SFWR 4G06B - V and V - Rev 0

Group 9
Gundeep Kanwal 400015267
Ivan Bauer 001418765
Yousaf Shaheen 400026476
Scott Williams 400031554
Lucas Shanks 40002993

February 16, 2020

Contents

List of Tables

List of Figures

Revision History

Revision	Date	Author(s)	Description
0	02/10/2020	GK, IB, YS, SW, LS	Initial Version of V & V

1 Purpose

The purpose of this document is to showcase how we will be testing our application, Nutribud. It will highlight our verification and validation process, and how we are ensuring Nutribud is meeting our expectations, and following the behaviours that we desire the application to follow. This document will layout the test cases that we will be developing that will verify Nurtibud's integrity, functionality and maintainability. It will elaborate on our QA practices and procedures.

2 Scope

This document will describe the test strategy, objectives, expected results and actual results of each module of our software system. We will run white box tests in regards to the Python back-end services the control the functionality of the application, and black box tests in regards to the application's user interface. In addition to this, having integration tests and end-to-end tests will be crucial for us to ensure that the application is fully functional.

3 Background

Nutribud is an Android mobile application. It is designed to allow a user to track their nutritional habits through a daily diary. It will allow the user to add food items in various ways, such as taking a picture of a food item, scanning a bar-code, or by adding a food item manually through a search. Users will be able to add multiple food items to a meal. These meal entries will be added to a diary entry for that specific day.

The nutritional data for each diary entry will be displayed to the user through a timeline. Furthermore, a user's fitness progress will be visualized to the user through a timeline of it's own, using data entries regarding a user's weight at a time point. Nutribud will allow the user to customize their experience through various settings and options. Nutribud will allow the user to store their settings onto a database, it will enable an account system that each user creates.

Nutribud will implement a game system, which provides a framework for users to compete with one another. It will create scores from how well a user is able to stick to their goals. It will retrieve that information from a user's goal settings and their food diary entries. Nutribud goal setting feature will enable users to create fitness goals based on their weight desires, for a period of time.

4 Test Cases

4.1 Testing Plan and Testing Factors

Testing was performed according to the modules that were defined for the NutriBud application. The degree of logic within components that are described in this document was the determining factor with whether white box or black box testing was used for verification of the requirements. For components that require complex logic involving equations and numerical summation of nutritional information (i.e. BMR Equation), then white box testing was performed. This is to ensure that branch coverage is achieved and that the flow of the application for different inputs is handled in NutriBud. For components and modules that do not require as much logic and focus on the interaction between the user and logic modules, then black box testing was used.

Integration testing is critical to ensure that the validated requirements are adhered to for NutriBud. There are components in the application that are affiliated with one another, such that state changes in one module can lead to a state change in another module. Due to this nature, it was important to test connected components and state changes associated with them.

Component	Test Plan Test Factors
Login Module	The main goal of testing the Login Module is to ensure that users are able to create an account in order to access the important features of NutriBud. The tests for this module were split up between White Box Testing for User Authentication with NutriBud's Firestore database and black box testing for User Interface elements to ensure proper form validation is performed. For example, Test Cases 06 and 08 involve white box unit tests that verify that registration is performed under different circumstances, such as an email being newly created or checking if an email account already exists. However, there are other tests that involve black box testing, such as checking if no Wi-Fi connection on the phone can result in user interface elements being displayed to the user. It was important to test the Login Module under white box and black box to validate if requirements are met.
Profile Module	The main objective of testing the Profile Module is to verify that users will able to update their custom own account settings. It is paramount that the users are able to make these changes in order to enable accurate caloric calculations for a user. Features such as the food diary, and user goals depend on the profile module, it is a core piece within the architecture of Nutribud's software. There will be a combination of white box testing, black box testing and integration testing. The integration tests will especially be important, as profile module sets a foundation for a user's experience of Nutribud, it must work with the entire application.
Image Processing	This component includes modules on testing how images are processed from bring captured within the application, how they are processed, and how results are returned to the user. Testing was performed on this module through interface testing to ensure that interface elements appear according to the options that are returned through the IBM Watson API service. Additionally, white box testing was performed to ensure that appropriate actions are taken for displaying to the user of abnormalities in the IBM Watson responses. An example of this occurring is in Test Case 05, where no list of matches could be found.
Back-end Mobile App Service Module	The goal of testing this module is to ensure all of the endpoints function as expected. When using the application, the user must be able to contact the Nutritionix API in order to retrieve food nutritional data, as well as search options for manual search. Test Cases 01 and 02 are black box tests on a user interface level, with integration testing being performed in Test Case 02 where users are directed to a different screen for submitting serving amounts. Test Cases 03-05 involve white box functional tests to ensure appropriate API calls return valid 200 OK Status codes or other alternatives.
Food Diary Module	Testing of the Food Diary application will include most of the everyday interactions with the user. This involves adding Meals and Food Items successfully and updating Caloric Goal values. A majority of these tests are white box integration tests, as multiple classes within this module decomposition work together. Tests in this sense were accomplished according to database entries being updated and ensuring that the changes are reflected on the user-interface.

Component	Test Plan Test Factors
Timeline Module	The goal for testing the Timeline Module is to ensure that weight entries can be inserted into the timeline and that caloric and weight goals can easily be displayed through nodes on a graph visual that can be zoomed in and out of. The purpose of this module is so that the user can easily see their progress over time, from the time of account creation. The testing factors for this involved communication to the Firestore database service for weight entries being added/updated, and interface elements that would update according to changes made on the end user's listening Stream (i.e. a Flutter element that will listen to database changes).
GainRivals Module	The goal for testing the GainRivals module is to validate and verify the correctness of our gaming feature. It is incredibly important that this feature works to it's specifications as it brings a lively and interactive aspect to NutriBud. The mechanics involved with the gameplay, more specifically the calculations and the algorithms attatched to the scores will be given extra attention to when testing. GainRivals must be reliable, in order to ensure fairness. The connection between GainRivals and the rest of the application, such as the data retrieval from the food diary, will be very critical in terms of testing. The nutritional data retrieved from the food diary for each user is used to calculate their relative scores, and comparing them with each other in a leader-board. Testing the credibility of that data transfer is paramount for GainRivals to be successful.

4.2 Login Module

Test Number	Description	Requirement Reference	Inputs	Expected Outputs	Actual Output	Results
01	Login to account	FR9	Valid email and password	"Login successful" message	"Login successful" message	Pass
02	User cannot login	FR9	Invalid password	"invalid login credentials" message	"invalid login credentials" message	Pass
03	User cannot login	FR9	Invalid email	"invalid login credentials" message	"invalid login credentials" message	Pass
04	User can request a new password	FR9	"Forgot password" option selected	Sends an email with steps to password recovery	Not Implemented	Fail
05	Users can choose "Remember me" option, auto login after	FR9	Remember me box selected	User does not have to enter login credentials on next login	User does not have to enter login credentials on next login message	Pass
06	Register account	FR9	registerEmailPassword(email, password)	registerEmailPassword(email, password) = True	registerEmailPassword(email, password) = True	Pass
07	Password checking	FR9	"Register" is selected with email and password	"Your password is too weak" message	"Your password is too weak" message	Pass
08	User cannot register an email already in use	FR9	"Register" is selected with email and password and registerEmailPassword(email, password) called	registerEmailPassword(email, password) = false	registerEmailPassword(email, password) = true	Fail
09	Prevent user from logging in if Wi-Fi is not connected	FR9	"Login" is selected	"Wi-Fi Connection Issue" message displayed	No Response	Fail

Test Number	**Description**	**Requirement Reference**	**Inputs**	**Expected Outputs**	**Actual Output**	**Results**
10	Prevent user from registering if Wi-Fi is not connected	FR9	"Register" is selected	"Wi-Fi Connection Issue" message displayed	No Response	Fail
11	Prevent user from logging in if Wi-Fi is not connected	FR9	User enters username and password, and taps on the "Log In" button	"Wi-Fi Connection Issue" message displayed	No Response	Fail

4.3 Profile Module

Test Number	Description	Requirement Reference	Inputs	Expected Outputs	Actual Output	Results
01	User information can be registered to the account	FR9	User adds First Name, Last Name, Gender, Username, Height, DoB info to their profile	Information is added successfully	Information is added successfully	Pass
02	User can register their Fitness Goal	FR9	User adds a date and a weight goal	Info is updated on the user account	Info is updated on the user account	Pass
03	User can discard their changes	FR9	User removes any added info from the account	Data removed successfully	Data removed successfully	Pass

4.4 Back-end Mobile App Service Module

Test Number	Description	Requirement Reference	Inputs	Expected Outputs	Actual Output	Results
01	API returns correct nutritional information via a manual search food entry	FR4, FR3	User enters a manual food entry	Results similar to user entry are returned	Results similar to user entry are returned	Pass
02	API returns correct nutritional information via a barcode food entry	FR4, FR3, FR5	Barcode is scanned	User is directed to add servings amount with the corresponding food item	User is directed to add servings amount with the corresponding food item	Pass
03	API returns correct nutritional information based on IBM Watson selection entry	FR4, FR3, FR12, FR1, FR2, FR6	Image capture feature is selected	Returns the user a list to select from possible foods	Returns the user a list to select from possible foods	Pass
04	GET requests work correctly	FR4, FR3	GET Request for Nutritionix v2/instant/(get food nutrition facts) is sent	Returns a 200 ok message	200 ok message	Pass
05	The application will let users know if nutritional information can't be found	FR4, FR3	GET Request for Nutritionix v2/instant/(get food nutrition facts) is sent for a food that doesn't have information	"Not Food" is returned	"Not Food" is returned from API call	Pass

4.5 Image Processing Module

Test Number	Description	Requirement Reference	Inputs	Expected Outputs	Actual Output	Results
01	Food options are presented to the user when using image recognition feature	FR1, FR2, FR10, FR12	A picture of a food item	A list of 5 food items	a List of 5 food items	Pass
02	Food options that are returned are accurate	FR1, FR2, FR10, FR12, FR3	A picture of a food item	Correct food option returned in top 5 options at least 80% of the time after 10 different test cases	Correct food options returned in top 5 options 60% of the time after 10 different test cases	Fail
03	Users can manually search for food items if their food item is not found	FR1, FR2, FR10, FR12, FR6	User taps on the none of the above button	The user is directed to the manual search screen	The user is directed to the manual search screen	Pass
04	Adding a food item to a meal from the results	FR1, FR2, FR10, FR12	Selecting and submitting the desired food item from the list	Meal Adds Food Item and updates summary	Meal Adds Food Item and summary is updated	Pass
05	If no results are returned from IBM Watson, notify user that no matches could be found	FR1, FR2, FR10, FR12, FR6	Empty list or 400 Error Code exception caught	Dialog-ModalText	HTTP Status Exception (400 Error Code)	Fail

4.6 Food Diary Module

Test Number	Description	Requirement Reference	Inputs	Expected Outputs	Actual Output	Results
01	User can sign out	FR9	Sign out selected	Sends user to login screen and signs out of the account	Sends user to login screen and signs out of the account	Pass
02	User cannot modify Calorie Breakdown on previous day	FR4	Previous day is selected	Food item edit options appear	Not Implemented	Fail
03	User can access the timeline module	FR7, FR8, FR9	Timeline module selected	Timeline displayed	Timeline displayed	Pass
04	User can access GainRivals	FR9, FR11	GainRivals selected	GainRivals screen displayed	GainRivals screen displayed	Pass
05	User can access profile	FR9	User profile screen is selected	User profile is displayed	User profile is displayed	Pass
06	User can view diaries of other days	FR4	Diary date is selected	User is sent to the specified diary date	User is sent to the specified diary date	Pass
07	User can add a meal entry with multiple food items	FR4	User adds a second meal for the day	Meal is added underneath previously added entry	Meal is added underneath previously added entry	Pass
08	User can remove a Food Item from a Meal	FR4	User removes an item	Item disappears from meal	Item disappears from meal	Pass
09	User can delete a meal	FR4	User removes a meal	Meal with all food items are deleted	Meal with all food items are deleted	Pass
10	Meal items successfully summarize all food item nutritional information	FR4	Food item is added to the meal	Meal nutritional information is updated	Nutritional information is updated	Pass
11	User can add a food entry to a meal with camera	FR4, FR1, FR2, FR3, FR12, FR10	User adds a meal item using a picture	Food item is added to the meal	Food item is added to the meal	Pass
12	User can add a food entry to a meal with manual search	FR4	User adds a meal item using a search	Food item is added to the meal	Food item is added to the meal	Pass

Test Number	Description	Requirement Reference	Inputs	Expected Outputs	Actual Output	Results
13	User can add a food entry with barcode	FR4, FR5	User adds a meal item using the barcode scanner	Food item is added to the meal	Food item is added to the meal	Pass
14	User can add calories they burned in a day	FR4	User adds burned calories to the day	Values needed to reach daily goal are updated	Values needed to reach daily goal are updated	Pass
15	User can set their protein, carb and fat goals	FR4	User sets goal values	Goals are updated	Not Implemented	Fail
16	User calorie, protein, carbs, and fat counters change correctly as they add meals	FR4	Meal is added	Values are reflected accordingly	Values are reflected accordingly	Pass
17	User nutrition information changes correctly as calories burned is added	FR4	Food item is added	Needed daily calories are updated	Not Implemented	Fail
18	Serving selection	FR4	Food item is added	Select servings menu is opened	Select servings menu is opened	Pass

4.7 Timeline Module

Test Number	Description	Requirement Reference	Inputs	Expected Outputs	Actual Output	Results
01	Users can log weight entries	FR7, FR8	A User fills in a weight value after tapping the "Add Weight" button on screen	Weight is logged as a weight entry on screen	Weight is logged as a weight entry on screen	Pass
02	Weight entries are updated on the timeline as nodes	FR7, FR8	Weight entry is added	The new weight entry is added onto the timeline graph as a node	The new weight entry is added onto the timeline graph as a node	Pass
03	Weight timeline is visualized correctly as a line graph connecting the nodes	FR7, FR8	A list of weight points and their timestamps	A line graph displayed, connecting the weight entries together	A line graph displayed, connecting the weight entries together	Pass
04	Caloric entries are added to the timeline as nodes	FR7, FR8	A food diary is entered as the day rolls over	The caloric intake for that day is saved onto the timeline	The caloric intakes are not saved onto the timeline	Fail
05	Caloric timeline nodes are updated correctly	FR7, FR8	Food item is added to the diary	Chart updated with new calorie count for the day	The caloric values of a day are not updated as they are changed throughout the day	Fail
06	User can return to the food diary module	FR7, FR8	User presses back	User is navigated to the previous menu	User is sent to food diary module	Pass
07	User can zoom in and out on all timelines	FR7, FR8	User uses two fingers on the graph	Graphs zoom in	Graphs zoom in	Pass

4.8 GainRivals Module

Test Number	Description	Requirement Reference	Inputs	Expected Outputs	Actual Output	Results
01	User can create a game with their personalized parameters	FR 11	User presses create a game option	Unique game is created	Unique game is created	Pass
02	User can join game with a game ID and their username	FR9, FR11	Game ID and Username	User is added to the game	User is added to the game	Pass
03	User can leave game, that redirects to main game screen	FR9	User leaves the game	User is removed from the game	User is removed from the game	Pass
04	User can return to food diary module	FR9, FR4	User presses the back button	User is navigated back to the main menu	User is navigated back to the main menu	Pass
05	Calculated scores based on how closely users followed their goal (below 100 percent)	FR9	User fulfills goal	Score is updated on the GainRivals screen	Score is updated on the GainRivals screen	Pass
06	If above 100% on a particular macro goal, then score will be set to zero relating to a user's macro	FR9	calcu-lateScores(prot ein Percent: 80, fatsPercent: 106, carb-sPercent: 80)	userScores [userId-date-fats] = 94	userScores[userId-date-fats] = 100 -date-fats]=100	Fail

5 Beyond Testing

5.1 Code Walk Through and Code Reviews

A low-level inspection of the code will be conducted in order to optimize the overall performance and structure of the code. This includes time and space complexity, comments, variable nomenclature, etc. Improving the time complexity of the application will improve the overall responsiveness and will use less CPU utilization to complete tasks. Reducing the space complexity is particularly important due to the hardware limitations of cellular devices. Storage on a cellular device is not as abundant as other devices, as such the application needs to use as little space as possible. Commenting the code will allow fellow team members and future developers to easily implement new features and/or release patches, as well as refactoring. Having good nomenclature will also help fellow team members and future developers to implement features, release patches, and improve developer comprehension of the code base.

5.2 Review/Inspections

The application will undergo a usability and general functionality inspection and review. The usability review will ensure that the application is intuitive and easy to use. This will be achieved by bringing in testers who

have not seen the application in the past and asking them to perform basic tasks. Based on the results of the testing, certain aspects of the application may be changed (eg. GUI, controls, etc.). The functionality review will be performed to make sure the basic functions of the application are not only working correctly, but are working in a timely manner. Having a responsive application is critical, even if everything else if working, a slow app will be highly undesirable and will eventually end in failure.

5.3 Static Analysis

Static analysis will be performed on new code whenever it is added in chucks and merged into our application through GitHub. We will inspect added code on our code editors, taking the time to look through the code for any potential errors that may be introduced by integrating new software into Nutribud. The developer of any lines of code will be responsible for understanding the software that they create, in order to teach others how their software computes. It will be standard practice for developers to comment their code effectively in order to enable improved static analysis.

5.4 Formal Proof

Formal proofs will be carried out when validating and verifying how our application works with any of our algorithms that we have developed within our application. We will be proving our software application formally by ensuring our algorithms are working as expected. We will be testing the arithmetic of our algorithms by ensuring that they are calculating BMR, BMI, and the various goals for a user correctly. There will also be a formal proof to verify that the various timelines will be displaying values correctly based off the algorithms they use to gather the data required to build those graphs.

6 Supporting Material

Please refer to previous documentation for the Nutribud application.

7 Trace-ability Matrices

Test Cases:	Functional And Non-Functional Requirements
F1:Login Module	FR4 FR9 FR11 NFR6 NFR5 NFR4 NFR3 NFR1

Table 2: Login Page Trace-ability

Test Cases:	Functional And Non-Functional Requirements
F1:Profile Module	FR4 FR6 FR7 FR8 FR9 FR11 NFR6 NFR5

Table 3: Profile Page Trace-ability

Test Cases:	Functional And Non-Functional Requirements
F1:Back-end Mobile App Service Module	FR2 FR3 FR4 FR6 FR7 FR8 FR9 FR10 FR11 FR12 NFR5 NFR3 NFR1

Table 4: Back-end Mobile App Service Trace-ability

Test Cases:	Functional And Non-Functional Requirements
F1:Image Processing Module	FR2 FR5 FR10 FR12 NFR6 NFR3 NFR1

Table 5: Image Processing Trace-ability

Test Cases:	Functional And Non-Functional Requirements
F1:Food Diary Module	FR3 FR4 FR6 FR7 FR8 FR11 NFR6 NFR3 NFR2 NFR1

Table 6: Food Diary Trace-ability

Test Cases:	Functional And Non-Functional Requirements
F1:Timeline Module	FR4 FR7 FR8 FR11 NFR6

Table 7: Timeline Trace-ability

Test Cases:	Functional And Non-Functional Requirements
F1:GainRivals Module	FR11 NFR6 NFR3

Table 8: GainRivals Trace-ability