Verification and Validation Report: Utrition

Team 16, Durum Wheat Semolina
Alexander Moica
Yasmine Jolly
Jeffrey Wang
Jack Theriault
Catherine Chen
Justina Srebrnjak

March 8, 2023

1 Revision History

Date	Version	Notes
March 8, 2023	1.0	Initial Version

2 Symbols, Abbreviations and Acronyms

symbol	description
API	Application programming interface
App	Application
BMI	Body Mass Index
CP	Cultural and Political Requirement
CT	Cultural Test
FR	Functional Requirement
LF	Look and Feel
LR	Legal Requirement
LT	Legal Test
MS	Maintainability and Support
OE	Operational and Environmental
POC	Proof of Concept
PR	Performance Requirement
PT	Performance Test
SRS	Software Requirements Specification
SR	Security Requirement
ST	Security Test
UH	Usability and Humanity
UI	User Interface
VnV	Verification and Validation

Contents

1 Revision History **i**

2 Symbols, Abbreviations and Acronyms **ii**

3 Functional Requirements Evaluation **1**

3.1 Image Upload 1
3.2 Image Identification 3
3.3 Call and Fetch API Response 4
3.4 Logging Data 5
3.5 Nutritional Facts Display 6
3.6 Text Upload 8
3.7 Voice Upload 13
3.8 Profile Page 21
3.9 Taskbar 28

4 Nonfunctional Requirements Evaluation **30**

4.1 Look and Feel 30
4.2 Usability 31
4.3 Performance 36
4.4 Operational and Environmental 43
4.5 Maintainability and Support 44
4.6 Security 45
4.7 Cultural 47
4.8 Legal 47

5 Comparison to Existing Implementation **49**

6 Unit Testing **49**

7 Changes Due to Testing **49**

7.1 Changes Due to Functional Requirements Testing 49
7.2 Changes Due to Usability Testing 49

8 Automated Testing **50**

9 Trace to Requirements **51**

10 Trace to Modules **62**

11 Code Coverage Metrics **65**

List of Tables

1 Traceability between Tests and Functional Requirements . . . 51
2 Traceability between Tests and NonFunctional Requirements . 59
3 Traceability between Modules and Functional Tests 63

List of Figures

The purpose of this document is to display the results of Utrition's testing. All tests were previously discussed in the VnV plan. Based on the results of the tests, changes being made to Utrition's implementation is also provided.

3 Functional Requirements Evaluation

3.1 Image Upload

1. Test Name: one-upload-1

   Type: Functional, Dynamic, and Manual.

   Initial State: User is on the "Upload" page with "Image Upload" selected.

   Input: A User clicks on "Select Image" and selects the image file "apple.jpeg" found in the utrition/test/testPhotos directory. User clicks on "Text Upload" and then clicks on "Image Upload" and then clicks on the "Submit" button.

   Expected Results: The system accepts the image upload and returns the nutritional information for an apple.

   How The Test Was Performed: The tester opened Utrition and clicked on "Upload" found in the taskbar. The tester clicked on "Image Upload" and then "Select Image". The tester selects "apple.jpeg" found in the utrition/test/testPhotos directory. The tester clicks on the "Image Upload" button and then clicks on the "Submit" button.

   Actual Results: Test passed. The actual results matched up with the expected results.

2. Test Name: one-upload-2

   Type: Functional, Dynamic, and Manual.

   Initial State: No image in the system. User is on the "Upload" page with "Image Upload" selected.

   Input: A User clicks on the "Submit" button.

   Expected Results: No nutritional information will be returned to the user.

How The Test Was Performed: The tester opened Utrition and clicked on “Upload” found in the taskbar. The tester clicked on “Image Upload” and then the “Submit” button.

Actual Results: Test passed. The actual results matched up with the expected results.

3. Test Name: one-upload-auto-1

   Type: Functional, Dynamic, Automatic.

   Initial State: The system contains a generated JPG image of red pixels with dimensions 128 x 128.

   Input: The generated image is passed as an argument to interface.open(), with the result saved to the variable ’result’.

   Expected Results: The variable ’result’ is not None and the userFlag for interface has been set to 1. There is also exception handling for an UnpicklingError, an error that can occur on some computers that is unrelated to the scope of the test.

   How The Test Was Performed: The generated image is passed to interface.open(), which is the starting point for the machine learning image classification pipeline. If the test returns not None that means the pipeline executed successfully for the passed file. If the userFlag is set to 1 that means that interface.open() was passed some data.

   Actual Results: Test passed. The actual results matched up with the expected results.

4. Test Name: one-upload-auto-2

   Type: Functional, Dynamic, Automatic.

   Initial State: The system contains a generated txt file with text “test content”.

   Input: The generated text file is passed as an argument to interface.open().

   Expected Results: An UnidentifiedImageError is thrown and the userFlag for interface has been set to 1.

   How The Test Was Performed: The generated text file is passed to interface.open(), which is the starting point for the machine learning

image classification pipeline. If the test throws an UnidentifiedImageError that means that the format of the data passed is not valid for image classification. If the userFlag is set to 1 that means that interface.open() was passed some data.

Actual Results: Test passed. The actual results matched up with the expected results.

3.2 Image Identification

1. Test Name: image-identification-1

   Type: Functional and Automatic.

   Initial State: The system contains a generated PNG image of red pixels with dimensions 128 x 128.

   Input: The generated image is passed as an argument to interface.open(), with the result saved to the variable 'result'.

   Expected Results: The variable 'result' is a string. There is also exception handling for an UnpicklingError, an error that can occur on some computers that is unrelated to the scope of the test.

   How The Test Was Performed: The generated image is passed to interface.open(), which is the starting point for the machine learning image classification pipeline. If the test returns a string that means that the pipeline not only executed successfully, but also returned an image classification result for the submitted image.

   Actual Results: Test passed. The actual results matched up with the expected results.

2. Test Name: image-identification-2

   Type: Functional and Automatic.

   Initial State: The system contains a generated JPG image of red pixels with dimensions 128 x 128.

   Input: The generated image is passed as an argument to interface.open(), with the result saved to the variable 'result'.

   Expected Results: The variable 'result' is a string. There is also exception handling for an UnpicklingError, an error that can occur on some computers that is unrelated to the scope of the test.

How The Test Was Performed: The generated image is passed to interface.open(), which is the starting point for the machine learning image classification pipeline. If the test returns a string that means that the pipeline not only executed successfully, but also returned an image classification result for the submitted image.

Actual Results: Test passed. The actual results matched up with the expected results.

3. Test Name: image-identification-3

   Type: Functional and Automatic.

   Initial State: The system contains a generated JPEG image of red pixels with dimensions 128 x 128.

   Input: The generated image is passed as an argument to interface.open(), with the result saved to the variable 'result'.

   Expected Results: The variable 'result' is a string. There is also exception handling for an UnpicklingError, an error that can occur on some computers that is unrelated to the scope of the test.

   How The Test Was Performed: The generated image is passed to interface.open(), which is the starting point for the machine learning image classification pipeline. If the test returns a string that means that the pipeline not only executed successfully, but also returned an image classification result for the submitted image.

   Actual Results: Test passed. The actual results matched up with the expected results.

3.3 Call and Fetch API Response

1. Test Name: api-1

   Type: Functional, Automatic.

   Initial State: The system contains a mocked JSON response for the food "banana".

   Input: A request is made to get_nutritional_data() with the argument "banana" and the result is stored in the variable "result".

   Expected Results: The variable "result" will be a list of length 1 and will match the mocked data for "banana". Additionally, the contents

of “result” will be able to be converted to JSON format without any errors.

How The Test Was Performed: Nutritional JSON data is mocked in our system for the food “banana”, meaning that when a request is made to our nutritional data fetcher for “banana”, it will return the mocked result if the request processes successfully. The returning value will be a list of length 1, signifying that one JSON object was returned for the input “banana”.

Actual Results: Test passed. The actual results matched up with the expected results.

2. Test Name: axios-call-1

   Type: Functional, Automatic.

   Initial State: The text upload functionality has not yet been used.

   Input: The tester will attempt to make an axios call

   Expected Results: The tester will receive an http request back from the backend.

   How The Test Was Performed: Tester will create mocked text upload function to test to see if the front end can successfully make and receive http calls.

   Actual Results: Test passed. The actual results matched up with the expected results.

3.4 Logging Data

1. Test Name: log-data-read-data-1

   Type: Functional, Automatic.

   Initial State: The system contains a list of one object of sample JSON data.

   Input: The sample data is logged to a CSV file and then a request to read the CSV file is made with the results of the request saved to the variable “lines”.

   Expected Results: The variable “lines” will be of length 2. The value at the second index of the list will equal the sample data when converted from a list format to a JSON object format.

How The Test Was Performed: The sample data is sent to our CSV logger function to create an entry for the data as well as its header row detailing the keys of the key value JSON pairings. To account for this header row, after we read the CSV file line by line with the built-in Python function readlines(), we check that the length of the returning list is 2.

Actual Results: The actual results matched up with the expected results.

3.5 Nutritional Facts Display

1. Test Name: current-nutrition-1

   Type: Functional, Dynamic, Automatic.

   Initial State: The system contains a generated JPEG image of red pixels with dimensions 128 x 128. The system contains a list of one object of mocked JSON data. The classification of the sample image and the food_name of the mocked JSON data are set to the same value, linking the two pieces of data.

   Input: A nested function call is made to get_nutrition_data() with an argument interface.open() whose own argument is the generated image. The result of this call is saved to the variable 'result'.

   Expected Results: The variable "result" will be a list of length 1. The value of "result" will be equal to the mocked JSON data. The value of "result" will successfully convert to JSON format without any errors. There is also exception handling for an UnpicklingError, an error that can occur on some computers that is unrelated to the scope of the test.

   How The Test Was Performed: This is the overall black box test of our image classification system. It tests the whole process of classifying an image and retrieving JSON nutrition data for that image classification. If the test returns a list of one item, it means that the image was successfully classified and that the classification was equal to the food_name in the mocked JSON data.

   Actual Results: Test passed. The actual results matched up with the expected results.

2. Test Name: past-nutrition-text-read-data-2

Type: Functional, Dynamic, Automatic.

Initial State: The system contains a list of four distinct object of sample JSON data.

Input: The sample data is logged to a CSV file and then the read_file() function is called with its result saved to the variable "result".

Expected Results: The variable "result" will be of length 4.

How The Test Was Performed: The sample data is sent to our CSV logger function to create entries for each data as well as an overall header row detailing the keys of the key value JSON pairings. Our read_file() function then reads the CSV file, ignoring the header row, and returns a sorted list of data objects. Since the header row is ignored, we can check that the length of the list of data returned is exactly equal to the number of data points we had in our sample data.

Actual Results: Test passed. The actual results matched up with the expected results.

3. Test Name: display-elements-1

   Type: Functional, Automatic.

   Initial State: Tester will be located on a page of the Utrition application.

   Input: The tester will attempt to render page to see if the necessary elements exist.

   Expected Results: The framework replies with if elements exist.

   How The Test Was Performed: the tester will mock the Utrition application home page screen and will check to see if the elements exist for the user.

   Actual Results: Test passed. The actual results matched up with the expected results.

4. Test Name: correct-path-1

   Type: Functional, Automatic.

   Initial State: Tester will start on home page.

   Input: Tester will click hyperlink to go to a different page.

Expected Results: The tester will be taken to the new page specified in hyperlink.

How The Test Was Performed: The tester will attempt to mock a user input event clicking onto the specified hyperlink area by rendering a mocked version of the home page. Upon clicking the hyperlink area of the profile page, the testing framework should reply with a rendered profile page.

Actual Results: Test passed. The actual results matched up with the expected results.

3.6 Text Upload

1. Test Name: text-upload-1

   Type: Functional, Dynamic, Manual.

   Initial State: User is on the "Upload" page with "Text Upload" selected.

   Input: User inputs "cup of yogurt" into the text upload box and then clicks on the "Submit" button

   Expected Results: The nutritional facts for a cup of yogurt are displayed to the user.

   How The Test Was Performed: The tester opened Utrition and clicked on "Upload" found in the taskbar. The tester typed "cup of yogurt" into the text upload box and pressed the "Submit" button.

   Actual Results: Test passed. The actual results matched up with the expected results.

2. Test Name: text-upload-2

   Type: Functional, Dynamic, Manual.

   Initial State: User is on the "Upload" page with "Text Upload" selected.

   Input: User inputs "i had and ate 3 pieces of shumai" into the text upload box and then clicks on the "Submit" button

   Expected Results: The nutritional facts for 3 pieces of shumai is displayed to the user.

How The Test Was Performed: The tester opened Utrition and clicked on “Upload” found in the taskbar. The tester typed “i had and ate 3 pieces of shumai” into the text upload box and pressed the “Submit” button.

Actual Results: Test passed. The actual results matched up with the expected results.

3. Test Name: text-upload-3

   Type: Functional, Dynamic, Manual.

   Initial State: User is on the “Upload” page with “Text Upload” selected.

   Input: User inputs “avacadoe” into the text upload box and then clicks on the “Submit” button

   Expected Results: No nutritional facts will be displayed to the user.

   How The Test Was Performed: The tester opened Utrition and clicked on “Upload” found in the taskbar. The tester typed “avacadoe” into the text upload box and pressed the “Submit” button.

   Actual Results: Test passed. The actual results matched up with the expected results.

4. Test Name: text-upload-4

   Type: Functional, Dynamic, Manual.

   Initial State: User is on the “Upload” page with “Text Upload” selected.

   Input: User inputs nothing into the text upload box and then clicks on the “Submit” button

   Expected Results: No nutritional facts will be displayed to the user.

   How The Test Was Performed: The tester opened Utrition and clicked on “Upload” found in the taskbar. The tester pressed the “Submit” button.

   Actual Results: Test passed. The actual results matched up with the expected results.

5. Test Name: text-upload-5

   Type: Functional, Dynamic, Manual.

   Initial State: User is on the "Upload" page with "Text Upload" selected.

   Input: User inputs "chiken" into the text upload box and then clicks on the "Submit" button

   Expected Results: No nutritional facts will be displayed to the user.

   How The Test Was Performed: The tester opened Utrition and clicked on "Upload" found in the taskbar. The tester typed "chiken" into the text upload box and pressed the "Submit" button.

   Actual Results: Test failed. The nutritional facts for 3 ounces of chicken were displayed to the user.

6. Test Name: text-upload-6

   Type: Functional, Dynamic, Manual.

   Initial State: User is on the "Upload" page with "Text Upload" selected.

   Input: User inputs "oreo mcflurry" into the text upload box and then clicks on the "Submit" button

   Expected Results: The nutritional facts for an Oreo McFlurry are displayed to the user.

   How The Test Was Performed: The tester opened Utrition and clicked on "Upload" found in the taskbar. The tester typed "oreo mcflurry" into the text upload box and pressed the "Submit" button.

   Actual Results: Test failed. The nutritional facts for an Oreo were displayed in combination with the nutritional facts for a McFlurry instead of an Oreo McFlurry.

7. Test Name: text-upload-7

   Type: Functional, Dynamic, Manual.

   Initial State: User is on the "Upload" page with "Text Upload" selected.

   Input: User inputs "yuhddasdihijdsahdaihdhi chicken shawarma wrap asgyudy1hu13h123hej" into the text upload box and then clicks on the

"Voice Upload" button. User clicks on the "Text Upload" button and clicks "Submit"

Expected Results: The nutritional facts for a chicken shawarma wrap are displayed to the user.

How The Test Was Performed: The tester opened Utrition and clicked on "Upload" found in the taskbar. The tester typed "yuhddasdihijdsahdaihdhi chicken shawarma wrap asgyudy1hu13h123hej" into the text upload box and pressed the "Voice Upload" button. The tester then clicked on the "Text Upload" button and then the "Submit" button.

Actual Results: Test passed. The actual results matched up with the expected results.

8. Test Name: text-upload-8

   Type: Functional, Dynamic, Manual.

   Initial State: User is on the "Upload" page with "Text Upload" selected.

   Input: User inputs "half a McDonald's cheeseburger" into the text upload box and then clicks on the "Submit" button

   Expected Results: The nutritional facts for half of a McDonald's cheeseburger are displayed to the user.

   How The Test Was Performed: The tester opened Utrition and clicked on "Upload" found in the taskbar. The tester typed "half a McDonald's cheeseburger" into the text upload box and pressed the "Submit" button.

   Actual Results: Test failed. The nutritional facts for half of a standard cheeseburger were displayed to the user instead of a McDonald's cheeseburger.

9. Test Name: multi-text-upload-1

   Type: Functional, Dynamic, Manual.

   Initial State: User is on the "Upload" page with "Text Upload" selected.

   Input: User inputs "apple" and "banana" on two separate lines into the text upload box and then clicks on the "Submit" button.

Expected Results: The nutritional facts for an apple and a banana are displayed to the user.

How The Test Was Performed: The tester opened Utrition and clicked on "Upload" found in the taskbar. The tester typed "apple" into the text upload box and then pressed enter. The tester typed "banana" and pressed the "Submit" button.

Actual Results: Test failed. No nutritional information was displayed to the user.

10. Test Name: multi-text-upload-2

    Type: Functional, Dynamic, Manual.

    Initial State: User is on the "Upload" page with "Text Upload" selected.

    Input: User inputs "3 kit kat bars, 100g of salt, and 2 cucumbers" into the text upload box and then clicks on the "Submit" button.

    Expected Results: The nutritional facts for 3 Kit Kat bars, 100g of salt, and 2 cucumbers are displayed to the user.

    How The Test Was Performed: The tester opened Utrition and clicked on "Upload" found in the taskbar. The tester typed "3 kit kat bars, 100g of salt, and 2 cucumbers" into the text upload box and then pressed the "Submit" button.

    Actual Results: Test passed. The actual results matched up with the expected results.

11. Test Name: multi-text-upload-3

    Type: Functional, Dynamic, Manual.

    Initial State: User is on the "Upload" page with "Text Upload" selected.

    Input: User inputs "i ate two pieces of chicken breast... also had 100g of a martini. . . . . 2 Cheeseburger!!!!!11" into the text upload box and then clicks on the "Submit" button.

    Expected Results: The nutritional facts for 2 pieces of chicken breast, 100g of a martini, and 2 cheeseburgers are displayed to the user.

How The Test Was Performed: The tester opened Utrition and clicked on "Upload" found in the taskbar. The tester typed "i ate two pieces of chicken breast... also had 100g of a martini. . . . . 2 Cheeseburger!!!!!11" into the text upload box and then pressed the "Submit" button.

Actual Results: Test passed. The actual results matched up with the expected results.

12. Test Name: multi-text-upload-4

    Type: Functional, Dynamic, Manual.

    Initial State: User is on the "Upload" page with "Text Upload" selected.

    Input: User inputs "ubdhsjsbubahasjvdhasbhdabhsdbh asbdh ahbsdbajsdbhashbdb 1 slice of cheese b2b23b2b3b2b3b2b 2 grapefruits aaahhhhh 100g of apple sauce ajjdjj2j2j3j23j who goes there? yup its me! aaa 1 slice of white bread celery" into the text upload box and then clicks on the "Submit" button.

    Expected Results: The nutritional facts for 1 slice of cheese and 2 grapefruits, 100g of apple sauce, 1 slice of white bread, and 1 celery are displayed to the user.

    How The Test Was Performed: The tester opened Utrition and clicked on "Upload" found in the taskbar. The tester typed "ubdhsjsbubahasjvdhasbhdabhsdbh asbdh ahbsdbajsdbhashbdb 1 slice of cheese b2b23b2b3b2b3b2b 2 grapefruits aaahhhhh 100g of apple sauce ajjdjj2j2j3j23j who goes there? yup its me! aaa 1 slice of white bread celery" into the text upload box and then pressed the "Submit" button.

    Actual Results: Test passed. The actual results matched up with the expected results.

3.7 Voice Upload

1. Test Name: voice-upload-1

   Type: Functional, Dynamic, Manual.

   Initial State: User is on the "Upload" page with "Voice Upload" selected.

Input: User clicks on the "Start Talking" button and says out loud "I ate a slice of apple pie". User clicks on the "Submit" button.

Expected Results: The nutritional facts for a slice of apple pie are displayed to the user.

How The Test Was Performed: The tester opened Utrition and clicked on "Upload" found in the taskbar. The tester clicked on the "Voice Upload" button and then clicked on the "Start Talking" button. The tester verbally said "I ate a slice of apple pie." and pressed the "Submit" button.

Actual Results: Test passed. The actual results matched up with the expected results.

2. Test Name: voice-upload-2

   Type: Functional, Dynamic, Manual.

   Initial State: User is on the "Upload" page with "Voice Upload" selected.

   Input: User clicks on the "Start Talking" button and says out loud "I ate two pieces of Kimchi period king yes". User clicks on the "Stop Talking" button and then clicks on the "Submit" Button.

   Expected Results: The nutritional facts for 2 pieces of Kimchi are displayed to the user.

   How The Test Was Performed: The tester opened Utrition and clicked on "Upload" found in the taskbar. The tester clicked on the "Voice Upload" button and then clicked on the "Start Talking" button. The tester verbally said "I ate two pieces of Kimchi period king yes" and pressed the "Stop Talking" button. Then the tester pressed the "Submit" button.

   Actual Results: Test passed. The actual results matched up with the expected results.

3. Test Name: voice-upload-3

   Type: Functional, Dynamic, Manual.

   Initial State: User has just completed test case voice-upload-2.

Input: User clicks on the "Start Talking" button and says out loud "three chicken wings". User clicks on the "Stop Talking" button and then clicks on the "Submit" Button.

Expected Results: The nutritional facts for 2 pieces of Kimchi and 3 chicken wings was displayed to the user.

How The Test Was Performed: After the tester completed the steps laid out in test case voice-upload-2, the tester clicked on the "Start Talking" button. The tester verbally said "three chicken wings" and pressed the "Stop Talking" button. Then the tester pressed the "Submit" button.

Actual Results: Test passed. The actual results matched up with the expected results.

4. Test Name: voice-upload-4

   Type: Functional, Dynamic, Manual.

   Initial State: User has just completed test case voice-upload-3.

   Input: User clicks on the "Start Talking" button and then the "Reset" button.

   Expected Results: The "Upload" page refreshes and resets, which means the user is met with the "Text Upload" button selected. After clicking on the "Voice Upload" button, the user sees no previous entries and is able to click the "Start Talking" button.

   How The Test Was Performed: After the tester completed the steps laid out in test case voice-upload-3, the tester then clicked on the "Start Talking" button and then clicked on the "Reset" button. The tester clicked on the "Voice Upload" button again to double check the page was cleared.

   Actual Results: Test passed. The actual results matched up with the expected results.

5. Test Name: voice-upload-5

   Type: Functional, Dynamic, Manual.

   Initial State: User is on the "Upload" page with "Voice Upload" selected.

   Input: User clicks on the "Reset" button.

Expected Results: The "Upload" page refreshes and resets, which means the user is met with the "Text Upload" button selected. After clicking on the "Voice Upload" button, the user is met with the normal "Voice Upload" page.

How The Test Was Performed: The tester opened Utrition and clicked on "Upload" found in the taskbar. The tester clicked on the "Voice Upload" button and then clicked on the "Reset" button. The tester clicked on the "Voice Upload" button again to double check the page was cleared.

Actual Results: Test passed. The actual results matched up with the expected results.

6. Test Name: voice-upload-6

   Type: Functional, Dynamic, Manual.

   Initial State: User is on the "Upload" page and has typed "apple" in the text box found in "Text Upload".

   Input: User clicks on "Voice Upload" and then User clicks on the "Reset" button.

   Expected Results: The "Upload" page refreshes and resets, which means the user is met with the "Text Upload" button selected. The user is able to see "apple" in the text box.

   How The Test Was Performed: The tester opened Utrition and clicked on "Upload" found in the taskbar. The tester typed "apple" in the text box. The tester clicked on the "Voice Upload" button and then clicked on the "Reset" button.

   Actual Results: Test failed. "apple" no longer appears in the text box in "Text Upload".

7. Test Name: voice-upload-7

   Type: Functional, Dynamic, Manual.

   Initial State: User is on the "Upload" page with "Voice Upload" selected.

   Input: User clicks on the "Submit" button.

   Expected Results: No nutritional facts will be displayed to the user.

How The Test Was Performed: The tester opened Utrition and clicked on “Upload” found in the taskbar. The tester clicked on the “Voice Upload” button and then clicked on the “Submit” button.

Actual Results: Test passed. The actual results matched up with the expected results.

8. Test Name: voice-upload-8

   Type: Functional, Dynamic, Manual.

   Initial State: User is on the “Upload” page with “Voice Upload” selected.

   Input: User clicks on the “Start Talking” button and says out loud “Hey now, you’re a rock star. Get the show on, get paid”. User clicks on the “Stop Talking” button and then clicks on the “Submit” Button.

   Expected Results: No nutritional facts will be displayed to the user.

   How The Test Was Performed: The tester opened Utrition and clicked on “Upload” found in the taskbar. The tester clicked on the “Voice Upload” button and then clicked on the “Start Talking” button. The tester verbally said “Hey now, you’re a rock star. Get the show on, get paid” and pressed the “Stop Talking” button. Then the tester pressed the “Submit” button.

   Actual Results: Test passed. The actual results matched up with the expected results.

9. Test Name: voice-upload-9

   Type: Functional, Dynamic, Manual.

   Initial State: User has just completed test case voice-upload-8.

   Input: User clicks on the “Start Talking” button and says out loud “a can of sprite”. User clicks on the “Stop Talking” button and then clicks on the “Submit” Button.

   Expected Results: The nutritional facts for a can of sprite are displayed to the user.

   How The Test Was Performed: After the tester completed the steps laid out in test case voice-upload-9, the tester clicked on the “Start Talking” button. The tester verbally said “a can of sprite” and pressed the “Stop Talking” button. Then the tester pressed the “Submit” button.

Actual Results: Test passed. The actual results matched up with the expected results.

10. Test Name: voice-upload-10

    Type: Functional, Dynamic, Manual.

    Initial State: User is on the "Upload" page with "Voice Upload" selected.

    Input: User clicks on the "Start Talking" button and says out loud "Ahhh! Don't scare me like that. a pound of white rice. so by the way im vegan lol". User clicks on the "Stop Talking" button and then clicks on the "Submit" Button.

    Expected Results: The nutritional facts for a pound of white rice are displayed to the user.

    How The Test Was Performed: The tester opened Utrition and clicked on "Upload" found in the taskbar. The tester clicked on the "Voice Upload" button and then clicked on the "Start Talking" button. The tester verbally said "Ahhh! Don't scare me like that. a pound of white rice. so by the way im vegan lol" and pressed the "Stop Talking" button. Then the tester pressed the "Submit" button.

    Actual Results: Test passed. The actual results matched up with the expected results.

11. Test Name: voice-upload-11

    Type: Functional, Dynamic, Manual.

    Initial State: User is on the "Upload" page with "Voice Upload" selected.

    Input: User clicks on the "Start Talking" button and says out loud "I ate 2 eggs benedicts". User clicks on "Stop Talking" and then clicks on "Text Upload". User clicks on "Voice Upload" and then the "Submit" button.

    Expected Results: The nutritional facts for 2 eggs benedicts are displayed to the user.

    How The Test Was Performed: The tester opened Utrition and clicked on "Upload" found in the taskbar. The tester clicked on the "Voice Upload" button and then clicked on the "Start Talking" button. The

tester verbally said “I ate 2 eggs benedicts” and pressed the “Stop Talking” button. Then the tester pressed on the “Text Upload” button and then the “Voice Upload” button. The tester pressed the “Submit” button.

Actual Results: Test passed. The actual results matched up with the expected results.

12. Test Name: voice-upload-12

    Type: Functional, Dynamic, Manual.

    Initial State: User is on the “Upload” page with “Voice Upload” selected.

    Input: User clicks on the “Start Talking” button and says out loud “2 oranges”. User clicks on “Text Upload”. User says out loud “a slice of an apple” User clicks on “Voice Upload” and then the “Submit” button.

    Expected Results: The nutritional facts for 2 oranges are displayed to the user.

    How The Test Was Performed: The tester opened Utrition and clicked on “Upload” found in the taskbar. The tester clicked on the “Voice Upload” button and then clicked on the “Start Talking” button. The tester verbally said “2 oranges” and then the tester pressed on the “Text Upload” button. The tester said “a slice of an apple” and then clicked on the “Voice Upload” button. The tester pressed the “Submit” button.

    Actual Results: Test failed. The nutritional facts for 2 oranges and a slice of an apple were displayed to the user.

13. Test Name: multi-voice-upload-1

    Type: Functional, Dynamic, Manual.

    Initial State: User is on the “Upload” page with “Voice Upload” selected.

    Input: User clicks on the “Start Talking” button and says out loud “Today I ate 3 spoons of peanut butter, I know im a mess. I also had 3 whole baguettes with a side of 100 G of salsa. Then to top it all off I ate 1 carrot cake.” User clicks the “Submit” button.

Expected Results: The nutritional facts for 3 spoons of peanut butter, 3 whole baguettes, 100g of salsa, and 1 whole carrot cake are displayed to the user.

How The Test Was Performed: The tester opened Utrition and clicked on "Upload" found in the taskbar. The tester clicked on the "Voice Upload" button and then clicked on the "Start Talking" button. The tester verbally said "Today I ate 3 spoons of peanut butter, I know im a mess. I also had 3 whole baguettes with a side of 100 G of salsa. Then to top it all off I ate 1 carrot cake". The tester pressed the "Submit" button.

Actual Results: Test passed. The actual results matched up with the expected results.

14. Test Name: multi-voice-upload-2

    Type: Functional, Dynamic, Manual.

    Initial State: User is on the "Upload" page with "Voice Upload" selected.

    Input: User clicks on the "Start Talking" button and says out loud "100 grams of hummus, 2 bowls of mac and cheese". The user pauses for 5 seconds and then says "2 teaspoons of vegetable oil". User clicks the "Submit" button.

    Expected Results: The nutritional facts for 100g of hummus, 2 bowls of mac and cheese, and 2 teaspoons of vegetable oil are displayed to the user.

    How The Test Was Performed: The tester opened Utrition and clicked on "Upload" found in the taskbar. The tester clicked on the "Voice Upload" button and then clicked on the "Start Talking" button. The tester verbally said "100 grams of hummus, 2 bowls of mac and cheese". The tester paused for 5 seconds and said "2 teaspoons of vegetable oil". The tester pressed the "Submit" button.

    Actual Results: Test passed. The actual results matched up with the expected results.

15. Test Name: multi-voice-upload-3

    Type: Functional, Dynamic, Manual.

Initial State: User is on the "Upload" page with "Voice Upload" selected.

Input: User clicks on the "Start Talking" button and says out loud "ausdhjasdihjashduih (gibberish) 2 grams of Pina Colada ausdhjasdihjashduih (gibberish) penguin 1 pineapple ausdhjasdihjashduih (gibberish)". User clicks the "Submit" button.

Expected Results: The nutritional facts for 2g of Pina Colada and 1 whole pineapple are displayed to the user.

How The Test Was Performed: The tester opened Utrition and clicked on "Upload" found in the taskbar. The tester clicked on the "Voice Upload" button and then clicked on the "Start Talking" button. The tester verbally said "ausdhjasdihjashduih (gibberish) 2 grams of Pina Colada ausdhjasdihjashduih (gibberish) penguin 1 pineapple ausdhjasdihjashduih (gibberish)". The tester pressed the "Submit" button.

Actual Results: Test passed. The actual results matched up with the expected results.

3.8 Profile Page

1. Test Name: profile-1

   Type: Functional, Dynamic, Manual.

   Initial State: User has completed test case "text-upload-1".

   Input: User clicks on "Profile" found in the taskbar.

   Expected Results: The user is able to see that their most eaten food is yogurt, their total calories consumed for today is 142.88 (calories in 1 cup of yogurt), and the nutritional values of yogurt next to the current date.

   How The Test Was Performed: After the tester completed the steps laid out in test case text-upload-1, the tester clicked on "Profile" found in the taskbar.

   Actual Results: Test passed. The actual results matched up with the expected results.

2. Test Name: profile-2

Type: Functional, Dynamic, Manual.

Initial State: User has completed test case "text-upload-1" three times.

Input: User clicks on "Profile" found in the taskbar.

Expected Results: The user is able to see that their most eaten food is yogurt, their total calories consumed for today is 428.64, the nutritional values of yogurt next to the current date is shown three times, and the total calories of three cups of yogurt is summed on the right table.

How The Test Was Performed: After the tester completed the steps laid out in test case text-upload-1 three times, the tester clicked on "Profile" found in the taskbar.

Actual Results: Test passed. The actual results matched up with the expected results.

3. Test Name: profile-3

   Type: Functional, Dynamic, Manual.

   Initial State: User has completed test case "profile-2".

   Input: User completes test case "multi-text-upload-3" and then clicks on "Profile" found in the taskbar.

   Expected Results: The user is able to see that their most eaten food is yogurt, their total calories consumed for today is 2137.12, the nutritional values of 2 chicken breasts next to the current date, the nutritional value of 100g of a martini next to the current date, the nutritional value of 2 cheeseburgers next to the current date, and the nutritional value of a cup of yogurt next to the current date (in that order). The user is able to see the total calories of 3 cups of yogurt, 2 chicken breasts, 100g of a martini, and 2 cheeseburgers is summed on the right table.

   How The Test Was Performed: After the tester completed the steps laid out in test case profile-2, the tester completed test case "multi-text-upload-3". The tester clicked on "Profile" found in the taskbar.

   Actual Results: Test passed. The actual results matched up with the expected results.

4. Test Name: profile-4

Type: Functional, Dynamic, Manual.

Initial State: User has completed test case "profile-3".

Input: User clicks on "Upload" found in the taskbar. User clicks on "Voice Upload" and then "Start Talking". User says "2 cheeseburgers random random random how was your day?" and then clicks on "Stop Talking". User clicks on the "Submit" button and clicks on "Profile" found in the taskbar.

Expected Results: The user is able to see that their most eaten food is a cheeseburger, their total calories consumed for today is 3207.74, the nutritional value of 2 cheeseburgers next to the current date, the nutritional values of 2 chicken breasts next to the current date, the nutritional value of 100g of a martini next to the current date, and the nutritional value of 2 cheeseburgers next to the current date (in that order). The user is able to see the total calories of 3 cups of yogurt, 2 chicken breasts, 100g of a martini, and 4 cheeseburgers is summed on the right table.

How The Test Was Performed: After the tester completed the steps laid out in test case profile-3, the tester clicked on "Upload" in the taskbar. The tester clicked on "Voice Upload" and then "Start Talking". Tester said "2 cheeseburgers random random random how was your day?" and then clicked on "Stop Talking". Tester clicked on the "Submit" button. The tester clicked on "Profile" found in the taskbar.

Actual Results: Test failed. The most eaten food is still yogurt instead of cheeseburger due to yogurt being inputted 3 different times and cheeseburger being inputted twice.

5. Test Name: profile-5

   Type: Functional, Dynamic, Manual.

   Initial State: User has completed test case "profile-4" and is viewing the "Upload" page

   Input: User clicks on "Text Upload" and then types "cheeseburger and cheeseburger". User clicks on the "Submit" button and clicks on "Profile" found in the taskbar.

   Expected Results: The user is able to see that their most eaten food is a cheeseburger, their total calories consumed for today is 4278.36,

the nutritional value of 1 cheeseburger next to the current date, the nutritional values of 1 cheeseburger next to the current date, the nutritional value of 2 cheeseburgers next to the current date, and the nutritional value of 1 chicken breast next to the current date (in that order). The user is able to see the total calories of 3 cups of yogurt, 2 chicken breasts, 100g of a martini, and 6 cheeseburgers is summed on the right table.

How The Test Was Performed: After the tester completed the steps laid out in test case profile-4, the tester clicked on "Upload" found in the taskbar. The tester clicked on "Text Upload" and then typed "cheeseburger and cheeseburger". Tester clicked on the "Submit" button. The tester clicked on "Profile" found in the taskbar.

Actual Results: Test failed. The most eaten food is still yogurt instead of cheeseburger due to 2 cheeseburgers and 1 cheeseburger being seen by the system as two different food items.

6. Test Name: profile-6

   Type: Functional, Dynamic, Manual.

   Initial State: User has completed test case "profile-5" and is viewing the "Profile" page.

   Input: User clicks on "Look at next 4 entries".

   Expected Results: User is able to see the nutritional information for: 100g of a martini, 2 cheeseburgers, a cup of yogurt, and another cup of yogurt all next to the current date.

   How The Test Was Performed: After the tester completed the steps laid out in test case profile-5, the tester clicked on the "Look at next 4 entries" button.

   Actual Results: Test passed. The actual results matched up with the expected results.

7. Test Name: profile-7

   Type: Functional, Dynamic, Manual.

   Initial State: User has completed test case "profile-6" and is viewing the "Profile" page.

   Input: User clicks on "Look at next 4 entries".

Expected Results: User is able to see the nutritional information for 1 cup of yogurt.

How The Test Was Performed: After the tester completed the steps laid out in test case profile-6, the tester clicked on the "Look at next 4 entries" button.

Actual Results: Test passed. The actual results matched up with the expected results.

8. Test Name: profile-8

   Type: Functional, Dynamic, Manual.

   Initial State: User has completed test case "profile-7" and is viewing the "Profile" page.

   Input: User clicks on "Look at previous 4 entries".

   Expected Results: User is able to see the nutritional information for: 100g of a martini, 2 cheeseburgers, a cup of yogurt, and another cup of yogurt all next to the current date.

   How The Test Was Performed: After the tester completed the steps laid out in test case profile-7, the tester clicked on the "Look at previous 4 entries" button.

   Actual Results: Test passed. The actual results matched up with the expected results.

9. Test Name: profile-9

   Type: Functional, Dynamic, Manual.

   Initial State: User has inputted a cup of yogurt into Utrition for 15 days and is now viewing the "Profile" page.

   Input: User clicks on the "Previous Week" button.

   Expected Results: The table is updated to show the previous 7 days in which a cup of yogurt was inputted into Utrition.

   How The Test Was Performed: The tester went into the directory "utrition\src\utrition\utrition-backend" and opened the "nutrition_log.csv" file. The tester faked data for yogurt by making sure each column was filled out with appropriate data (Time having format d/m/Y H:M:S, Name must be a string, serving unit must be a string, and the rest filled

with floats), and then copying and pasting the data for 14 other rows (15 in total). Each row had a different date associated with it. The tester saved the "nutrition_log.csv" changes and refreshed the "Profile" page. The tester clicked on the "Previous Week" button.

Actual Results: Test passed. The actual results matched up with the expected results.

10. Test Name: profile-10

    Type: Functional, Dynamic, Manual.

    Initial State: User has completed test case "profile-9" and is viewing the "Profile" page.

    Input: User clicks on the "Previous Week" button.

    Expected Results: The table is updated to show the first day the user inputted a food item (yogurt) into Utrition.

    How The Test Was Performed: After the tester completed the steps laid out in test case profile-9, the tester clicked on the "Previous Week" button.

    Actual Results: Test passed. The actual results matched up with the expected results.

11. Test Name: profile-11

    Type: Functional, Dynamic, Manual.

    Initial State: Food items were inputted into Utrition on 15 different days. In increasing order, the following food items were inputted: Apple, Banana, Celery, Orange, Grapefruit, Chicken, Cheeseburger, Pizza, Sprite, Cashew, Cucumber, Pickle, Poutine, Sour Cream, and Salsa.

    Input: User clicks on the "Previous Week" button twice. Then, the user clicks on the "Next Week" button once.

    Expected Results: The table is updated to show the following food items with their attached date and calorie information: Pizza, Cheeseburger, Chicken, Grapefruit, Orange, Celery, and Banana.

    How The Test Was Performed: The tester went into the directory "utrition\src\utrition\utrition-backend" and opened the "nutrition_log.csv"

file. The tester faked data for the food items by making sure each column was filled out with appropriate data (Time having format d/m/Y H:M:S, serving unit must be a string, and the rest filled with floats), and then copying and pasting the data for 14 other rows (15 in total). Each row had a different date associated with it. The tester added in the "Name" column the following 15 food items in increasing order by date: Apple, Banana, Celery, Orange, Grapefruit, Chicken, Cheeseburger, Pizza, Sprite, Cashew, Cucumber, Pickle, Poutine, Sour Cream, and Salsa. The tester saved the "nutrition_log.csv" changes and refreshed the "Profile" page. The tester clicked on the "Previous Week" button twice and the "Next Week" button once.

Actual Results: Test passed. The actual results matched up with the expected results.

12. Test Name: profile-12

    Type: Functional, Dynamic, Manual.

    Initial State: User has completed test case "profile-11" but has now added an extra row of salsa having the same data as the previous salsa entry.

    Input: User clicks on the "Previous Week" button twice. Then, the user clicks on the "Next Week" button twice.

    Expected Results: The table is updated to show the following food items with their attached date and calorie information: Salsa and Salsa, Sour Cream, Poutine, Pickle, Cucumber, Cashew, and Sprite.

    How The Test Was Performed: After the tester completed the steps laid out in test case profile-11, the tester went into the directory "utrition\src\utrition\utrition-backend" and opened the "nutrition_log.csv" file. The tester copied and pasted the Salsa row once more at the bottom of the excel file. The tester saved the "nutrition_log.csv" changes and refreshed the "Profile" page. The tester clicked on the "Previous Week" button twice and the "Next Week" button twice.

    Actual Results: Test passed. The actual results matched up with the expected results.

13. Test Name: profile-13

    Type: Functional, Dynamic, Manual.

Initial State: User has no entries saved on their device and is viewing the “Profile” page.

Input: User clicks on “Profile” found in the taskbar.

Expected Results: The user is able to see the following headers with no nutritional information near them: Your most eaten food, Your total calories consumed for today, PAST MEALS, and My Caloric Intake.

How The Test Was Performed: The tester opened Utrition and clicked on “Profile” found in the taskbar. The tester clicked on “Profile” one more time.

Actual Results: Test passed. The actual results matched up with the expected results.

3.9 Taskbar

1. Test Name: upload-page-1

   Type: Functional, Dynamic, Manual.

   Initial State: User is on the “Home” page.

   Input: User clicks on the “Upload” button.

   Expected Results: The user views the “Upload” page.

   How The Test Was Performed: The tester opened Utrition and clicked on “Upload” found in the taskbar.

   Actual Results: Test passed. The actual results matched up with the expected results.

2. Test Name: profile-page-1

   Type: Functional, Dynamic, Manual.

   Initial State: User is on the “Home” page.

   Input: User clicks on the “Profile” button.

   Expected Results: The user views the “Profile” page.

   How The Test Was Performed: The tester opened Utrition and clicked on “Profile” found in the taskbar.

   Actual Results: Test passed. The actual results matched up with the expected results.

3. Test Name: home-page-1

   Type: Functional, Dynamic, Manual.

   Initial State: User is on the "Upload" page.

   Input: User clicks on the "Utrition" button.

   Expected Results: The user views the "Home" page.

   How The Test Was Performed: The tester opened Utrition and clicked on "Upload" found in the taskbar. The tester clicked on "Utrition" found in the taskbar.

   Actual Results: Test passed. The actual results matched up with the expected results.

4. Test Name: home-page-2

   Type: Functional, Dynamic, Manual.

   Initial State: User is on the "Profile" page.

   Input: User clicks on the "Utrition" button.

   Expected Results: The user views the "Home" page.

   How The Test Was Performed: The tester opened Utrition and clicked on "Profile" found in the taskbar. The tester clicked on "Utrition" found in the taskbar.

   Actual Results: Test passed. The actual results matched up with the expected results.

5. Test Name: profile-page-2

   Type: Functional, Dynamic, Manual.

   Initial State: User is on the "Upload" page.

   Input: User clicks on the "Profile" button.

   Expected Results: The user views the "Profile" page.

   How The Test Was Performed: The tester opened Utrition and clicked on "Upload" found in the taskbar. The tester clicked on "Profile" found in the taskbar.

   Actual Results: Test passed. The actual results matched up with the expected results.

6. Test Name: upload-page-2

   Type: Functional, Dynamic, Manual.

   Initial State: User is on the "Profile" page.

   Input: User clicks on the "Upload" button.

   Expected Results: The user views the "Upload" page.

   How The Test Was Performed: The tester opened Utrition and clicked on "Profile" found in the taskbar. The tester clicked on "Upload" found in the taskbar.

   Actual Results: Test passed. The actual results matched up with the expected results.

4 Nonfunctional Requirements Evaluation

4.1 Look and Feel

1. LF-1

   Type: Functional, Dynamic, and Manual.

   Initial State: A developer from Durum Wheat Semolina has launched the Utrition app.

   Input: The developer measures the distance between the displayed user interface components.

   Expected Results: The distance between user interface components must surpass 20 pixels.

   How The Test Was Performed: Each developer launches the Utrition app and opens the Pixel Measuring Tool. The developer will use the tool by viewing a page on Utrition, then clicking "Print Screen" on their keyboard. The developer then pastes the screenshot into the tool. The developer clicks the two spots he would like to measure distance from.

   Actual Results: Test passed. The actual results matched up with the expected results. All testing developers reported that all the visually distinct user interface components were at least 20 pixels away from one another. This ensures the visual components of the application will be visually pleasing, easy to read, and more intuitive to follow.

4.2 Usability

1. UH-1

   Type: Functional, Dynamic, and Manual.

   Initial State: The user is viewing a page of the user interface.

   Input: The user clicks on the "Menu" button located at the bottom of the user interface.

   Expected Results: The user will be brought to the main menu screen.

   How The Test Was Performed: A developer of Durum Wheat Semolina will access every page listed in "Initial State" and will attempt to reach the main menu in 2 or less clicks from the respective page.

   Actual Results: Test passed. The actual results matched up with the expected results. All testing developers were able to access the main menu in exactly 1 click from any page. All pages were tested; Main page, Upload page, Profile page.

2. UH-2

   Type: Functional, Dynamic, and Manual.

   Initial State: The user has previously entered 5 items in Utrition and is now viewing the main menu.

   Input: The user makes the request to view past nutritional data.

   Expected Results: The user views a list displaying all 5 food items with their respective details in: food name, calories, fat, sodium, proteins, carbohydrates, sugars, and date entered.

   How The Test Was Performed: A developer of Durum Wheat Semolina will open Utrition on their personal device and access the main menu. The developer will upload an image of different food items 5 times, with a minimum 1 minute time difference between each input. The uploaded images of the food items will be randomly selected by the developer in the utrition/test/testPhotos directory. The developer navigates to the past inputted foods area and views the list displaying the 5 different food items and their respective details.

   Actual Results: Test passed. The actual results matched up with the expected results. After submitting the 5 food items into the image

upload component, the developer navigated to the Profile page, where they reported that the expected nutritional data corresponding to the uploaded food items were displayed on the page.

3. UH-3

   Type: Functional, Dynamic, and Manual.

   Initial State: The user has previously entered 5 items in Utrition and has requested to view their past inputted food items.

   Input: The user navigates to the screen where they can view their past nutritional data in a chart format.

   Expected Results: The user views a chart displaying all 5 food items with their respective details in: food name, calories, proteins, carbohydrates, sugars, and date entered.

   How The Test Was Performed: A developer of Durum Wheat Semolina will open Utrition on their personal device and access the main menu. The developer will upload an image of different food items 5 times, with a minimum 1 minute time difference between each input. The uploaded images of the food items will be randomly selected by the developer in the utrition/test/testPhotos directory. The developer navigates to the past inputted foods area where they can view their statistics as a chart. The developer views a chart displaying 5 different food items and their respective details.

   Actual Results: Test passed. The actual results matched up with the expected results. After submitting the 5 food items into the image upload component, the developer navigated to the Profile page, where they reported that the expected nutritional data corresponding to the uploaded food items were displayed on the page within a chart.

4. UH-4

   Type: Functional, Dynamic, and Manual.

   Initial State: Utrition's main menu is opened on the user's personal device.

   Input: The user uploads a photo of food.

   Expected Results: The user views the food's nutritional information.

How The Test Was Performed: Each developer of Durum Wheat Semolina will load Utrition on to their personal device. Each developer will ask 2 people aged 14 or older to upload and retrieve the nutritional data of the food item "pineapple.jpg" found in the utrition/test/testPhotos directory. The developer will track the time it takes for each test. 90% of the panel must complete the task in under 2 minutes.

Actual Results: Test passed. The actual results matched up with the expected results. All 6 testing users were able complete the task in under 2 minutes, with an average time of 1.5 minutes.

5. UH-5

   Type: Functional, Dynamic, and Manual.

   Initial State: The user loads into Utrition's main menu.

   Input: The user views every page of the user interface.

   Expected Results: The user will see the respective symbol associated with every mention of calories, fat, sodium carbohydrates, sugar, and protein.

   How The Test Was Performed: On each screen, including the main menu, a developer on Durum Wheat Semolina will check if there is a symbol associated with every mention of calories, fat, sodium carbohydrates, sugar, and protein. The developer will open Utrition on their personal device. The developer will upload a .png image of food found in the utrition/test/testPhotos directory. The developer will view the nutritional information venture through each page to confirm the correct symbols appear.

   Actual Results: Test passed. The actual results matched up with the expected results. The testing developer reported that every expected nutritional data type had a corresponding label that was easily identifiable.

6. UH-6

   Type: Functional, Dynamic, and Manual.

   Initial State: The user has previously entered 2 items in Utrition and is now viewing the main menu.

   Input: The user views every page of the user interface.

Expected Results: The user does not see any backend calculations.

How The Test Was Performed: On each screen, a developer on Durum Wheat Semolina will check if any backend calculations are displayed to the user. The developer will open Utrition on their personal device and access the main menu. The developer will upload an image of two different food items, with a minimum of 1 minute between the inputs. The uploaded images of the food items will be randomly selected by the developer in the utrition/test/testPhotos directory. The developer will view all pages to ensure no backend calculations are visible.

Actual Results: Test passed. The actual results matched up with the expected results. The testing developer reported that no backend calculations were visible on the user interface during the image upload user experience process.

7. UH-7

   Type: Functional, Dynamic, and Manual.

   Initial State: The user opens Utrition with a stable internet connection.

   Input: The user views every page of the user interface.

   Expected Results:

   - Utrition is available to be used by users at all times.
   - The user does not hear Utrition generate any sound.

   How The Test Was Performed: On each screen, a developer on Durum Wheat Semolina will check if a sound plays. The developer will open Utrition on their personal device. The developer will upload an image of food. The uploaded image of the food item will be randomly selected by the developer in the utrition/test/testPhotos directory. The developer will view the nutritional information. The developer then will view the past inputs and the nutritional information chart.

   Actual Results: Test passed. The actual results matched up with the expected results. The testing developer reported a smooth user experience during the image upload process with a stable internet connection. The testing developer also noted that the application did not generate any audio, with speakers on maximum volume.

8. UH-8

   Type: Static and Manual.

   Initial State: A developer on Durum Wheat Semolina views Utrition's GitHub page.

   Input: The developer checks every directory for an audio file.

   Expected Results: The developer removes all audio files found in Utrition's GitHub.

   How The Test Was Performed: The developer loads into Utrition's GitHub and views the files in every directory. For each directory the developer will check for the following file types:

   - MP3
   - AAC
   - Ogg Vorbis
   - FLAC
   - ALAC
   - WAV
   - AIFF
   - DSD
   - PCM

   Actual Results: Test passed. The actual results matched up with the expected results. After inspecting Utrition's public repository, the testing developer reported no outstanding files of the above file types.

Apart from the more defined tests discussed above, additional usability testing was conducted on several participants to obtain feedback regarding Utrition's overall quality and user experience. A semi-structured interview was conducted on a total of 5 individuals of different genders and ethnicities to provide a wide range of opinions and suggestions. The interview process consisted of a brief explanation of the application, sample tasks for the participant to perform, and a series of questions. The raw notes from each interview can be found in Appendix A.

Many useful insights were gathered from the interview results. When participants were asked to input multiple food entries, most opted for text or voice upload options. The majority of participants did not read the instructions but completed the task quickly and with ease. Common likes brought up by participants included the colour scheme and layout of the application, the ease of use regarding food entry upload, and the depth of information found on the profile page.

In terms of common dislikes, many participants wanted to see units beside the nutritional data they were receiving, more error messages when something goes wrong, and to have meals grouped together based on input time when viewing the profile page. Additional features that individuals wish to see in feature revisions include BMI inputs from the user, goal setting, and a graph displaying past food input information. Greater explanation on implementation changes is discussed in section 7.2.

4.3 Performance

1. PT-1

   Type: Functional, Dynamic, and Manual.

   Initial State: The user has previously entered 10 items in Utrition.

   Input: The user views every page of the user interface.

   Expected Results: The user will be able to access every page of the user interface in 2 seconds or less.

   How The Test Was Performed: A developer on Durum Wheat Semolina will measure the time it takes for each new screen to load using a stopwatch. The developer will open Utrition on their personal device. The developer will upload an image of different food items 10 times, with a minimum 1 minute time difference between the inputs. The uploaded images of the food items will be randomly selected by the developer in the utrition/test/testPhotos directory. The developer will navigate to each screen, timing each transition.

   Actual Results: Test passed. The actual results matched up with the expected results. The testing developer reported that the navigation and loading time for every page took less than 1 second.

2. PT-2

Type: Functional, Dynamic, and Manual.

Initial State: The user loads into Utrition.

Input: The user uploads 3 images simultaneously.

Expected Results:

- The user is able to view the identification for all 3 food items in 10 or less seconds.
- The user is able to view the nutritional information for all 3 food items in 5 or less seconds.

How The Test Was Performed: A developer on Durum Wheat Semolina will open Utrition. The developer will upload three images of food found in the utrition/test/testPhotos directory. The developer clicks to view the foods' nutritional information. The developer measures the amount of time it takes for Utrition to notify the user of the name of the identified food items. The developer measures the amount of time it takes for the system to change from the food identification interface to the nutritional information interface.

Actual Results: Test failed. The developer reported that multiple images were not able to be uploaded simultaneously and had to be uploaded consecutively. After uploading an image, the developer reported that it took less than 5 seconds for the food to be identified and for the food's nutritional data to be displayed.

3. PT-3

   Type: Functional, Dynamic, Automatic

   Initial State: Nutritional data of a food item is contained in the system.

   Input: The user requests to display the nutritional information of a food item.

   Expected Results: The system will generate the HTML body to be displayed on the user interface.

   How The Test Was Performed: The test will measure if this interaction was completed in 5 seconds or less.

   Actual Results: Test passed. The actual results matched up with the expected results. The testing developer requested for the nutrirtional

information of 15 different items, and reported that the nutritional data of a food item was fetched and displayed on the user interface in less than 5 seconds, with an average time of 0.6 seconds.

4. PT-4

   Type: Functional, Dynamic, and Manual.

   Initial State: The user loads into Utrition.

   Input: The user uploads an image of food.

   Expected Results: The food is correctly identified, and the correlated nutritional facts are correct.

   How The Test Was Performed: A developer on Durum Wheat Semolina will open Utrition and upload the images of the following foods separately (images found in utrition/test/testPhotos directory):

   - Uncooked Pork
   - Corn
   - Lettuce
   - Beef
   - Penne Pasta
   - Rice
   - Milk
   - Butter
   - Wheat Bread
   - Orange Juice

   The developer will check the accuracy of each identified food item and their respective nutritional facts. 70% of the images must be identified correctly. 90% of the nutrition facts for the correctly identified foods will be given correctly.

   Actual Results: Test failed. The testing developer reported that none of the above food items were correctly identified. This result was expected, as the image classification model training data sample size was constrained to a small number of food items.

5. PT-5

   Type: Functional, Dynamic, and Automatic.

   Initial State: System contains a set of images; all with foods, and of extension .jpg, .png, or .jpeg.

   Input: System prompted to process the image, for each image within the provided set (images found in utrition/test/testPhotos directory).

   Expected Results: For each image, the system should identify the food present in an image and return the name of the food item as a string.

   How The Test Was Performed: The test will check if among all the provided images, more than 90% of the images have been assessed correctly.

   Actual Results: Test passed. The actual results matched up with the expected results. From uploading the images in the provided set 15 times, the developer reported that the system processed the food item correctly 14 times. 93 % of the images were assessed correctly.

6. PT-6

   Type: Functional, Dynamic, and Automatic.

   Initial State: System contains a set of images; some with and others without foods, and of extension .jpg, .png, or .jpeg.

   Input: System prompted to process image, for each image within the provided set (images found in utrition/test/testPhotos directory).

   Expected Results: The system should identify the food present in an image, and return the name of the food item as a string, and prompt the user if no food is detected.

   How The Test Was Performed: The test will check if among all the provided images, more than 70% of the images have been assessed correctly.

   Actual Results: Test failed. The testing developer reported that 60% of the images had been assessed correctly, with the failing tests being with images of no food items.

7. PT-7

   Type: Functional, Dynamic, and Manual.

Initial State: The user loads into Utrition.

Input: The user uploads 4 photos of different food items.

Expected Results: The user is notified that they are not able to proceed with viewing the identified foods or their nutritional information.

How The Test Was Performed: A developer on Durum Wheat Semolina will open Utrition and attempt to upload four images of random food items found in the utrition/test/testPhotos directory. The developer is notified with an error informing them that only 3 images can be uploaded at once.

Actual Results: Test failed. The testing developer reported that no error message was displayed, and that the application only accepts multiple images to be uploaded consecutively.

8. PT-8

   Type: Static and Manual.

   Initial State: A developer on Durum Wheat Semolina opens their internet browser.

   Input: The developer is able to access Utrition's GitHub and download the repository on to their personal device.

   Expected Results: Utrition is able to be downloaded.

   How The Test Was Performed: The developer will access Utrition's GitHub on their personal device's web browser. The developer will click on "Code", and then "Download ZIP". The developer will verify that Utrition has been downloaded on to their device.

   Actual Results: Test passed. The actual results matched up with the expected results. All 6 testing developers were able to download and access Utrition's public GitHub repository.

9. PT-9

   Type: Functional, Dynamic, and Manual.

   Initial State: The user loads into Utrition.

   Input: The user uploads a very large photo (more than 30MB) for Utrition to identify.

Expected Results: The user is notified that they are not able to proceed with viewing the identified foods or their nutritional information.

How The Test Was Performed: A developer on Durum Wheat Semolina will open Utrition and upload "art.png" found in the utrition/test/testPhotos directory. The developer is notified that they cannot proceed with viewing the nutritional information unless they decrease the uploaded image's file size.

Actual Results: Test failed. The testing developer reported that no error message was displayed on the user interface when a large image file was uploaded.

10. PT-10

    Type: Functional, Dynamic, and Manual.

    Initial State: The user loads into Utrition.

    Input: The user uploads an unintelligible image for Utrition to identify.

    Expected Results: The user is notified that Utrition is not confident in the system's identification. The system will suggest uploading another image.

    How The Test Was Performed: A developer on Durum Wheat Semolina will open Utrition and upload "desk.jpg" found in the utrition/test/testPhotos directory. The developer is notified that the machine learning algorithm is not confident in the identification of the food, and that they should upload another image.

    Actual Results: Test failed. The testing developer reported that no message was displayed on the user interface describing the low image classification confidence.

11. PT-11

    Type: Functional, Dynamic, and Automatic.

    Initial State: The system is identifying food objects within the user's uploaded image(s).

    Input: Uploaded image.

    Results: An error message is displayed prompting the user of a failed identification.

How The Test Was Performed: The test will check if an error message element has been rendered.

Actual Results: Test failed. The testing developer reported that no error message was displayed on the user interface following a failed identification.

12. PT-12

    Type: Functional, Dynamic, and Manual.

    Initial State: The user loads into Utrition without connecting to the internet.

    Input: The user uploads an image for Utrition to identify.

    Expected Results: The user is notified that the system is unable to retrieve nutritional information because they are not connected to the internet.

    How The Test Was Performed: A developer on Durum Wheat Semolina will open Utrition upload a random working image found in the utrition/test/testPhotos directory. The developer is notified that they are not connected to the internet after 20 seconds.

    Actual Results: Test failed. The testing developer reported that no error message was displayed on the user interface when there was no internet connection.

13. PT-13

    Type: Functional, Dynamic, and Manual.

    Initial State: The user loads into Utrition.

    Input: The user uploads an image for Utrition to identify.

    Expected Results: The user is notified that the system has identified the food item but could not find the food's nutritional data.

    How The Test Was Performed: A developer on Durum Wheat Semolina will open Utrition and upload "dragonfruit.jpg" found in the utrition/test/testPhotos directory. The developer is notified that they have uploaded an image of a "Dragon Fruit", but the nutritional data cannot be found.

    Actual Results: Test failed. The testing developer reported that the image of a "Dragon Fruit" was not correctly identified.

14. PT-14

    Type: Functional, Dynamic, and Manual.

    Initial State: The user opens Utrition without previously inputting any food items.

    Input: The user attempts to view their past inputs' nutritional information as text and as a chart.

    Expected Results: The user views an empty list with column entries: food name, calories, proteins, carbohydrates, sugars, and date entered. The user will see an empty canvas for the chart, and text detailing that there are no previous nutritional logs.

    How The Test Was Performed: A developer of Durum Wheat Semolina will open Utrition on their personal device and view nutritional data from past inputs. A message displays "There are no previous food items recorded" and an empty chart.

    Actual Results: Test failed. The testing developer reported that an empty chart was displayed, but the message "There are no previous food items recorded" was not displayed on the user interface.

15. PT-15

    Type: Functional, Dynamic, and Manual.

    Initial State: A user loads into Utrition.

    Input: The user uploads a PDF document as an image.

    Expected Results: The user is notified that they are not able to proceed with viewing the identified foods or their nutritional information.

    How The Test Was Performed: A developer on Durum Wheat Semolina will open Utrition upload the image "waffle.pdf" to Utrition from the utrition/test/testPhotos directory. The developer is notified that they cannot upload the file since Utrition does not support ".pdf" file types.

    Actual Results: Test failed. The testing developer reported that no error message was visible on the user interface.

4.4 Operational and Environmental

1. OE-1

Type: Static and Manual.

Initial State: A user aged 14 or older has read Utrition's "README.md" located at the bottom of the GitHub page.

Input: The user installs Utrition on their personal device.

Expected Results: The intended demographic is able to install Utrition on their computer.

How The Test Was Performed: Each developer on Durum Wheat Semolina will test 5 individuals aged 14 and above to install Utrition. The test passes if 95% of users can install Utrition.

Actual Results: Test passed. The actual results matched up with the expected results. All 5 testing users were able to install Utrition from the public GitHub repository.

2. OE-2

   Type: Static and Manual.

   Initial State: A user visits Utrition's GitHub page.

   Input: The user checks that the latest change to Utrition's GitHub is not after April 30th, 2023.

   Expected Results: Utrition will not have any updates after the final release.

   How The Test Was Performed: A developer on Durum Wheat Semolina will visit Utrition's GitHub page bi-yearly to ensure no updates are made to Utrition's GitHub.

   Actual Results: Test passed. The actual results matched up with the expected results. The last modified date of the Utrition GitHub repository is before April 30th, 2023.

4.5 Maintainability and Support

1. MS-1

   Type: Functional, Dynamic, and Manual.

   Initial State: The developer is in the utrition/src/database directory.

   Input: The developer switches the database file found in the database directory for another database.

Expected Results: Backend developers are able to upload new data to Utrition's database.

How The Test Was Performed: The developer travels to the utrition/src/database directory on their device. The developer replaces the database found in the directory with "testdatabase.jpg", which can be found in the utrition/test/testPhotos directory. The developer launches Utrition on their device to see that the system still runs.

Actual Results: Test passed. The actual results matched up with the expected results. The testing developer was able to replace the previous database file with a new database file that contained different entries. Both database files were of the same schema.

2. MS-2

   Type: Functional, Dynamic, and Manual.

   Initial State: The user has installed Utrition on to their Windows, macOS, or Linux device.

   Input: The user is able to upload an image of food and view all pages of the user interface.

   Expected Results: All of Utrition's functionality can be accessed by Windows, macOS, and Linux devices.

   How The Test Was Performed: Developers of Durum Wheat Semolina will install Utrition on one of each Windows, macOS, and Linux device. The developer will open Utrition on the device and upload a random image of a food item found in the utrition/test/testPhotos directory. The developer sees their foods' nutritional information. The developer can view past inputs' nutritional data and chart.

   Actual Results: Test passed. The actual results matched up with the expected results. 3 users reported a smooth and expected user experience during the image upload process, with each user accessing the system through a different operating system (Windows, macOS, Linux).

4.6 Security

1. ST-1

Type: Functional, Dynamic, and Manual.

Initial State: The user is viewing the nutritional facts for a food item they uploaded.

Input: After idling for 3 minutes, the user closes Utrition and reopens it.

Expected Results: The user will open Utrition and will view the nutritional facts of the image in the past inputs section.

How The Test Was Performed: A developer on Durum Wheat Semolina will open Utrition and will upload "waffle.jpg" found in the utrition/test/testPhotos directory. The developer views the nutritional facts of a waffle, and then idles for 3 minutes. After the elapsed time, the developer closes their instance of Utrition and opens it again. The developer can view the nutritional facts of the waffle in the past inputs section.

Actual Results: Test passed. The actual results matched up with the expected results. The testing developer was able to consistently view the nutritional facts of the waffle after the elapsed idle time, and after restarting the application.

2. ST-2

   Type: Static and Manual.

   Initial State: A Durum Wheat Semolina developer visits Utrition's GitHub page.

   Input: The developer checks the userdata directory for any files.

   Expected Results: The userdata folder will be empty on GitHub, so Utrition will only contain personalized data once downloaded on a user's device.

   How The Test Was Performed: The developer loads into Utrition's GitHub and goes into the utrition/src/userdata directory. The developer ensures that no userdata is preloaded into Utrition's GitHub.

   Actual Results: Test passed. The actual results matched up with the expected results. The testing developer reported that there was no user data after downloading a fresh version of the system from Utrition's GitHub.

4.7 Cultural

1. CT-1

   Type: Functional, Dynamic, and Manual.

   Initial State: Utrition's main menu is being displayed to a user.

   Input: The user is told by a Durum Wheat Semolina developer to find culturally insensitive material.

   Expected Results: The user will be questioned after 5 minutes of searching if they have found any culturally insensitive material while using Utrition.

   How The Test Was Performed: Each developer will approach 5 peers with Utrition's main menu open on the developer's device. The developer will give full control over to their peers and will tell their peers to search for culturally insensitive material. After 5 minutes are up, the developer will ask their peers if they have found any culturally insensitive material. This test passes if 95% of users could not find any culturally insensitive material.

   Actual Results: Test passed. The actual results matched up with the expected results. A survey that was conducted after users were able to explore the application showed that 100% of users did not find any culturally insensitive material on the system.

4.8 Legal

1. LT-1

   Type: Static and Manual.

   Initial State: A developer of Durum Wheat Semolina views Utrition's GitHub repository.

   Input: The developer inspects all code to ensure Utrition is following Canada's Anti-Spam Legislation Requirements for Installing Computer Programs, Canada's Guide for Publishing Open Source Code, Google's HTML/CSS Style Guide, and Google's guide for material design.

   Expected Results: Utrition is in compliance with all the standards laid out by the SRS document.

How The Test Was Performed: The developer goes into the utrition/src directory, and then opens the link to one of the following guides:

- Canada's Anti-Spam Legislation Requirements for Installing Computer Programs
- Guide for Publishing Open Source Code
- Google JavaScript Style Guide
- Google HTML/CSS Style Guide
- Material Design Guide

The developer will go into each module found in the utrition/src directory and will inspect the code to ensure that each rule outlined in Canada's or Google's guide is strictly adhered to.

Actual Results: Test failed. The testing developer reported that there were inconsistencies in the code, regarding the standards outlined in the style guides.

2. LT-2

   Type: Functional, Dynamic, and Manual.

   Initial State: A Durum Wheat Semolina developer has completed a module for Utrition and has Pylint installed on their device.

   Input: The developer runs Pylint on their code.

   Expected Results: Pylint will notify the developer if there needs to be changes to the code to ensure Utrition adheres to the PEP8 Python coding conventions.

   How The Test Was Performed: The developer will open their command prompt and travel to the directory with the finished code (utrition/src) by using "cd". The developer runs Pylint on the finished code by typing "pylint finishedmodule.py" in their command prompt. The name "finishedmodule.py" is a placeholder for any future modules that Utrition will implement in the future. The command prompt will notify the developer to make any necessary changes.

   Actual Results: Test failed. The testing developer reported that there were inconsistencies in the code, regarding the standards outlined in the PEP8 conventions.

5 Comparison to Existing Implementation

N/A

6 Unit Testing

For this project, the majority of testing for functional and nonfunctional requirements was conducted through manual testing. The developers conducted manual tests to measure how the system reacted to possible scenarios and interactions. The automated unit testing of specific core functions in the backend of the application can be seen in the Functional Requirements Evaluation of Nutritional Facts Display (3.4), and Call and Fetch API Response (3.5).

7 Changes Due to Testing

7.1 Changes Due to Functional Requirements Testing

Due to the failed test cases during functional requirements testing, Utrition's implementation will be improved in the following ways:

- The "Enter" key on a user's keyboard will submit the user's response rather than create a new line during text upload.
- Switching to a new upload section (Text Upload, Voice Upload, or Image Upload) will remove all inputs from the previous upload page.
- The "Voice Upload" section will not continue listening after the section has been clicked off of.
- The "Reset" button will not push the user to the "Text Upload" section when they are not on it.
- "Your most eaten food" will be changed to "Your most logged food".

7.2 Changes Due to Usability Testing

Due to the feedback collected during usability testing, Utrition's implementation will undergo changes to provide the best user experience possible.

Taking into consideration the time constraints and the most commonly discussed improvements, these changes will be implemented:

- Add "Get Started" button to home page to take user to input a food item
- Add units to nutritional data output
- Create error messages for cases where inputted food item is not identifiable
- Group entries in food history table based on entries submitted together
- Add graph to display past food entry data
- Optional height and weight user inputs to calculate BMI and recommended calories per day

8 Automated Testing

The following tests are classified as automated tests:

- one-upload-auto-1
- one-upload-auto-2
- image-identification-1
- image-identification-2
- image-identification-3
- api-1
- log-data-read-data-1
- past-nutrition-text-read-data-2
- current-nutrition-1
- display-elements-1
- correct-path-1
- axios-call-1

9 Trace to Requirements

Table 1: Traceability between Tests and Functional Requirements

Requirement #	**Description**	**Test ID(s)**
FR1	The user must have the ability to upload a digital image of a standard image type to the system.	one-upload-1 one-upload-2 one-upload-auto-1 one-upload-auto-2
FR2	The system will be able to identify the type of food that is captured in an image.	one-upload-1 one-upload-2 image-identification-1 image-identification-2 image-identification-3 current-nutrition-1
FR3	The system will be able to make an API call to an external database of nutrition facts for a variety of foods, including their macro-nutrients, micro-nutrients, and caloric details.	api-1 current-nutrition-1 axios-call-1

Requirement #	Description	Test ID(s)
FR4	The system will be able to retrieve the nutrition facts for a specific food.	one-upload-1 api-1 log-data-read-data-1 current-nutrition-1 text-upload-1 text-upload-2 text-upload-6 text-upload-7 text-upload-8 multi-text-upload-1 multi-text-upload-2 multi-text-upload-3 voice-upload-1 voice-upload-2 voice-upload-3 voice-upload-9 voice-upload-10 voice-upload-11 voice-upload-12 multi-voice-upload-1 multi-voice-upload-2 multi-voice-upload-3 profile-4 profile-5

Requirement #	**Description**	**Test ID(s)**
FR5	The system will log the nutritional data of a food.	api-1 log-data-read-data-1 current-nutrition-1 profile-1 profile-2 profile-3 profile-4 profile-5
FR6	The user will be able to access the nutritional data of previously logged foods.	log-data-read-data-1 past-nutrition-read-data-2 profile-1 profile-2 profile-3 profile-4 profile-5 profile-6 profile-7 profile-8 profile-9 profile-10 profile-11 profile-12 profile-13

Requirement #	Description	Test ID(s)
FR7	The system will display the nutritional information of a food to the user.	current-nutrition-1 past-nutrition-read-data-2 text-upload-1 text-upload-2 text-upload-6 text-upload-7 text-upload-8 multi-text-upload-1 multi-text-upload-2 multi-text-upload-3 voice-upload-1 voice-upload-2 voice-upload-3 voice-upload-9 voice-upload-10 voice-upload-11 voice-upload-12 multi-voice-upload-1 multi-voice-upload-2 multi-voice-upload-3 display-elements-1 axios-call-1
FR8	The user must have the ability to upload the text of a food input to the system.	text-upload-1 text-upload-2 text-upload-3 text-upload-4 text-upload-5 text-upload-6 text-upload-7 text-upload-8

Requirement #	**Description**	**Test ID(s)**
FR9	The user must have the ability to upload multiple texts of food inputs to the system.	multi-text-upload-1 multi-text-upload-2 multi-text-upload-3 profile-5
FR10	The user must have the ability to upload a food input by speaking to the system.	voice-upload-1 voice-upload-2 voice-upload-7 voice-upload-8 voice-upload-9 voice-upload-10 voice-upload-11 voice-upload-12 profile-4
FR11	The user must have the ability to upload multiple food inputs by speaking to the system.	voice-upload-3 multi-voice-upload-1 multi-voice-upload-2 multi-voice-upload-3
FR12	The system will be able to convert vocal input into text.	voice-upload-1 voice-upload-2 voice-upload-3 voice-upload-8 voice-upload-9 voice-upload-10 voice-upload-11 voice-upload-12 multi-voice-upload-1 multi-voice-upload-2 multi-voice-upload-3 profile-4

Requirement #	Description	Test ID(s)
FR13	The user must have the ability to restart their voice upload to the system.	voice-upload-4 voice-upload-5 voice-upload-6
FR14	The system will be able to identify the type of food(s) that is captured in a complex sentence.	text-upload-2 text-upload-7 text-upload-8 multi-text-upload-3 voice-upload-1 voice-upload-2 voice-upload-3 voice-upload-10 voice-upload-11 multi-voice-upload-1 multi-voice-upload-2 multi-voice-upload-3 profile-4
FR15	The user must have the ability to select between uploading food(s) through image, text, or voice.	voice-upload-12 correct-path-1
FR16	The system will display the user's most logged food.	profile-1 profile-2 profile-3 profile-4 profile-5 profile-13 display-elements-1 axios-call-1

Requirement #	Description	Test ID(s)
FR17	The system will display the user's total calories consumed for the day.	profile-1 profile-2 profile-3 profile-4 profile-5 profile-13 display-elements-1 axios-call-1
FR18	The system will display a summary of the foods consumed and their total calories for each of the past 7 days.	profile-1 profile-2 profile-3 profile-4 profile-5 profile-9 profile-10 profile-11 profile-12 profile-13 display-elements-1 axios-call-1
FR19	The user will be able to access the summary of the foods consumed and their total calories for all days with a food inputted.	profile-1 profile-2 profile-3 profile-4 profile-5 profile-9 profile-10 profile-11 profile-12 profile-13

Requirement #	**Description**	**Test ID(s)**
FR20	The user will be able to access the Home page, the Upload page, and the Profile page on every page.	upload-page-1 profile-page-1 home-page-1 home-page-2 profile-page-2 upload-page-2 correct-path-1

Table 2: Traceability between Tests and NonFunctional Requirements

Requirement #	**Description of Fit Criterion**	**Test ID(s)**
LF1	The distance between user interface components must exceed 20 pixels.	LF-1
UH1	The user will be able to navigate to the main menu from any page in 2 clicks.	UH-1
UH2	The user must be able to view every past inputted food item.	UH-2, UH-3
UH3	90% of a test panel of the target demographic should be able to upload their food and retrieve the corresponding nutritional facts in under 2 minutes.	UH-4
UH4	Every clickable object will have a symbol associated with it. Calories, fat, sodium, carbohydrates, sugar, and protein will have a symbol associated next to it.	UH-5
UH5	All backend calculations, such as identifying the uploaded food and retrieving the nutritional information, will not be displayed to the user.	UH-6
UH6	There will be no sound used by the Utrition web application.	UH-7, UH-8
PR1	The time between the user's click and displaying the new UI will not exceed 2 seconds.	PT-1
PR2	After uploading a picture, Utrition will always display the identified food item within 10 seconds in a test with multiple images.	PT-2

Requirement #	Description of Fit Criterion	Test ID(s)
PR3	After identifying the correct food item, Utrition will always display the nutritional facts within 5 seconds in a test with multiple images.	PT-2, PT-3
PR4	Error message should prompt user upon upload of an abnormal image.	PT-15
PR5	Error message should prompt user upon upload of an image larger than 30MB.	PT-9
PR6	Error message should prompt user upon upload of more than 3 images.	PT-7
PR7	Error message should prompt user upon failure to identify any food items in the uploaded image.	PT-10, PT-11
PR8	Error message should prompt user to try again upon failure to request nutritional info regarding food items in the uploaded image.	PT-12
PR9	Error message should notify user that the nutritional information pertaining to their food item does not exist.	PT-13
PR10	Error message should notify user upon failure to access user's past nutritional info.	PT-14
PR11	Error message should notify user if user's past nutritional info has been deleted.	PT-14
PR12	Error message should prompt user if user's past nutritional info cannot be depicted on a chart.	PT-14
PR13	Request will time out after 20 seconds, and user will be prompted that their device is not connected to the internet.	PT-12

Requirement #	Description of Fit Criterion	Test ID(s)
PR14	In a test with multiple images, Utrition will identify and display the correct food with an average accuracy of 70%.	PT-4, PT-6
PR15	In a test with multiple images, Utrition will display the correct nutritional information for an identified food 90% of the time.	PT-4, PT-5
PR16	Since Utrition is a local web app, the user will always be allowed to use Utrition unless their device or internet connection is down.	PT-4
PR17	The system will not allow more than 3 photos to be uploaded at once.	PT-2, PT-7
PR18	Utrition is a local web app and does not depend on a server.	PT-8
PR19	Utrition will be available to use on GitHub for an indefinite amount of time.	PT-8
OE1	Users will be able to find and pull the repository on to their personal device.	PT-8
OE2	95% of a test panel with the intended demographic will be able to install Utrition after reading the installation guide.	OE-1
OE3	Utrition's GitHub repository will receive no updates after final implementation.	OE-2
MS1	Any backend developer is able to successfully upload new data to Utrition's machine learning model at any time during development.	MS-1

Requirement #	Description of Fit Criterion	Test ID(s)
MS2	All users will be able to view the support manuals on how to install and use Utrition.	OE-1
MS3	100% of the tested Windows, macOS, and Linux devices should be able to use all functionality provided for Utrition.	MS-2
SR1	Anyone will be able to pull the open-source project from GitHub.	PT-8, OE-1
SR2	Utrition will save user data every 3 minutes throughout the application runtime.	ST-1
SR3	Utrition will not contain personalized data from one user on another user's system.	ST-2
CP1	95% of the test panel agrees that Utrition does not provide any culturally insensitive content to the user.	CT-1
LR1	No malicious software will be present in Utrition's GitHub repository.	LT-1
LR2	Durum Wheat Semolina will not commit infractions when following Canada's guide.	LT-1
LR3	Pylint will be used throughout development to ensure every pull request contains properly formatted code.	LT-2
LR4	Before every pull request, code will be reviewed to ensure no violations of the standards occur.	LT-1
LR5	Utrition's user interface will be reviewed to ensure all fonts and colours adhere to the style guide.	LT-1

10 Trace to Modules

Table 3: Traceability between Modules and Functional Tests

Module #	**Name**	**Test ID(s)**
M1	Application Path Module	upload-page-1 upload-page-2 profile-page-1 profile-page-2 home-page-1 home-page-2 correct-path-1
M2	Home Page Module	display-elements-1
M3	Profile Page Module	profile-1 profile-2 profile-3 profile-4 profile-5 profile-6 profile-7 profile-8 profile-9 profile-10 profile-11 profile-12 profile-13
M4	Nutrition Log Module	profile-6 profile-7 profile-8 profile-9 profile-10 profile-11 profile-12 profile-13 log-data-read-data-1 past-nutrition-text-read-data-2
M5	Upload Page Module	one-upload-1
M6	Upload Container Module	one-upload-1 voice-upload-4 voice-upload-6
M7	Image Upload Module	one-upload-1 one-upload-2 axios-call-1

Module #	**Name**	**Test ID(s)**
M8	Text Upload Module	text-upload-1 text-upload-2 text-upload-3 text-upload-4 text-upload-5 text-upload-6 text-upload-7 text-upload-8 multi-text-upload-1 multi-text-upload-2 multi-text-upload-3 multi-text-upload-4 axios-call-1
M9	Voice Upload Module	voice-upload-1 voice-upload-2 voice-upload-3 voice-upload-4 voice-upload-5 voice-upload-6 voice-upload-7 voice-upload-8 voice-upload-9 voice-upload-10 voice-upload-11 voice-upload-12 multi-voice-upload-1 multi-voice-upload-2 multi-voice-upload-3 axios-call-1
M10	Navigation Bar Module	upload-page-1 upload-page-2 profile-page-1 profile-page-2 home-page-1 home-page-2
M11	Input Pre-Processing Module	one-upload-1
M12	Training Dataset Module	one-upload-1
M13	Image Classification Module	one-upload-auto-1 image-identification-1 image-identification-2 image-identification-3

Module #	Name	Test ID(s)
M14	Nutritional Data Retriever Module	current-nutrition-1 text-upload-3 text-upload-4 text-upload-5 text-upload-6 text-upload-7 text-upload-8 multi-text-upload-1 multi-text-upload-3 multi-text-upload-4 voice-upload-2 voice-upload-7 voice-upload-8 voice-upload-9 voice-upload-10 voice-upload-12 multi-voice-upload-1 multi-voice-upload-3 api-1
M15	Profile Data Calculation Module	profile-1 profile-2 profile-3 profile-4 profile-5

11 Code Coverage Metrics

The automated tests have the following code coverage. The remaining code that was not covered by these tests was covered through manual testing.

Filename	Statements	Missed statements	Coverage
ML/DataHandler.py	71	6	92%
ML/Interface.py	16	0	100%
ML/MainML.py	42	0	100%
home/nutritionalDataFetcher.py	11	0	100%
home/profileData.py	103	71	31%

Appendix A

Informal User Testing Notes

User #1, 22 year-old, male, interviewed on 03/01/2023:

Expectations:

- Do actions: search for food, see baseline nutritional stats, and more in depth stats
- Scan a variety of foods
- Track calories, see projection
- See if I'm over/under a goal
- Set goals
- Personalization via profile, personalized results (save meal prep, meal sets)
- Make recommendations on diet, substitutions on diet

Experience:

- Wants feedback on which type of upload is selected
- Inputting natural language into text/voice upload, with complete sentences

Likes:

- Instant feedback, fast

- Minimal steps to upload
- Likes how it confirms what foods got uploaded
- How it shows the general nutrition facts that most people are looking for
- Likes the summarization and history of past foods

Improvements:

- Wants to show the number of food/quantity eaten when food is submitted and nutritional total shown
- Wants to be able to show expanded form of breakdown
- Add units to the nutrition data shown
- Bug when you enter food and separate them with "enter" new line character
- Doesn't like how half the upload is "useless", wants the functionality to be most of the page; focus of page is only on half of the page (wants it more central in line of page)
- "How it works" text does not make sense, taking up space
- Want profile/upload buttons on main page instead of heading nav bar since there are only 2 options
- Paragraph on first page too long
- Main page: "Make a choice!" text does not make sense
- Want a sticky nav bar
- Text input

  Want to be able to press "enter" button to submit text
- Profile page

  Want more personalization to the profile page (goals ie. Lose/gain weight)

Add link to external source to see estimated calories for a person of height/weight

Hard to read nutritional facts on profile page under "past meals", also in caloric intake table, maybe break it down and format it better and make it more readable; it is hard to scan and find what you want

"look at next 4 entries" button: expended it to append 4 entries to bottom of list

Additional features:

- Show a weekly graph to show progression

  Good way to show progression visually, summarizes information

User #2, 22 year-old, male, interviewed on 02/28/2023:

Expectations:

- Give it name of meal, or meal components and be able to show the nutritional information about the meal
- Look at all the food eaten in a period of time and return nutritional value
- How will the program know what you are trying to describe? some feature to detect food
- Logging and Viewing → At a high level

Experience:

- Reads instructions on homepage
- Bouncing back from homepage to upload page to read instructions
- Instructions don't exist in the same place as where the interaction occurs
- Enters the food individually, and specifies some amount for the yogurt
- Submits and reads information about the food

- Forgets to stop the recording before submitting, application continues to record after submission

Likes:

- Good layout
- Good separate pages

Improvements:

- Wants units for nutrients

  Not consistent between nutrients, so should specify

- Should output the quantity inputted by the user to ensure the quantity was taken into consideration when calculating nutritional information
- Automatically stop recording?
- Uploaded image is only allowed to exist in specific folder, doesn't give a relative path back to the backend
- Meal components are listed separately even if they are one meal; need to communicate if items were eaten at the same time
- Instead of scrolling by 4 entries, can instead scroll by date/meal rather than an arbitrary 4 entries
- Reset brings the user back to the text upload (should go back to voice upload)

Additional features:

- Get started button?!
- Graph to measure eating trends over time
- Profile page information could be better displayed
- Instead of instantly displaying information on each meal item, have brief information, and can expand for more information

User #3, 22 year-old, female, interviewed on 03/02/2023:

Expectations:

- MyFitnessPal
- Nutritional information is returned based on brand of food
- Macronutrients, more than just calories
- Separated into meals or snacks
- Input user information; weight, track weight
- Estimate how much calories needed to reach some "goal weight" (also given by user)
- Tracking progress on some MACRONUTRIENT goal per day e.g. "daily protein goal"
- Give advice on types of food that you should be eating

Experience:

- Clicked upload page
- Didn't read instructions
- Wasn't completely sure if multiple inputs were allowed; referred to example to clarify
- Used Voice Upload
- Didn't press stop recording before submitting

  Kept recording
- Thai basil ground beef was recorded as "thai basil, ground beef"
- Nutritional information probably not accurate, portion size not communicated
- Didn't occur to specify exact portions when speaking to the applications

Likes:

- Showed calories right away
- Ease of input
- Aesthetic

Improvements:

- Portions are assumed; not exactly accurate

  Would prefer to put in servings themselves

  Accuracy over ease of input

- Date + Caloric Intake → Not necessary useful enough to be put front and centre; can be put onto a different page/accessed if necessary

Additional features:

- PAST Meals should be separated into breakfast, lunch, dinner, snack, etc.

  Bundled into one entry

- Water intake?!

User #4, 22 year-old, male, interviewed on 02/27/2023:

Expectations:

- Something like MyFitnessPal
- Enter food
- Gives calories, macro, carbs, etc.
- From homepage, expects picture to be given that is similar and it identifies everything
- Does not want to specify quantities manually with image upload

Experience:

- Read instructions before use

- Used text upload as default
- Uploaded multiple items using both text and voice upload
- Tested using & symbol (worked as intended)
- Viewed food entries in profile page

Likes:

- Likes pictures on website
- Really likes landing page
- Colours are nice
- Good text font and size
- Likes voice upload functionality
- Likes most eaten food on profile page

Improvements:

- Have get started button to take you to upload page
- Nutrition facts should be rearranged nicer
- Have units for nutrition facts
- Have calories and more important ones at the top or bigger
- Split calories for each individual food item for multiple?
- Submit food entry on enter
- Be able to change serving unit (how big serving is)
- Have more error handling (if not food, spell wrong)
- Reset button in voice upload takes you to text upload
- Might not need upload button at top (just have buttons on profile and landing page)

- Add vertical padding for bottom of page so content is not at very bottom
- Use arrows for going between pages of food entries

Additional features:

- Total nutrients in day summary (carbs, protein, fat)
- Be able to set goals of how much you want to eat

  Give insights on goal progression
- Gets sense of meals eaten and give suggestions on what to eat to reach goals
- Warnings if you are eating something really unhealthy
- Should have graph (visuals better than text)
- Time when input in past meals
- More concerned with what was apart of which meal
- Wants to be able to delete entries
- Wants to know how many pages of saved food entries there are
- Have filters for past food entry display (ex. sort by day)

User #5, 22 year-old, female, interviewed on 02/28/2023:

Expectations:

- MyFitnessPal
- Nutritional information for food items
- Be able to use food while going out (like McDonald's brand and Starbucks for example)
- Ethnic foods available
- Have to upload all the ingredients

- Would like an app to use on her phone

Experience:

- Wasn't sure if she could use specific brands
- Didn't know can input multiple items (thought she could with comma but probably needs it more explicit)
- Didn't use natural language
- Disappointed in image upload (only 6 foods and food pictures have to be already downloaded to the computer, says its useless)
- A little bit confused about the button language on the profile page "Next should look at most recent stuff" need to make more clear
- Doesn't really care about most eaten food
- Accepted pizza without asking what type
- Oreo mcflurry showed as oreo AND mcflurry

Likes:

- Colours are nice

Improvements:

- Bold food stats in upload page
- Remove serving unit and quantity if multiple food items inputted
- Need to put units in upload page
- Need a static graph so she can be able to spam click the next or previous buttons without having to move her mouse
- Should notify user if there is a typo/food item that is not being searched
- Put units in total calories
- Format upload page nicer: food item, serving, unit should be on left and nutritional facts on the right

- Don't upload photo or take picture from laptop

Additional features:

- Want BMI calculator to calculate calorie deficit and differentiate from other apps
- Protein calculator on how to tone up, can put other sections in task bar
- Fiber and iron addition
- Select what options they would like displayed to them on profile page

References

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.)

There have been many changes from what was originally stated in the VnV plan. Most changes were due to project scope revisions (i.e. adding and removing requirements) or infeasibility of tests. After our POC demonstration, Utrition's scope was altered from only allowing a user to upload an image to also allowing text and voice upload. Due to this, our development team did not have tests planned for these new requirements as they had not been created at the time. The VnV plan has now been updated to test this functionality. Additionally, some tests that were previously created were removed because they were either no longer relevant with the altered scope or infeasible in terms of time/resources. Now that all team members have worked on a large project, identifying these time/resource constraints will make it easier for planning in future projects. Lastly, semi-structured interviews (usability testing) were added to provide unrestrained feedback from users regarding all aspects of Utrition's functionality and UI. Previously, our development team had not considered the possibility of user testing, but this was later pursued after all the feedback received during the POC and Revision 0 demonstrations proved to be helpful. It was not possible for Utrition's development team to anticipate the specific scope changes that occurred. However, we now know going forward to anticipate requirements changes (causing additions and deletions of tests) when user testing or demonstrations are conducted.