualities that are important for an enjoyable user experience will also be emphasized in this VnV Plan. Testing should verify that the application demonstrates acceptable levels of usability and

performance according to stakeholder expectations.

In general, a majority of the requirements set by the SRS and HA will be considered during validation and verification. Maintainability, adaptability, and compliance tests however will be placed out of scope due to them having less direct impact on the main user experience. Additionally, personalization settings will be out of scope for testing as well due to them having less importance for the current build of the project. The primary focus during this development period will be to build a prototype application with all primary features and qualities rather than a full consumer product. With this in mind, these qualities seemed less relevant to this overall goal. Additionally, several features will leverage external libraries as part of the overall workflow and these components will be excluded from testing as well on a unit level. The team will assume that these libraries have already been tested extensively by the original developers and other users.

2.3 Relevant Documentation

Listed below are the relevant documents referred to in this VnV Plan:

- Software Requirements Specification Nam et al. (2023c)
  The SRS contains the primary functional and non-functional requirements that are being used as the basis for many of the tests in this document.

- Hazard Analysis Nam et al. (2023b)
  The HA contains a number of safety requirements that were identified to help mitigate risk that were also considered for testing in this document.

- Module Guide
  The MG references the different modules that make up the general overall application in which the VnV Plan tests.

- Module Interface Specification
  The MIS contains detailed information on each of the application modules which form the basis for the unit tests.

- Development Plan Nam et al. (2023a)
  The DP describes a roadmap for the project including timelines, communication plans, and technical details.

3 Plan

The following will outline a series of plans and processes that will be used for the verification and validation of the software solution and documentation the Grocery Spending Tracker project. The sections are as follows:

- Verification and Validation Team: the team members their involvement in the verification and validation efforts
- SRS Verification Plan: the approaches to be used for SRS verification
- Design Verification Plan: plans for design verification
- Verification and Validation Plan Verification Plan: plans for VnV verification
- Implementation Verification Plan: dynamic and static tests for verification of the implementation
- Automated Testing and Verification Tools: tools and metrics to be used for testing and verification of the implementation
- Software Validation Plan: plans for verification of the software requirements

The stakeholders referred to in this document are individuals who are members of low-income households in Hamilton and students from McMaster University as described in Section 2 of the SRS. As the primary client, their involvement in the VnV ensures that the system aligns with the needs and goals of the intended users.

3.1 Verification and Validation Team

Developers are responsible for creating, executing, and documenting the tests for verifying and validating the project. This includes SRS verification, design verification, VnV verification, system tests, and unit tests. Task distri-

bution will be flexible and distributed based on available resources. Individuals will also have areas of focus (listed below) where they will take leadership in to minimize fragmented results.

Name	**Role**
Allan Fang	Developer: SRS, VnV, Design Verification
Jason Nam	Developer: SRS, NFR System Tests, FR System Tests
Ryan Yeh	Developer: Unit Tests, Software Validation
Sawyer Tang	Developer: FR System Tests, Unit Tests

Table 1: VnV Team

3.2 SRS Verification Plan

The design documentation will be verified with an informal walk-through by stakeholders and peers of the SFWRENG 4G06 course who will provide ad-hoc review as well as an inspection by the development team. The inspection will address the following issues and be tracked by the checklist:

- Correctness: the SRS accurately represents the software expectations
- Ambiguity: the requirements and contents convey an unambiguous meaning
- Completeness: the SRS covers all necessary functional and non-functional requirements
- Verifiability: the requirements are testable and quantifiable
- Traceability: the requirements have clear source references
- Clarity: the document is written in a clear, concise, and unambiguous manner

3.3 Design Verification Plan

The software design documentation in the form of the MIS and MG will be verified with an informal walk-through by stakeholders and peers of the

SFWRENG 4G06 course who will provide ad-hoc review as well as an inspection by the development team. The review will address the following issues and be guided by the checklist:

- Design adequacy and conformance to software requirements: assess whether the design adequately meets the specified requirements (SRS)
- Internal consistency: different components of the design are coherent
- External consistency: the design aligns with external interfaces, standards, and environments
- Feasibility: whether the design can be practically realized within technical and resource constraints
- Maintenance: whether the design can be easily updated, modified, or repaired

3.4 Verification and Validation Plan Verification Plan

The VnV plan will be verified with an informal walk-through by stakeholders and peers of the SFWRENG 4G06 course who will provide ad-hoc review as well as an inspection by the development team. The reviews will address the following issues and be guided by the checklist:

- Completeness: the VnV plan covers all necessary documentation, functional requirements, and non-functional requirements
- Feasibility: whether the actions that the VnV plan details can be practically realized within technical and resource constraints
- Clarity: the document is written in a clear, concise, and unambiguous manner

3.5 Implementation Verification Plan

The verification of the implementation will be done with static and dynamic tests as well as surveys and interviews with stakeholders. Dynamic tests will be done according to Sections 4 and 5 of this document. For static tests, the development team will conduct a code walk-through to check for

incorrect code along with using the linter and coding standards detailed in Section 3.6. According to the version control process outlined in Section 4 of the Development Plan, all code will reviewed by at least 2 other team members before being merged to production. Survey and interview results will be documented and evaluated in the testing report. Elicited feedback that require changes will be added to the development pipeline.

3.6 Automated Testing and Verification Tools

Refer to Sections 6 and 7 of the Development Plan for the automated testing and verification tools the team will be using.

3.7 Software Validation Plan

A task-based inspection process with stakeholders will be used to validate the software. The inspection will be conducted according to the business events and user flows described in Sections 6 and 8 of the SRS.

4 System Test Description

This section will provide an outline of the various system tests that will be performed by the team during development. The system tests will primarily be based on the functional and nonfunctional requirements recorded in the SRS.

4.1 Tests for Functional Requirements

4.1.1 User Authentication

The following sections pertain to all tests that have to do with the user logging into and gaining access to the system.

Authentication Interface These tests pertain to all the functional requirements related to the authentication interface.

1. **FRT-FR1-1**

   Control: Manual

Initial State: User is not signed into the system.

Input: User attempts to sign into the system with the correct username and credentials.

Output: User is successfully signed in and on the application landing page in no more than three steps.

Test Case Derivation: If the login credentials are correct and match a user in the database, then login should be simple and successful.

How test will be performed: Tester will attempt to log into the application. Validate that it takes less than four steps to reach the landing page.

2. **FRT-FR1-2**

   Control: Manual

   Initial State: User is not signed into the system.

   Input: User attempts to sign into the system with the correct username and credentials.

   Output: User is successfully signed in and on the application landing page.

   Test Case Derivation: Gathering feedback on the overall login experience of the application to test for FR1.

   How test will be performed: Survey is presented to the user for them to respond how easy it is to log into the system. A success is determined by at least a PASSING_RESPONSE_RATE positive response. The survey question is detailed in Section 6.2.1.

Log-in and Authentication Credentials These tests pertain to all the functional requirements related to authentication credentials and the user logging in to the system.

1. **FRT-FR2-1**

   Control: Manual

   Initial State: User with no existing account on file.

   Input: User signs up for account with valid name, email, and password.

Output: User is directed to landing page. Account is successfully created and verification email is sent.

Test Case Derivation: As long as all fields required for sign up are filled with the appropriate data types and formats, then sign up should be completed successfully.

How test will be performed: Tester will open the application and attempt to create a new account with the pertaining credentials and details.

2. **FRT-FR2-2**

   Control: Manual

   Initial State: User with no existing account on file.

   Input: User signs up for account with a valid password however an invalid email is used (i.e. not following the format of a prefix prior to the @ symbol and a domain afterwards).

   Output: Error message should be displayed on the screen detailing an invalid email within MAX_ERROR_DELAY seconds.

   Test Case Derivation: If incorrect/invalid data is used during sign up, the user should be notified and sign up should not be allowed.

   How test will be performed: Tester will open the application and attempt to create a new account with the pertaining credentials and details.

3. ~~**FRT-FR2-3**~~

   ~~Control: Manual~~

   ~~Initial State: User with no existing account on file.~~

   ~~Input: User signs up for account with valid name, email, and password however an invalid phone number is used, for instance, a number less than 10 digits.~~

   ~~Output: Error message should be displayed on the screen detailing an invalid phone number within MAX_ERROR_DELAY seconds.~~

   ~~Test Case Derivation: If incorrect/invalid data is used during sign up, the user should be notified and sign up should not be allowed.~~

~~How test will be performed: Tester will open the application and attempt to create a new account with the pertaining credentials and details.~~

4. **FRT-FR2-4**

   Control: Manual

   Initial State: User with no existing account on file.

   Input: User signs up for account with valid name, email, and phone-number however an invalid password is used, such as an empty string.

   Output: Error message should be displayed on the screen detailing an invalid password within MAX_ERROR_DELAY seconds.

   Test Case Derivation: If incorrect/invalid data is used during sign up, the user should be notified and sign up should not be allowed.

   How test will be performed: Tester will open the application and attempt to create a new account with the pertaining credentials and details.

5. **FRT-FR2-5**

   Control: Manual

   Initial State: User with existing account on file.

   Input: User signs into their account with email and password combination that exists in the user database.

   Output: User is directed to landing page.

   Test Case Derivation: If a user signs in using credentials that match an existing user, sign in should be successful.

   How test will be performed: Tester will open the application and attempt to sign into an account with the pertaining credentials.

6. **FRT-FR2-6**

   Control: Manual

   Initial State: User with existing account on file.

   Input: User signs in using an email that is not stored in the user database.

Output: Error message displayed on the screen saying the email is not associated to an existing account. Prompt to create an account.

Test Case Derivation: The system should prevent login if the email is not associated with a user in the database.

How test will be performed: Tester will open the application and attempt to sign into an account with an email that does not have an account.

7. **FRT-FR2-7**

   Control: Manual

   Initial State: User with existing account on file.

   Input: User signs into an account with an existing email and non-matching password to that user.

   Output: Error message displayed on the screen saying the password is incorrect for the entered email. Prompt to reset password is highlighted within MAX_ERROR_DELAY seconds.

   Test Case Derivation: If both email and password do not match the credentials set for a user in the database, sign in should be rejected.

   How test will be performed: Tester will open the application and attempt to sign into an account with the pertaining credentials.

4.1.2 Receipt Scanning

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

1. **FRT-FR3-1**

   Control: Manual

   Initial State: User has physical receipt and the system has access to the device camera.

   Input: A photo of a physical receipt. User takes photo of receipt with their device.

   Output: User is provided with a list of all their grocery items and prices on the receipt as well as an option to manually enter or edit them.

Test Case Derivation: Assuming optimal conditions, the application should be able to capture the items and corresponding prices of each. The user should then have the opportunity to review the data and make changes before submitting.

How test will be performed: In even lighting, tester will run the system and photograph a receipt. They will then validate that the systems interpretation of the photograph matches that of the physical receipt with above MIN_PRECISION_THRESHOLD precision.

2. **FRT-FR11-1**

   Control: Manual

   Initial State: User has physical receipt and the system has access to the device camera.

   Input: User scans a physical receipt in bad conditions.

   Output: User is provided with a list of all the scanned grocery items and prices on the receipt with the option to retake the photo in better conditions or manually enter or edit them.

   Test Case Derivation: In poor lighting conditions where the receipt is not clearly visible by the camera, the application should still attempt to scan the receipt but should prompt the user to retake the photo in better conditions to ensure accuracy of the data.

   How test will be performed: In a dark environment, tester will run the system and photograph a receipt. They will then validate that the system prompts the user to retake the photo or manually review all scanned items.

3. **FRT-FR11-2**

   Control: Manual

   Initial State: User has no physical receipt.

   Input: User prompts the system that they do not have a receipt to scan.

   Output: User is prompted to manually enter their grocery items and prices.

Test Case Derivation: If the user does not have a usable receipt that they can scan, the system should allow users to manually enter data if necessary.

How test will be performed: Tester will prompt the system interface that they have no receipt. Tester will validate they are able to manually enter their items.

4. **FRT-FR11-3**

   Control: Manual

   Initial State: The system does not have access to the device camera.

   Input: User attempts to scan a receipt.

   Output: User is prompted to give permissions for the application to access camera. If not given, user is asked to manually enter grocery items and prices.

   Test Case Derivation: Without permissions, the application will not work properly. Therefore, the application should prompt the user to give camera access and restrict users to manual receipt entry if not given.

   How test will be performed: Tester will revoke application access to device camera and then attempt to scan a receipt. Verify that the system prompts the user for camera access and asks the user for manual input if not given.

4.1.3 Account Goals and Preferences

These tests pertain to all the functional requirements related to the user's account preferences, goals, or details that affect the system's performance.

~~**Location-based Recommendations** These tests pertain to all the functional requirements related to the user's location and how it affects their grocery recommendations.~~

1. ~~**FRT-FR4-1** Control: Manual~~

   ~~Initial State: The system does not have access to the user's device location.~~

~~Input: User prevents location access for the application~~

~~Output: User is prompted that the system has no location access and given the option to manually enter their general home location and desired recommendation radius.~~

~~Test Case Derivation: If the application is not given access to the device's location, then the application should allow for manual entry by the user to prevent feature disruption.~~

~~How test will be performed: Tester will revoke application access to location and then attempt to receive location-based grocery recommendations. Validate that system provides prompt.~~

2. **~~FRT-FR4-2~~**

   ~~Control: Manual~~

   ~~Initial State: The system has access to the user's device location but user wants to manually set a different one.~~

   ~~Input: User changes preferences to use manually entered home location and recommendation radius rather than device location.~~

   ~~Output: System provides accurate location-based recommendations regarding the manually entered location.~~

   ~~Test Case Derivation: If the user wishes to use a manually set location rather than their device's location, then the application should still function the same in both scenarios.~~

   ~~How test will be performed: Tester will go to preferences and opt to use manually provided home location and recommendation radius. Validate that the system is providing correct grocery recommendations for the manually inputted location and radius.~~

3. **~~FRT-FR4-3~~**

   ~~Control: Manual~~

   ~~Initial State: The system has access to the user's device location.~~

   ~~Input: System has location access on device.~~

   ~~Output: Application provides recommendations to the user for stored that are within the default radius of their location.~~

~~Test Case Derivation: Should the application have access to the device location, then all location preferences should be relative to the device's active location.~~

~~How test will be performed: Tester will enable location services for the application and validate that the system is providing correct grocery recommendations for the current device location.~~

User Objectives These tests pertain to all the functional requirements related to the user's goals and spending objectives.

1. **FRT-FR5-1**

   Control: Manual

   Initial State: The system has no spending goals and desired outcomes stored in the system.

   Input: User navigates to the preferences of their application.

   Output: The system provides a space for the user to input their spending goals.

   Test Case Derivation: The system must provide an interface for the user to input their spending goals.

   How test will be performed: Tester will navigate to the application preferences and validate that an interface exists for inputting spending goals.

2. **FRT-FR5-2**

   Control: Manual

   Initial State: The system has existing spending goals and desired outcomes stored in the system. The system displayed the current spending goals.

   Input: User navigates to the preferences of their application and changes their spending goals.

   Output: The system is updated to reflect the new spending goals.

   Test Case Derivation: If spending goals are ever updated, then the application should take the new spending goals into account for correctness.

How test will be performed: Tester will navigate to the application preferences and validate that the existing spending goals are displayed. They will then attempt to change the values of the spending goals and then save the changes.

3. ~~**FRT-FR6-1**~~

   ~~Control: Manual~~

   ~~Initial State: The system has no budgeting constraints stored.~~

   ~~Input: User navigates to the preferences of their application.~~

   ~~Output: The system provides a space for the user to input their budgeting constraints.~~

   ~~Test Case Derivation: The system must provide an interface for the user to input their budgeting constraints.~~

   ~~How test will be performed: Tester will navigate to the application preferences and validate that an interface exists for inputting user budgeting constraints.~~

4. ~~**FRT-FR6-2**~~

   ~~Control: Manual~~

   ~~Initial State: The system has existing budgeting constraints stored.~~

   ~~Input: User navigates to the preferences of their application and changes their budgeting constraints.~~

   ~~Output: The system displays the new budgeting constraints.~~

   ~~Test Case Derivation: If budgeting constraints are ever updated, then the application should take the new constraints into account for correctness.~~

   ~~How test will be performed: Tester will navigate to the application preferences and validate that the existing goals are displayed. They will then attempt to change the budgeting constraints values and save the changes.~~

4.1.4 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 the recommendations provided to the user.

Receiving Grocery Data These tests pertain to all the functional requirements related to how the system shall treat and handle data it receives.

1. **FRT-FR7-1**

   Control: Automatic

   Initial State: Test database is empty.

   Input: API calls from client (front-end) to back-end containing receipt data. Receipt data is comprised of an array of *Item(string name, double price, string date, string purchaseLocation, int userID).*

   Output: Database contains correctly organized items. The user's database table should include all items they purchased.

   Test Case Derivation: System back-end must accurately map and sort receipt entries provided, keeping a record of the user who purchased each item.

   How test will be performed: Begin with an empty testing database. Test cases comprised of items and prices are sent through an API call to the system back-end by testing client. Then an additional API call will be called from the testing client against the system back-end to validate that the items and prices are correctly stored in the database. *Postman* will be used to facilitate API calls.

2. **FRT-FR7-2**

   Control: Automatic

   Initial State: Test database is populated.

   Input: API calls from client (front-end) to back-end containing receipt data. Receipt data only contains items that are not already in the database. Receipt data is comprised of an array of *Item(string name, double price, string date, string purchaseLocation, int userID).*

   Output: System back-end must accurately map and sort receipt entries provided, keeping a record of the user who purchased each item while preserving existing data.

   Test Case Derivation: System back-end must accurately map and sort receipt entries provided, keeping a record of each item all users purchase.

How test will be performed: Begin with a populated testing database. Test cases comprised of items and prices are sent through an API call to the system back-end by testing client. Then an additional API call will be called from the testing client against the system back-end to validate that the items and prices are correctly stored in the database. *Postman* will be used to facilitate API calls.

3. **FRT-FR7-3**

   Control: Automatic

   Initial State: Test database is populated.

   Input: API calls from client (front-end) to back-end containing receipt data. Data containing items that are the same as existing items in the database but with different name identifiers. An example of this would be a difference of "Rtz Crackers" or "Ritz Cracker". Receipt data is comprised of an array of *Item(string name, int cents, double price, string purchaseLocation, int userID).*

   Output: Database contains correctly organized items and understanding the relationship between like items. If like products are added to the database, the database correctly maps them to the existing items of the same type, even if the name string is different. The database also keeps track of which users purchase items.

   Test Case Derivation: System back-end must accurately map and sort receipt entries provided, keeping a record of the user who purchased each item while preserving existing data.

   How test will be performed: Begin with a populated testing database. Test cases comprised of items and prices are sent through an API call to the system back-end by testing client. Then an additional API call will be called from the testing client against the system back-end to validate that the items and prices are correctly stored in the database. *Postman* will be used to facilitate API calls.

Grocery Recommendations These tests pertain to all the functional requirements related to the system's grocery recommendations based on grocery data received from user and their preferences.

1. **FRT-FR8-1**

Control: Manual

Initial State: User has set budget information.

Input: User submits a receipt that puts them over budget.

Output: The system should indicate that the new purchases has caused the user to go over budget.

Test Case Derivation: System must be aware of the users purchasing activity and budget. If the user ever goes over budget when entering a receipt, the system should identify why they went over budget.

How test will be performed: Tester will input receipt meeting the above conditions. The tester will then validate that the new item is correctly identified as putting the user over budget.

2. **FRT-FR8-2**

   Control: Manual

   Initial State: User an account on the app with some purchase history.

   Input: Recommendations based on user's purchase history.

   Output: The system should produce recommendations relevant to the user.

   How test will be performed: Survey is presented to the user for them to respond how useful the system recommendations are. A success is determined by at least a PASSING_RESPONSE_RATE positive response. The survey question is detailed in Section 6.2.2.

3. **FRT-FR9-1**

   Control: Manual

   Initial State: The database has an entry for a grocery item that is purchased from a location X.

   Input: User inputs a receipt containing the grocery item in the database but at a higher price from location Y.

   Output: System provides recommendation to purchase item from location X.

   Test Case Derivation: System must be aware of the users purchasing activity and recommend the user with cheaper alternative locations for the same item.

How test will be performed: Tester will submit items purchased for a cheap price on one user account (A) and then submit the same item purchased at a greater price from a different store with another user account (B). Tester verifies that user B receives a recommendation from the system to go to the same place as user A.

4. **FRT-FR9-2**

   Control: Manual

   Initial State: The database has an entry for a grocery item that is purchased from a location X.

   Input: User submits a receipt containing the grocery item in the database but at a cheaper price from location Y.

   Output: System provides recommendation to purchase item from location Y.

   Test Case Derivation: System must be aware of the users purchasing activity and recommend the user with cheaper alternative locations for the same item.

   How test will be performed: Tester will submit an item purchased for some amount on one user account (A) and then submit the same item for a cheaper amount from a different store with another user account (B). Tester verifies that user A receives a recommendation from the system to go to the same place as user B.

Grocery Items Interface These tests pertain to all the functional requirements related to the system's grocery item interface.

1. **FRT-FR10-1**

   Control: Manual

   Initial State: User has no spending data or submitted items.

   Input: User submits a receipt.

   Output: System provides an interface showing all purchased items along with price, location, and date-time.

   Test Case Derivation: System must provide the user with all of the data that the user has provided the system such that the user can review their own purchasing data.

How test will be performed: Tester will submit a receipt and then verify to see that the items have been added in the purchases interface.

2. **FRT-FR10-2**

   Control: Manual

   Initial State: User has existing spending data and submitted items.

   Input: User submits a receipt.

   Output: System provides an interface showing all purchased items along with price, location, and date-time.

   Test Case Derivation: System must provide the user with all of the data that the user has provided the system such that the user can review their own purchasing data.

   How test will be performed: Tester will submit a receipt and then verify to see that the items have been added in the purchases interface along with all of the preexisting items.

4.2 Tests for Nonfunctional Requirements

4.2.1 Look and Feel Requirements

1. NFRT-LF1

   Type: Manual, Dynamic

   Initial State: User has not previously used the application.

   Input: User is given the application for the first time.

   Output: The user should communicate opinions on look and feel of the app.

   How test will be performed: Testers will be presented with the application and guided through navigation of various pages available and asked to take note of the aesthetics, appearance and feel of the application. A survey will then be provided to evaluate these qualities which can be found in Section 6.3 of this document.

2. NFRT-LF2

Type: Manual

Initial State: User has not previously used the application.

Input: User is given a list of features for the application and a list of unlabelled icons corresponding to each of the features.

Output: The user be able to match all icons with features.

How test will be performed: Testers will be given a list of features for the application and a list of unlabelled icons corresponding to each of the features. They will be asked to match which feature they believe pairs with which icon.

3. ~~NFRT-LF3~~

   ~~Type: Manual, Static~~

   ~~How test will be performed: Walkthroughs will be performed on the code of each of the frontend pages using the *Flutter Inspector* to check size of components and layouts. This will be to ensure no more than 85% of the page is occupied and a numerical balance is achieved in the layout.~~

4. NFRT-LF4

   Type: Manual, Dynamic

   Initial State: The application is opened on a device and logged into a user's account with internet access available.

   Input: The user disconnects from the internet.

   Output: The application should notify the user that they do not have internet connection.

   How test will be performed: Testers will be asked to disconnect from the internet while the application is open.

4.2.2 Usability and Humanity Requirements

1. NFRT-UH1

   Type: Manual, Dynamic

Initial State: The tester has no prior experience with the application.

Input: A fresh initial state of the application.

Output: PASSING_RESPONSE_RATE of testers should be able to perform basic functionality of inputting a receipt and viewing purchase analytics after using the app for FIRST_IMPRESSION_TIME.

How test will be performed: Testers will be presented with the application and given a brief overview of its purpose. They will then be given FIRST_IMPRESSION_TIME to use the application under supervision by the team but without instruction. After that time period, the testers will be asked to input a receipt and navigate to the purchase analytics.

2. NFRT-UH2

   Type: Manual, Dynamic

   Initial State: The application is opened on a device and logged into a user's account with internet access available.

   Input: The user attempts to input a receipt but aborts the process.

   Output: No changes should be made to the user's account data.

   How test will be performed: Testers will be given a receipt to input and asked to input it into the application. Testers will then be asked to abort the input and observe for unintended changes in the app.

3. NFRT-UH3

   Type: Manual, Dynamic

   Initial State: The application is opened on a device and logged into a user's account with internet access available.

   Input: The user tries changing the application's font size.

   Output: The font size should be changed throughout the application without negatively impacting readability or general functionality.

   How test will be performed: Testers will be asked to change the font size to their desired size and navigate through the application to check usability.

4. ~~NFRT-UH4~~

   ~~Type: Manual, Dynamic~~

   ~~Initial State: The application is opened on a device and logged in to a user's account with internet access available.~~

   ~~Input: The user changes the accessibility colour scheme.~~

   ~~Output: The colour scheme should be changed throughout the application without negatively impacting readability or general functionality.~~

   ~~How test will be performed: Testers will be asked to change the accessibility colour scheme to their desired setting and navigate through the application to check usability.~~

5. NFRT-UH5

   Type: Manual, Dynamic

   Initial State: The application is opened on a device and logged in to a user's account with accessibility settings having been changed.

   Input: The user closes the application.

   Output: When re-opened, the application should still be logged into the user's account and accessibility settings should still be in place.

   How test will be performed: Following *NFRT-UH4* and *NFRT-UH5*, testers will be asked to close the application on their device while logged in and re-open it afterwards.

6. NFRT-UH6

   Type: Manual

   Initial State: Completed NFRT-UH1

   Input: The user is to freely explore the application.

   Output: The user will provide feedback on the usability of the app.

   How test will be performed: A survey will be presented to testers to evaluate the overall usability of the application after first going through *NFRT-UH1*. Afterwards, the team will show them any remaining features they had not yet experienced and given the opportunity to freely

experiment with the applications. A survey will then be provided to them. The questions asked will be as written in Section 6.4 of this document.

4.2.3 Performance Requirements

1. NFRT-PR1

   Type: Manual, Dynamic

   Initial State: App is running.

   Input: All pages of the app will be navigated to.

   Output: Flutter performance profiling will provide performance timing data.

   How test will be performed: All pages of the application will be navigated through while using the *Flutter Performance Profiling* tools. Times for rendering and frontend loading will then be recorded to ensure that frontend performance is within expectation.

2. NFRT-PR2

   Type: Automatic, Dynamic

   Initial State: Backend is running.

   Input: All backend endpoints will be hit.

   Output: Response times will be recorded.

   How test will be performed: Requests will be sent to all backend endpoints via *Postman* and after all requests are sent, the times for each request will be recorded. The team will then confirm that the backend performs within expectation.

3. NFRT-PR3

   Type: Automatic, Dynamic

   Initial State: The backend server is running and operational.

   Input: For all endpoints, incorrect data types and incorrect inputs (such as empty inputs) will be passed.

Output: The server should have appropriate error handling where no invalid data is accepted and the server remains operational.

How test will be performed: A test suite with invalid data for the major endpoints will be created with *Postman*. The test suite will then be automatically executed by *Postman* and the results will be analyzed after.

4. NFRT-PR4

   Type: Manual, Dynamic

   Initial State: FRT-FR3-1 has been performed.

   Input: Receipt is scanned.

   Output: The computed results should have accuracy greater than MIN_PRECISION_THRESH of items.

   How test will be performed: Based on tests performed for *FRT-FR3-1*, the tester should not have to modify and correct data scanned for greater than MIN_PRECISION_THRESHOLD of total scanned items.

5. NFRT-PR5

   Type: Manual, Dynamic

   Initial State: *FRT-FR10-1* and *FRT-FR10-2* have been performed.

   Input: *FRT-FR10-1* and *FRT-FR10-2* test results.

   Output: The relative error of data such as average spending will be computed and should be less than MAX_RELATIVE_ERROR.

   How test will be performed: For tests performed for *FRT-FR10-1* and *FRT-FR10-2*, the relative error of data such as average spending will be computed and should be less than MAX_RELATIVE_ERROR.

6. NFRT-PR6

   Type: Manual, Dynamic

   Initial State: A device does not have the application installed.

   Input: The application is installed onto the device.

Output: The device storage should indicate that the application uses less than MAX_APP_SIZE of storage.

How test will be performed: The tester will install the application onto a device without the application currently installed.

4.2.4 Cultural Requirements

1. ~~NFRT-CUR1~~

   ~~Type: Manual, Dynamic~~

   ~~Initial State: The app is running and set to the default system of measurement (e.g., metric).~~

   ~~Input: The user selects their preferred units of measurement from the app settings.~~

   ~~Output: The app interface and displayed measurements change to the selected units of measurement as per the user's choice.~~

   ~~How test will be performed: The tester will change the units of measurement from the app settings and verify that the interface and displayed measurements switch to the selected units.~~

2. NFRT-CUR2

   Type: Manual, Dynamic

   Initial State: The app contains common purchase items in the product database.

   Input: The user attempts to add a purchase item to purchase history that is not present in the database.

   Output: The app successfully adds the new purchase item to the user's purchase history.

   How test will be performed: The tester will attempt to add a product not present in the app product database to the purchase history and verify that the item is added without any issues.

3. ~~NFRT-CUR3~~

   ~~Type: Manual, Dynamic~~

~~Initial State: The app is set to the default currency (e.g., CAD).~~

~~Input: The user selects the American currency in the app settings.~~

~~Output: The app logs all future purchases all in USD based on the user's selection and will not compare the prices of similar products nearby of a different currency.~~

~~How test will be performed: The tester will access the app's settings, change the currency to USD. The tester will then verify that all prices of future purchases are logged in USD and similar products priced with CAD are not compared.~~

4.2.5 Operational and Environmental Requirements

1. NFRT-OER1

   Type: Manual, Dynamic

   Initial State: Application is not yet installed on the device.

   Input: The user will install and run the application on various devices.

   Output: The application will be successfully installed. The application is operational.

   How test will be performed: The tester will install the application on various mobile devices running Android on version 13 (IASR2). The tester will also test on various tablet devices running Android. The tester will confirm that all functionalities work as expected on each device by conducting variety of actions and system events (including logging in to an user account, input receipt data, analyzing spending analytics, etc.).

2. NFRT-OER2

   Type: Manual, Dynamic

   Initial State: The application is installed on a device that has an operable camera peripheral and has access to the internet.

   Input: The user will attempt to use application features that require a camera and/or internet access.

Output: The application will successfully operate the features that require the use of camera and/or internet.

How test will be performed: The tester will test the application on devices with and without camera peripherals to validate (WER1). The tester will disconnect devices from the internet to test offline functionality. The tester will then reconnect the device to the internet to test online features (WER2). The tester will confirm that all functionalities work on devices that have access to camera peripherals and internet access.

3. NFRT-OER3

Type: Automatic, Dynamic, Static

Initial State: The application is ready for new releases or any updates.

Input: Test suites will be executed against new releases and updates.

Output: The application will identify defects. The team will resolve identified defects. In doing so, the application testing prior to new releases and updates will achieve MIN_CODE_COVERAGE code coverage.

How test will be performed: The tester will use automated testing tools (that can include *Postman*, *Flutter*, etc) to achieve MIN_CODE_COVERAGE code coverage (RER1). The tester will then perform issue tracking and resolution processes (RER2). The tester will organize a Final Demonstration (RER3) between March 18 and March 29 of 2024, involving stakeholders to validate the final product against requirements.

4.2.6 Security Requirements

1. ~~NFRT-SR1~~

~~Type: Manual, Dynamic~~

~~Initial State: User is not logged in or terms are not accepted.~~

~~Input: User tries to access the application's database.~~

~~Output: Access will be denied until the user accepts terms and services.~~

~~How test will be performed: Attempts will take place to access the database without accepting terms. The tester will confirm denial of access. Then, attempt access after accepting terms and services. The tester will confirm successful database access.~~

2. NFRT-SR2

   Type: Manual, Dynamic

   Initial State: Data is stored in the application database.

   Input: User will attempt to modify the data directly from the application database.

   Output: Stored data is encrypted. Therefore, modifications to the data are denied.

   How test will be performed: The tester will attempt to use database tools to try to modify data within the application database. Then, the tester will confirm that the application backend database is secure and has not been modified.

3. ~~NFRT-SR3~~

   ~~Type: Manual, Dynamic~~

   ~~Initial State: Application is in operation.~~

   ~~Input: User will attempt simulated attacks like SQL injection, cross-site scripting, etc.~~

   ~~Output: Application will remain secure without any data breaches.~~

   ~~How test will be performed: The tester will conduct a series of different types of simulated cyber attacks. The tester will then observe application response and monitor application database.~~

4. ~~NFRT-SR4~~

   ~~Type: Manual, Dynamic~~

   ~~Initial State: Raw user data is stored in the application database.~~

   ~~Input: The raw user data is processed and then stored in the application database.~~

~~Output: The data is anonymized by data perturbation (inducing small amount of noise) and encryption. The data is also sanitized by data deletion or masking.~~

~~How test will be performed: The tester will inspect the stored user data for any personal identifiers. The tester will search for any presence of accessible sensitive user data.~~

5. ~~NFRT-SR5~~

   ~~Type: Manual, Dynamic~~

   ~~Initial State: Application has yet to collect user data.~~

   ~~Input: User will start using the application.~~

   ~~Output: A consent form will pop up before any user data is collected. No user data will be collected without explicit consent given by user.~~

   ~~How test will be performed: The tester will check for prompts that ask users to consent to data collection. The tester will then test the application with and without user consent to ensure compliance. The tester will confirm no user data was collected and stored in the database without user consent.~~

6. ~~NFRT-SR6~~

   ~~Type: Manual, Dynamic~~

   ~~Initial State: Application is in operation.~~

   ~~Input: User will attempt various actions to induce system events (including logging in to an user account, input receipt data, analyzing spending analytics, etc.).~~

   ~~Output: Activities and events will be accurately logged. The log can be reviewed by authorized administrators.~~

   ~~How test will be performed: The tester will perform a variety of user actions and try to trigger system events (including logging in to an user account, input receipt data, analyzing spending analytics, etc.). The tester will verify if these actions and events are accurately logged to system log. The tester will confirm that only authorized administrators can access these logs.~~

4.3 Traceability Between Test Cases and Requirements

Table 2: Traceability Between Functional System Test Cases and Functional Requirements

Functional Requirement ID	**Test ID**										
	FR1	**FR2**	**FR3**	**FR4**	**FR5**	**FR6**	**FR7**	**FR8**	**FR9**	**FR10**	**FR11**
FRT-FR1-1	X										
FRT-FR1-2	X										
FRT-FR2-1		X									
FRT-FR2-2		X									
FRT-FR2-3		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-FR4-1				X							
FRT-FR4-2				X							
FRT-FR4-3				X							

FRT-FR5-1			X						
FRT-FR5-2			X						
FRT-FR6-1				X					
FRT-FR6-2				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 3: Traceability Between Nonfunctional System Test Cases and Look and Feel Requirements

NFR ID	**Test ID**					
	AR1	**AR2**	**~~AR3~~**	**SR1**	**SR2**	**SR3**
NFRT-LF1	X		X	X	X	X
NFRT-LF2		X				
NFRT-LF3			X			
NFRT-LF4						X

Table 4: Traceability Between Nonfunctional System Test Cases and Usability and Humanity Requirements

NFR ID	**Test ID**										
	EUR1	**EUR2**	**EUR3**	**PIR1**	**LR1**	**~~LR2~~**	**LR3**	**UPR1**	**UPR2**	**ACR1**	**~~ACR2~~**
NFRT-UH1							X				
NFRT-UH2		X									
NFRT-UH3										X	
NFRT-UH4											X
NFRT-UH5	X										
NFRT-UH6			X	X	X	X		X	X	X	X

Table 5: 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 6: Traceability Between Nonfunctional System Test Cases and Cultural Requirements

NFR ID	**Test ID**		
	CUR1	**CUR2**	**CUR3**
NFRT-CUR1	X		
NFRT-CUR2		X	
NFRT-CUR3			X

Table 7: Traceability Between Nonfunctional System Test Cases and Operational and Environmental Requirements

NFR ID	**Test ID**									
	EPER1	~~**EPER2**~~	**EPER3**	**WER1**	**WER2**	~~**IASR1**~~	**IASR2**	**RER1**	**RER2**	**RER3**
NFRT-OER1	X	X	X			X	X			
NFRT-OER2				X	X					
NFRT-OER3								X	X	X

Table 8: Traceability Between Nonfunctional System Test Cases and Security Requirements

NFR ID	**Test ID**					
	ACR1	**INR1**	**INR2**	~~**PRR1**~~	~~**PRR2**~~	~~**AUR1**~~
NFRT-SR1	X					
NFRT-SR2		X				
NFRT-SR3			X			
NFRT-SR4				X		
NFRT-SR5					X	
NFRT-SR6						X

5 Unit Test Description

This section details the unit tests that will be performed by the team during development. The unit tests will be focused on the modules of the application as written in the MIS.

5.1 Unit Testing Scope

All modules described in the MIS will be within the scope of unit testing except for the Receipt Vision Module, Database Driver Module and Locations Module. The Receipt Vision Module is excluded due to it largely being implemented via a third-party package called Google ML Kit. It is assumed this is a well-tested package and is thus excluded. Other functionality such as camera operation is already being tested via System Tests. Similarly, the Database Driver Module is being implemented using PostgreSQL which is also assumed to be properly tested. Additionally, the database will be tested through System Tests rather than individual unit tests. Finally, the Locations Module is being excluded because the team has decided it is out of scope for our current implementation. As a result, implementation and testing related to the module is being pushed to a future version.

5.2 Tests for Functional Requirements

This section goes over all the Unit Tests related to the functional requirements of the application. The unit tests are all grouped by the module they are related to, as written in the Module Guide and Module Interface Specification.

5.2.1 Receipt Extraction Module

All the unit tests for the Receipt Extraction Module can be found in the following file: Receipt Extraction Unit Tests. The tests focus on the individual aspects/tasks of the Receipt Extraction process and try to verify the "happy paths" and edge cases such as empty inputs for each. The tests are all being performed automatically and dynamically through Flutter's testing package.

5.2.2 User Analytics Module

The front-end unit tests for the User Analytics Module can be found in the following files: Front-end User Analytics Module Spending Goals Tests, Front-end User Analytics Module Spending History Tests. The tests are created to have complete path coverage constrained by the user interface control flows. The tests are all being performed automatically and dynamically through Flutter's testing package.
The back-end unit tests for the User Analytics Module can be found in the following file: Back-end User Analytics Module Controller Tests. The tests conform to an MIN_CODE_COVERAGE coverage of the endpoint and logic functions and are run using Mocha, Chai, and Sinon for the framework and c8 for the coverage.

5.2.3 Users Module

The back-end unit tests for the Users Module can be found in the following file: Back-end User Module Controller Tests. The tests conform to an MIN_CODE_COVERAGE coverage of the endpoint and logic functions and are run using Mocha, Chai, and Sinon for the framework and c8 for the coverage.

5.2.4 Authentication Module

The front-end unit tests for the Authentication Module can be found in the following file: Front-end Authentication Module Tests. The tests are created to have complete path coverage constrained by the user interface control flows. The tests are all being performed automatically and dynamically through Flutter's testing package.
The back-end unit tests for the Authentication Module can be found in the following files: Back-end Authentication Module Controller Tests and Back-end Authentication Module Utility Tests. The tests conform to an MIN_CODE_COVERAGE coverage of the endpoint and logic functions and are run using Mocha, Chai, and Sinon for the framework and c8 for the coverage.

5.2.5 Recommendation Engine

The back-end unit tests for the Recommendation Module can be found in the following files: Back-end Recommendation Module Tests. The tests conform to an MIN_CODE_COVERAGE coverage of the endpoint and logic functions and are tested using Mocha, Chai, and Sinon for the framework and c8 for the coverage of the tests.

5.2.6 Classification Engine

The back-end unit tests for the Classification Engine Module can be found in the following files: Back-end Classification Module Main Test, Back-end Classification Module Item Matching Tests, Back-end Classification Module Web Scraping Tests. The tests conform to an MIN_CODE_COVERAGE coverage of the logic functions and are tested using Mocha, Chai, Sinon for the framework and c8 for the coverage of the tests.

5.3 Tests for Nonfunctional Requirements

The team does not plan to use unit testing for nonfunctional requirements. All testing for nonfunctional requirements will be done via System Tests.

5.4 Traceability Between Test Cases and Modules

Table 9: 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

References

Jason Nam, Allan Fang, Sawyer Tang, and Ryan Yeh. Development plan. https://github.com/r-yeh/grocery-spending-tracker/blob/master/docs/DevelopmentPlan/DevelopmentPlan.pdf, 2023a.

Jason Nam, Allan Fang, Sawyer Tang, and Ryan Yeh. Hazard analysis. https://github.com/r-yeh/grocery-spending-tracker/blob/master/docs/HazardAnalysis/HazardAnalysis.pdf, 2023b.

Jason Nam, Allan Fang, Sawyer Tang, and Ryan Yeh. Software requirements specification. https://github.com/r-yeh/grocery-spending-tracker/blob/master/docs/SRS/SRS.pdf, 2023c.

6 Appendix

6.1 Symbolic Parameters

Constant Name	Constant Value
PASSING_RESPONSE_RATE	70%
MAX_ERROR_DELAY	3
MIN_PRECISION_THRESHOLD	90%
MAX_RELATIVE_ERROR	5%
MAX_APP_SIZE	1GB
MIN_CODE_COVERAGE	80%
FIRST_IMPRESSION_TIME	15 minutes

6.2 Functional Requirement Survey Questions

6.2.1 Survey questions for FRT-FR1-2.

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

6.2.2 Survey questions for FRT-FR8-2.

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

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

6.3 Look and Feel Survey Questions

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

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

3. Did the general layouts of the application pages appear consistent throughout navigation?

4. On a scale of 1-10, how would you rate how features were organized?

5. Were there any places where you expected loading indicators but there weren't any or vice versa?

6. Do you have any feedback on the overall look and feel of the application?

6.4 Usability Survey Questions

1. What parts of the application were easiest for you to understand/learn?

2. What parts of the application do you feel additional explanation or clarity could be provided?

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?

4. Were there any issues you encountered while using the application?

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?

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

Appendix — Reflection

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

1. **What knowledge and skills will the team collectively need to acquire to successfully complete the verification and validation of your project? Examples of possible knowledge and skills include dynamic testing knowledge, static testing knowledge, specific tool usage etc. You should look to identify at least one item for each team member.**

   For the verification and validation of this project, several skills will be required to ensure that this project is well-tested and meets all requirements. This will involve some general testing knowledge as well as some knowledge specific to the technologies we are using. By learning the following, the team will be able to produce a better project for this capstone in addition to more confidence in the testing process in our careers.

   (a) Knowledge of the different forms and approaches of dynamic testing

   (b) Knowledge of the different forms and approaches of static testing

   (c) Experience and knowledge working with *Flutter's* internal unit testing framework and *Flutter Performance Profiling* for frontend performance

   (d) Experience and knowledge using *Postman* for backend tests

   (e) Communication skills to write surveys and elicit feedback from stakeholders/testers

2. **For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?**

   For skills *(a)* and *(b)*, one approach that could be taken is to review notes from *SFWRENG 3S03* where topics such as dynamic and static

testing were covered. The different approaches for dynamic and static testing were also part of the lecture material which would be useful to go over again. A second approach is to conduct research online for tutorials, articles and documents covering the static and dynamic testing approaches. Hands-on experience could also be employed with different technologies that support static and dynamic testing.

Skills *(c)* and *(d)* involve more specific technologies, so one approach to learning these could be to read documentation. For both technologies, documentation is present to learn how the developers had intended them to be used. For a second approach, *YouTube* tutorials can be referenced to go over the basics and learn these technologies. Since many tutorials are made by users, it can provide more insight on how these applications can be used in a more practical setting. Lastly, third-party articles can also be used for additional learning.

Finally, skill *(e)* is more of a social skill rather than a technical skill compared to the other skills listed. One approach to improve communication skills for testing is to have our team actively engage in group collaboration and seek feedback from fellow team members when preparing for feedback elicitation. For instance, reviewing a proposed survey with the team will identify the good and bad aspects present that team members can internalize. For a second approach, *SFWRENG 3S03* has lectures on surveys (such as the kinds of questions to ask and not ask) and similar testing approaches that can also be reviewed to better learn and improve this skill. Additionally, coursework in *SFWRENG 4HC3* could also be used as survey formats and guidelines are covered in the material. Last, familiarization with software that supports surveys and elicitation could also be used as another approach.

Below are the chosen skills from each team member, their rationale, and approaches they will use to learn or improve said skills.

Ryan Yeh
Chosen Skills: (c), (d), (e)
For verification and validation, I will be taking a leadership role in designing unit tests and software validation. Therefore, I want to ensure familiarity with *Flutter's* testing technology and *Postman* given that unit tests will primarily utilize these technologies. By ensuring proper learning of this software, I should be able to better guide the rest of the

team during unit test development resulting in a stronger test suite. In addition, I also chose communication skills as a skill to focus on since this application is primarily user-focused. Therefore, ensuring proper user feedback and testing with stakeholders will be very important to this app's success.

To learn and improve my skills with *Flutter's* testing software and *Postman*, I will refer to the official documentation and *YouTube* tutorials. I feel this will give me a balance between original developer knowledge and practical user knowledge. In terms of communication skills, I will be employing group feedback and collaboration to ensure high-quality questions are being asked to maximize the feedback and information we receive from stakeholders and testers.

Sawyer Tang
Chosen Skills: (a), (b), (d)
I am going to be responsible for the FR system tests and the unit tests for the system. Therefore, I believe it is important for me to have a deep understanding of dynamic testing, static testing, and the use of *Postman* for API testing. By having a deep understanding of these concepts and skills, I can contribute to the project's correctness and testing depth. To develop my Dynamic testing and Static testing skills, I plan to reference my *SFWRENG 3S03* notes and resources, as well as put into practice skills from online articles and forums. With regards to improving my abilities with *Postman*, I plan to reference *YouTube*, blogs, and *Postman's* own documentation.

Jason Nam
Chosen Skills: (a), (b), (e)
I will be responsible for SRS Verification, NFR system tests, FR system tests. These roles will require relevant skills and knowledge for dynamic testing, static testing, and writing surveys and eliciting feedback from stakeholders and/or testers. Through thorough understanding of the the listed skills and knowledge, I believe that I can be of use and contribute to the verification and validation of the application.

I plan to develop my dynamic testing and static testing skills by referring back to *SFWRENG 3S03* practices. I also plan to understand the basics of each testing methods first by partaking in online courses, articles, and tutorials. I plan to try hands-on testing and practice cases.

I will also take advantage of code review practices and analysis tools like *flutter_lints* and *Pylint* to enhance static testing skills. For survey and eliciting feedback related skills, I will study different survey designs. I will reference back to *SFWRENG 4HC3* for understanding survey procedures and guidelines as well as eliciting feedback. I plan to familiarize myself with *Google Forms* and other tools to optimize surveys and eliciting feedback.

Allan Fang

Chosen Skills: (b), (e)

I will be taking the lead on SRS Verification, VnV Verification, and Design Verification which will primarily be the verification of documentation. Therefore, I will focus on acquiring the relevant skills and knowledge for documentation verification according to the VnV plan (guided walkthroughs with stakeholders and inspections with checklists) which will be static testing and communication for eliciting feedback. I will be using both proposed approaches for acquiring knowledge on static testing and both proposed approaches for acquiring communication skills to elicit feedback.

I have chosen the first two approaches for understanding static testing because I can gain both theoretical knowledge (course notes) and practical insights (online research) into the various aspects of static testing. This comprehensive approach will help me be more effective in my verification.

I have also chosen the first two approaches for communication skills for eliciting feedback. The first approach of group collaboration and feedback will enable me to practice and refine this skill while the second approach will provide a solid theoretical foundation to start with. Using both approaches will enhance my ability to elicit valuable feedback from stakeholders during the verification process, ensuring well-validated documentation.