Verification and Validation Report: REVITALIZE

Team 13,
Bill Nguyen
Syed Bokhari
Hasan Kibria
Mahmoud Anklis
Youssef Dahab
Logan Brown

April 5, 2023

1 Revision History

Date	Version	Notes
March 5th, 2023	Bill Nguyen	Adding Unit Tests for Workout and Rest Section
March 5th, 2023	Youssef Dahab	Added Functional Requirements Evaluation
March 6th, 2023	Youssef Dahab	Added Changes Due To Testing
March 8th, 2023	Hasan Kibria	Adding Unit Tests for Diet Section
March 8th, 2023	Youssef Dahab	Added Reflection
March 8th, 2023	Logan Brown	Added Non-Functional Requirements Evaluation
March 8th, 2023	Mahmoud Anklis	Added Unit Tests for the User Section and Reflection

2 Symbols, Abbreviations and Acronyms

symbol	description
REVITALIZE	Name of application
SRS	Software Requirements Specification
VnV	Verification and Validation
FR	Functional Requirement
NFR	Non Functional Requirement
LP	Login Page
SP	Sign-up Page
MP	Main Page or Maintainability and Portability Requirements
DS	Diet Section
WS	Workout Section
RS	Rest Section
LF	Look and Feel Requirements
UH	Usability and Humanity Requirements
PE	Performance Requirement
OE	Operational Requirement
SE	Security Requirement
CU	Cultural Requirement

2.1 Symbolic Parameters

MINIMUM_TEST_SCORE = 8.5
MINIMUM_TEST_SCORE_2 = 9.5
MAXIMUM_ACCESS_TIME = 10
MIN_APPROVAL_RATING = 85%
MIN_APPROVAL_RATING_2 = 95%
MIN_USER_LOAD = 50
MIN_DATA_POINTS = 1000000

Contents

List of Tables

List of Figures

This document details the complete testing process for REVITALIZE, as laid out in the project test plan. It contains an evaluation of the project's functional and non-functional requirements that are defined in the **SRS**, the changes made due to testing, and an analysis of the traceability between requirements and modules.

3 Functional Requirements Evaluation

3.1 Login Page

Test #1:	**FR-LP-1**
Description:	Testing that login page is displayed upon starting the application
Type:	Manual
Initial State:	Loading stage of the login page
Input:	An event that loads the login page
Output:	Login page is displayed with all necessary components
Expected:	
Result:	PASS

Test #2:	**FR-LP-2**
Description:	Testing that login page displays fillable username textboxes
Type:	Manual
Initial State:	Login page is displayed with username and password textboxes
Input:	~~Enter username information in textbox~~ Valid username/email and password information entered in their respective textboxes (Valid Inputs: email = "johndoe@gmail.com" and password = "qwerty123")
Output:	~~Username information entered is displayed in textbox~~ Returns a success message and user is successfully logged in and redirected to the main page
Expected:	
Result:	PASS

Test #3:	**FR-LP-3**
Description:	Verifies that a user fails to log when providing invalid username/email and password information
Type:	Automatic, Functional, Dynamic
Initial State:	Login page is displayed with username and password textboxes (Users Current Data: email = "johndoe@gmail.com" and password = "qwerty123")
Input:	Invalid username/email and password information entered in their respective textboxes (Invalid Input 1: email = "notjohndoe@gmail.com" and password = "qwerty123"), (Invalid Input 2: email = "" and password = ""), (Invalid Input 3: email = "johndoe@gmail.com" and password = "wrongqwerty123")
Output:	User unsuccessfully logins and returns 400 message (Invalid Output 1: "Email not found. Please sign up"), (Invalid Output 2: "Please fill in all fields"), (Invalid Output 3: "Incorrect password")
Expected:	
Result:	PASS

Test #4:	**FR-LP-4**
Description:	Verifies that the forget password page loads correctly when requested, and that all necessary components are present on the page
Type:	Manual, Functional, Dynamic
Initial State:	Login page is displayed with "forgot password" button
Input:	~~Click forgot password button~~ User clicks on the "forgot password" button
Output:	~~Display forgot password screen with textbox to enter email~~ The system displays the "forgot password" screen with a text box to enter the email.
Expected:	Tester will click on forgot password button and checks if forgot password screen is displayed with textbox to enter email
Result:	PASS

Test #5:	**FR-LP-5**
Description:	Verifies the system must allow users to reset their password if they forget it
Type:	Manual, Functional, Dynamic
Initial State:	The user is on the login page and has forgotten their password
Input:	One string: the user's registered email address
Output:	An email sent to the user's registered email address with a link to reset their password
Expected:	
Result:	PASS

Test #6:	**FR-LP-6**
Description:	Verifies that the login page displays a sign up button that redirects to the sign up page when clicked
Type:	Manual, Functional, Dynamic
Initial State:	Login page is displayed with sign up button
Input:	Click on the sign up button
Output:	~~Loads and displays sign up page~~ The system redirects the user to the sign up page.
Expected:	
Result:	PASS

3.2 Signup Page

Test #7:	**FR-SP-1**
Description:	Testing that signup page displays fillable username textbox
Type:	Manual
Initial State:	~~Signup page is displayed with username textbox~~ Signup page is displayed with a blank textboxes for username, email, password and confirm password
Input:	~~Enter username information in textbox~~ Enter username, email, password and confirm password information in their respective textboxes. (Valid Inputs: Username = john123, Password = PasSw0rd145, Email = john123@gmail.com, Confirm Password = PasSw0rd145)
Output:	~~Username information entered is displayed in textbox~~ Upon clicking the "Sign up" button, the user's account is created and the user is logged in to the system. (Valid Output: successful message of "Congrats!','Your account has been successfully created" and user added to database)
Expected:	
Result:	PASS

Test #8:	**FR-SP-2**
Description:	Testing that a new user can not sign up to REVITALIZE when there is invalid information
Type:	~~Manual~~ Automatic, Functional, Dynamic
Initial State:	~~Signup page is displayed with password textbox~~ Signup page is displayed with a blank textboxes for username, email, password and confirm password
Input:	~~Enter password information in textbox~~ Enter invalid username, email, password and confirm password information in their respective textboxes. (Invalid Input: Username = !* , Password = abc123, Email = invalid-a-gmail, Confirm Password = abc123)
Output:	~~Password information entered is displayed in textbox via hidden text~~ Upon clicking the "sign up" button, the new user's account is not added to database and alert with failure message will appear. (Invalid Output: failure message of "Invalid Input, Improper Email Address")
Expected:	
Result:	PASS

Test #9:	**FR-SP-3**
Description:	Tests that a new user can not sign up to REVITALIZE when email already exists in database
Type:	Automatic, Functional, Dynamic
Initial State:	Signup page is displayed with a blank textboxes for username, email, password and confirm password. One user already exists in database (Username = john123, Password = PasSw0rd145, Email = john123@gmail.com, Confirm Password = PasSw0rd145)
Input:	Enter same username, email, password and confirm password of existing user. (Invalid Input: Username = john123, Password = PasSw0rd145, Email = john123@gmail.com, Confirm Password = PasSw0rd145)
Output:	Upon clicking the "sign up" button, the new user's account is not added to database and alert with failure message will appear. (Invalid Output: failure message of "User already exists. Please login")
Expected:	
Result:	PASS

Test #10:	**FR-SP-4**
Description:	Tests that a new user can not sign up to REVITALIZE when there are empty fields
Type:	Automatic, Functional, Dynamic
Initial State:	Signup page is displayed with a blank textboxes for username, email, password and confirm password.
Input:	Do not enter username, email, password and confirm password textboxes. (Invalid Input: Username = "" , Password = "", Email = "", Confirm Password = "")
Output:	Upon clicking the "sign up" button, the new user's account is not added to database and alert with failure message will appear. (Invalid Output: failure message of "Please fill in all fields")
Expected:	
Result:	PASS

Test #11:	**FR-SP-5**
Description:	Tests that a new user can not sign up to REVITALIZE when password and confirm password fields do not match
Type:	Automatic, Functional, Dynamic
Initial State:	Signup page is displayed with a blank textboxes for username, email, password and confirm password.
Input:	Enter any username and email but enter password and confirm password that do not match. (Invalid Input: Username = "Bob Test" , Password = "qwerty123", Email = "bobtest@gmail.com", Confirm Password = "ytrewq321")
Output:	Upon clicking the "sign up" button, the new user's account is not added to database and alert with failure message will appear. (Invalid Output: failure message of "Invalid Input, Passwords do not match")
Expected:	
Result:	PASS

3.3 Main Page

Test #12:	**FR-MP-1**
Description:	Testing that the application displays a calendar with current date on successful login
Type:	Manual, Functional, Dynamic
Initial State:	Main page is displayed with calender of current date
Input:	An event that loads the main page
Output:	Main page is displayed with all necessary components
Expected:	
Result:	PASS

Test #13:	**FR-MP-2**
Description:	Testing that the application has a previous day and a next day button on each page after successful login
Type:	Manual,Functional, Dynamic
Initial State:	Main page and Diet, Workout, Rest sections are displayed with previous day and next day buttons
Input:	An event that loads the main page, Diet, Workout, Rest sections and the previous day and next day buttons are clicked
Output:	Main page, Diet, Workout, Rest sections are displayed with previous day and next day buttons. Once the next day button is clicked, the calendar refreshes the calendar information for the next day. Once the previous day button is clicked, the calendar refreshes the calendar information for the previous day
Expected:	
Result:	PASS

Test #14:	**FR-MP-3**
Description:	Testing that a back button is displayed on each user interface after a section is selected
Type:	Manual,Functional, Dynamic
Initial State:	Each interaction after leaving the main page must have a visible back button
Input:	An event that loads the next user interface after leaving the main page and the back button is clicked
Output:	The next user interface after leaving the main page is displayed with a back button. Once the back button is clicked the main page is loaded
Expected:	
Result:	PASS

Test #15:	**FR-MP-4**
Description:	Testing that the application displays the sections Diet, Exercise, and Rest on the current calendar day
Type:	Manual,Functional, Dynamic
Initial State:	Main page is displayed with Diet, Exercise and Rest buttons available to click
Input:	An event that loads the main page and the Diet, Exercise and Rest buttons are clicked
Output:	Main page is displayed with Diet, Exercise and Rest buttons. If the Diet button is clicked, the Diet interface is loaded. If the Exercise button is clicked, the Exercise interface is loaded. If the Rest button is clicked, the Rest interface is loaded
Expected:	
Result:	PASS

3.4 Diet Section Page

Test #16:	**FR-DS-1**
Description:	Tests that the Diet section must initialize with a list of food logged on the current calendar day
Type:	Manual,Functional, Dynamic
Initial State:	~~section~~ Diet section is initialized with a list of food logged for the current calender day. Assume this is the current data in the database for test@gmail.com and the date of 2022-10-25: [(foodName = "Oven Fried Chicken II", calories = 200, carbs = 50, fats = 120, protein = 125, foodDate = 2022-10-25 email = "test@gmail.com"), (foodName = "Lasagna", calories = 100, carbs = 70, fats = 140, protein = 145, foodDate = 2022-10-25 email = "test@gmail.com")]
Input:	~~An event that loads the rest section~~ An event that loads/gets the food log for email = test@gmail.com and foodDate = 2022-10-25
Output:	A list of inputted food is loaded for the current calender day . For email = test@gmail.com and foodDate = 2022-10-25, will return a successful message = "Success in getting food log", a list of food names = ["Oven Fried Chicken II", "Lasagna"] and get the total calories, carbs, fats and proteins for the calendar day [calories = 200 + 100, carbs = 50 + 70, fats = 120 + 140, protein = 125 + 145]
Expected:	
Result:	PASS

Test #17:	**FR-DS-2**
Description:	Verifies that a new meal can be added to database successfully
Type:	~~Manual~~ Automatic, Functional, Dynamic
Initial State:	~~Diet section is displayed with add food button~~ Database for Diet Section is empty and user (test@gmail.com) wants to add any type of meal (searched recipe or custom meal), for the selected day (2022-10-25)
Input:	~~Click add food button~~ Fill in the required fields for adding a new meal in the database (foodName = "Oven Fried Chicken II", calories = 200, carbs = 50, fats = 120, protein = 125, foodDate = 2022-10-25 email = "test@gmail.com")
Output:	~~A user interface is launched that lets the user select between searching for food or adding a custom meal~~ Input gets added to the database and should have the following success message = "Meal successfully added"
Expected:	
Result:	PASS

Test #18:	**FR-DS-3**
Description:	Tests that an existing meal can be updated to database successfully
Type:	Automatic, Functional, Dynamic
Initial State:	Database for Diet Section has an existing meal for user (test@gmail.com) with the following data (foodName = "Oven Fried Chicken II", calories = 200, carbs = 50, fats = 120, protein = 125, foodDate = 2022-10-25 email = "test@gmail.com") and user wants to edit calories and protein data
Input:	Fill in the fields for the user wants to update in the database (calories = 300, protein = 225, foodDate = 2022-10-25 email = "test@gmail.com")
Output:	Input gets updated to the database and should have the following success message = "Success in updating meal" and the updated meal data should look like this (foodName = "Oven Fried Chicken II", calories = 300, carbs = 50, fats = 120, protein = 225, foodDate = 2022-10-25 email = "test@gmail.com")
Expected:	
Result:	PASS

Test #19:	**FR-DS-4**
Description:	Verifies that an existing meal can be deleted from database successfully
Type:	Automatic, Functional, Dynamic
Initial State:	Database for Diet Section has an existing meal for user (test@gmail.com) with the following data (foodName = "Oven Fried Chicken II" calories = 200, carbs = 50, fats = 120, protein = 125, foodDate = 2022-10-25 email = "test@gmail.com") and user wants to delete this meal data
Input:	Fill in the email, food date and food name fields for the user to delete meal data in the database (foodName = "Oven Fried Chicken II", foodDate = 2022-10-25 email = "test@gmail.com")
Output:	Selected meal data is deleted from the database and should have the following success message = "Success in deleting meal"
Expected:	
Result:	PASS

Test #20:	**FR-DS-5**
Description:	Tests that searching for a recipe/meal is successful
Type:	Manual, Functional, Dynamic
Initial State:	User wants to search for recipe and Diet Recipe Search Screen is loaded, with appropriate textboxes (Name of Meal and Calories) and appropriate dropdowns (List of Diet Types and List of Food Restrictions and Preferences)
Input:	Fill all textboxes and dropdowns (foodName = "Chicken", Calories = 1000, Diet Type = "high-protein", Food Restrictions and Preferences = "low-sugar")
Output:	Should return a wide range of recipes and meals that satisfies all the conditions from the input
Expected:	
Result:	PASS

3.5 Workout Section Page

Test #21:	**FR-WS-1**
Description:	Testing that the Workout section initializes with a preset list of exercises on the current calendar day
Type:	~~Manual~~ Automatic, Functional, Dynamic
Initial State:	Workout section is initialized with a preset list of exercises of the current calendar day
Input:	An event that loads the workout section
Output:	A preset list of exercises is loaded for the current calendar day
Expected:	
Result:	PASS

Test #22:	**FR-WS-2**
Description:	Verifies that a new workout can be added to database successfully
Type:	Automatic, Functional, Dynamic
Initial State:	Database for Workout Section is empty and user (test@gmail.com) wants to add new workout, for the selected day (2022-10-25)
Input:	Fill in the required fields for adding a new workout in the database (name = "Bicep Curl", weight = 50, sets = 10, repetitions = 15, dateAdded = 2022-10-25, email = "test@gmail.com")
Output:	Input gets added to the database and should have the following success message = "Success in adding exercise data"
Expected:	
Result:	PASS

Test #23:	**FR-WS-3**
Description:	Tests that an existing workout can be updated to database successfully
Type:	Automatic, Functional, Dynamic
Initial State:	Database for Workout Section has an existing workout for user (test@gmail.com) with the following data (name = "Bicep Curl", weight = 50, sets = 10, repetitions = 15, dateAdded = 2022-10-25, email = "test@gmail.com") and user wants to edit sets and repetitions data
Input:	Fill in the fields for the user wants to update in the database (sets = 20, repetitions = 25, dateAdded = 2022-10-25 email = "test@gmail.com")
Output:	Input gets updated to the database and should have the following success message = "Success in editing exercise data" and the updated workout data should look like this (name = "Bicep Curl", weight = 50, sets = 20, repetitions = 25, dateAdded = 2022-10-25, email = "test@gmail.com")
Expected:	
Result:	PASS

Test #24:	**FR-WS-4**
Description:	Tests that an existing workout can be deleted from database successfully
Type:	Automatic, Functional, Dynamic
Initial State:	Initial State: Database for Workout Section has an existing workout for user (test@gmail.com) with the following data (name = "Bicep Curl", weight = 50, sets = 10, repetitions = 15, dateAdded = 2022-10-25, email = "test@gmail.com") and user wants to delete this workout data
Input:	Fill in the email, date added and name fields for the user to delete workout data in the database (name = "Bicep Curl", dateAdded = 2022-10-25 email = "test@gmail.com")
Output:	Selected workout data is deleted from the database and should have the following success message = "Success in deleting exercise data"
Expected:	
Result:	PASS

Test #25:	**FR-WS-5**
Description:	Tests that the Workout section must initialize with a list of workouts on the current calendar day
Type:	Automatic, Functional, Dynamic
Initial State:	Workout section is initialized with a list of workouts logged for the current calender day. Assume this is the current data in the database for test@gmail.com and the date of 2022-10-25: [((name = "Bicep Curl", weight = 50, sets = 10, repetitions = 15, dateAdded = 2022-10-25, email = "test@gmail.com"),(name = "Bench Press", weight = 50, sets = 13, repetitions = 12, dateAdded = 2022-10-25, email = "test@gmail.com")]
Input:	An event that loads/gets the workout list for email = test@gmail.com and dateAdded = 2022-10-25
Output:	A list of workouts is loaded for the current calender day. For email = test@gmail.com and dateAdded = 2022-10-25, will return a successful message = "Success in getting exercise list" and will return the following list [((name = "Bicep Curl", weight = 50, sets = 10, repetitions = 15, dateAdded = 2022-10-25, email = "test@gmail.com"),(name = "Bench Press", weight = 50, sets = 13, repetitions = 12, dateAdded = 2022-10-25, email = "test@gmail.com")
Expected:	
Result:	PASS

3.6 Rest Section Page

Test #26:	**FR-RS-1**
Description:	Tests that Rest section launches with sleep statistics of current calendar day
Type:	Manual, Functional, Dynamic
Initial State:	Rest section is initialized with the sleep statistics of the current calendar day
Input:	An event that loads the rest section
Output:	Sleep statistics are loaded for the current calendar day
Expected:	
Result:	PASS

Test #27:	**FR-RS-2**
Description:	Tests that user can alter inaccurate sleep data
Type:	Manual, Functional, Dynamic
Initial State:	Rest section is initialized with the sleep statistics of the current calendar day
Input:	Alter sleep data
Output:	The sleep data is updated with user changes
Expected:	
Result:	PASS

Test #28:	**FR-RS-3**
Description:	Tests that a new sleep data can be added to database successfully
Type:	Automatic, Functional, Dynamic
Initial State:	Initial State: Database for Rest Section is empty and user (test@gmail.com) wants to add new sleep data, for the selected day (2022-10-25)
Input:	Fill in the required fields for adding new sleep data in the database (email: "test@gmail.com", sleepHour: 12, bedHour: 10, sleepMinute: 5, bedMinute: 5, dateAdded: 2022-10-25)
Output:	Input gets added to the database and should have the following success message = "Success in adding sleep data"
Expected:	
Result:	PASS

Test #29:	**FR-RS-4**
Description:	Tests that an existing sleep data can be updated to database successfully
Type:	Automatic, Functional, Dynamic
Initial State:	Initial State: Database for Rest Section has an existing sleep data for user (test@gmail.com) with the following data (email: "test@gmail.com", sleepHour: 12, bedHour: 10, sleepMinute: 5, bedMinute: 5, dateAdded: 2022-10-25) and user wants to edit sleep hour and sleep minute data
Input:	Fill in the fields for the user wants to update in the database (sleepHour: 2, sleepMinute: 10, dateAdded = 2022-10-25 email = "test@gmail.com")
Output:	Input gets updated to the database and should have the following success message = "Success in editing sleep data" and the updated sleep data should look like this (email: "test@gmail.com", sleepHour: 2, bedHour: 10, sleepMinute: 10, bedMinute: 5, dateAdded: 2022-10-25)
Expected:	
Result:	PASS

Test #30:	**FR-RS-5**
Description:	Tests that an existing sleep data can be deleted from database successfully
Type:	Automatic, Functional, Dynamic
Initial State:	Initial State: Database for Rest Section has an existing sleep data for user (test@gmail.com) with the following data (email: "test@gmail.com", sleepHour: 2, bedHour: 10, sleepMinute: 10, bedMinute: 5, dateAdded: 2022-10-25) and user wants to delete this sleep data
Input:	Fill in the email and date added fields for the user to delete sleep data in the database (dateAdded = 2022-10-25 email = "test@gmail.com")
Output:	Selected sleep data is deleted from the database and should have the following success message = "Success in deleting sleep data"
Expected:	
Result:	PASS

Test #31:	**FR-RS-6**
Description:	Tests that the Rest section must initialize with last saved sleep data on the current calendar day
Type:	Automatic, Functional, Dynamic
Initial State:	Rest section is initialized with last saved sleep data for the current calender day. Assume this is the current data in the database for test@gmail.com and the date of 2022-10-25: (email: "test@gmail.com", sleepHour: 2, bedHour: 10, sleepMinute: 10, bedMinute: 5, dateAdded: 2022-10-25)
Input:	An event that loads/gets the last saved sleep data for email = test@gmail.com and dateAdded = 2022-10-25
Output:	The last saved sleep data is loaded for the current calender day. For email = test@gmail.com and dateAdded = 2022-10-25, will return a successful message = "Success in getting sleep data"
Expected:	
Result:	PASS

4 Nonfunctional Requirements Evaluation

4.1 Look and Feel

Test #32:	**NFR-LF1**
Description:	Testing that UI/UX elements are displayed neatly and correctly
Type:	Manual
Tester(s):	Stakeholders
Pass:	Average Q1 survey score of at least MINIMUM_TEST_SCORE
Result:	PASSED with an agreement of 8.8 out of 10

Test #33:	**NFR-LF2**
Description:	Testing that colours used are acceptable
Type:	Manual
Tester(s):	Stakeholders
Pass:	Average Q2 survey score of at least MINIMUM_TEST_SCORE
Result:	PASSED with an agreement of 10 out of 10

4.2 Usability and Humanity

Test #34:	**NFR-UH1**
Description:	Testing accessibility of application using navigation ability with one finger
Type:	Manual
Tester(s):	Stakeholders
Pass:	Average survey score of at least MINIMUM_TEST_SCORE_2
Result:	PASSED with an agreement of 10 out of 10

Test #35:	**NFR-UH2**
Description:	Testing navigation speed between screens
Type:	Manual
Tester(s):	Stakeholders
Pass:	Average survey score of at least MAXIMUM_ACCESS_TIME
Result:	PASSED with an agreement of 10 out of 10

Test #36:	**NFR-UH3**
Description:	Testing overall accessibility through average survey result
Type:	Manual
Tester(s):	Stakeholders
Pass:	Average survey score of at least MINIMUM_TEST_SCORE
Result:	PASSED with an agreement of 9.3

Test #37:	**NFR-UH4**
Description:	Testing learnability of application
Type:	Manual
Tester(s):	Stakeholders
Pass:	MIN_APPROVAL_RATING of stakeholders understand functionality in 3 iterations or less
Result:	PASS

Test #38:	**NFR-UH5**
Description:	Testing consistency of UI
Type:	Manual
Tester(s):	Stakeholders
Pass:	Average survey score of at least MINIMUM_TEST_SCORE
Result:	PASSED with an agreement of 9 out of 10

4.3 Performance

Test #39:	**NFR-PE1**
Description:	Testing load times of API responses and outputs
Type:	Manual
Tester(s):	Developers
Pass:	Load times below 5 seconds
Result:	PASS

Test #40:	**NFR-PE2**
Description:	Testing accuracy of calculated values that contain data/numbers
Type:	Manual
Tester(s):	Developers
Pass:	
Result:	PASS

Test #41:	**NFR-PE3**
Description:	Testing system performance under high load
Type:	Manual
Tester(s):	Developers
Pass:	Previous metrics still pass with MIN_USER_LOAD users
Result:	Tentative Pass

Test #42:	**NFR-PE4**
Description:	Testing system performance with large amounts of user data
Type:	Manual
Tester(s):	Developers
Pass:	Previous metrics still pass with MIN_DATA_POINTS per user
Result:	PASS

4.4 Operational

Test #43:	**NFR-OE1**
Description:	Testing if all features are loaded with stable internet connection
Type:	Manual
Tester(s):	Developers
Pass:	
Result:	PASS

4.5 Maintainability and Portability

Test #44:	**NFR-MP1**
Description:	Testing maintainability through cross referencing developer comments
Type:	Manual
Tester(s):	Developers
Pass:	
Result:	PASS

4.6 Security

Test #45:	**NFR-SE1**
Description:	Testing that passwords are hashed and user data is secure
Type:	Manual
Tester(s):	Developers
Pass:	
Result:	PASS

Test #46:	**NFR-SE2**
Description:	Testing that emails can only have 1 associated account
Type:	Manual
Tester(s):	Developers
Pass:	
Result:	PASS

4.7 Cultural and Political

Test #47:	**NFR-CU1**
Description:	Testing that the displayed language is in English
Type:	Manual
Tester(s):	Developers
Pass:	
Result:	PASS

4.8 Overview of Testers

NFS tests were conducted by Logan, Faiq, and Youssef

5 Comparison to Existing Implementation

N/A

6 Unit Testing

6.1 Workout Section

Unit tests for the workout section: https://github.com/BillNguyen1999/REVITALIZE/blob/main/src/SERVER/backend/test/exercise.test.js.

Test ID	FR	Inputs	Expected Values	Actual Values	Result
WS1	FR-WS-1 and FR-WS-5	{email: 'test@gmail.com', dateAdded: '2022-01-01'}	[name: 'Exercise 1' , name: 'Exercise 2']	[name: 'Exercise 1' , name: 'Exercise 2']	PASS
WS2	FR-WS-1 and FR-WS-5	{email: 'fail@gmail.com', 'dateAdded: 2022-01-01'}	'Error in getting exercise list'	'Error in getting exercise list'	PASS
WS3	FR-WS-2	{ success: true, message: 'Success in adding exercise data', id: 'exerciseid', email: 'test@gmail.com', name: 'Test Exercise', sets: 3, repetitions: 10, weight: 50, dateAdded: '2022-03-07'}			PASS
WS4	FR-WS-3	{email: 'test@gmail.com', dateAdded: '2022-01-01', name:'push-ups'}	{success: true, message: 'Success in deleting exercise data'}	{success: true, message: 'Success in deleting exercise data'}	PASS
WS5	FR-WS-3	{email: 'notfound@gmail.com', dateAdded: '2022-01-01', name:'push-ups'}	{success: false, message: 'Was not able to delete selected exercise data'}	{success: false, message: 'Was not able to delete selected exercise data'}	PASS

Table 1: Workout Section Unit Tests Part 1

Test ID	FR	Inputs	Expected Values	Actual Values	Result
WS6	FR-WS-4	params: { email: 'example@gmail.com', dateAdded: '2022-01-01', name: 'exerciseName' }, body: { reps: 10, sets: 3 }	{success: true, message: 'Success in editing exercise data'}	{success: true, message: 'Success in editing exercise data'}	PASS
WS7	FR-WS-4	params: { email: 'notfound@gmail.com', dateAdded: '2022-03-07', name: 'push-ups' }, body: { sets: 3, reps: 10}	{success: false, message: "Was not able to find appropriate exercise data to edit" }	{success: false, message: "Was not able to find appropriate exercise data to edit" }	PASS
WS8	FR-WS-1 and FR-WS-5	{email: test@gmail.com, name:'pushup', dateAdded: 2022-01-01}	{success: true, message: 'Success in getting exercise data' }	{success: true, message: 'Success in getting exercise data' }	PASS
WS9	FR-WS-1 and FR-WS-5	{email: 'test@gmail.com', name: 'pushup', dateAdded: 'invalid-date' }	{success: false, message: 'Error in getting exercise data' }	{success: false, message: 'Error in getting exercise data' }	PASS

Table 2: Workout Section Unit Tests Part 2

6.2 Rest Section

Unit tests for the rest section: https://github.com/BillNguyen1999/REVITALIZE/blob/main/src/SERVER/backend/test/sleep.test.js.

Test ID	FR	Inputs	Expected Values	Actual Values	Result
RS1	FR-RS-1 and FR-RS-2	{email: 'test@gmail.com', dateAdded: '2022-01-01'}	{success: true, message: 'Success in getting sleep data'}	{success: true, message: 'Success in getting sleep data'}	PASS
RS2	FR-RS-1 and FR-RS-2	{email: 'test@gmail.com', dateAdded: 'invalid-date'}	{success: false, message: 'Error in getting sleep data'}	{success: false, message: 'Error in getting sleep data'}	PASS
RS3	FR-RS-1	{ success: true, message: 'Success in adding sleep data', id: 'sleepid', email: 'test@gmail.com', sleepHour: 12, bedHour: 10, sleepMinute: 5, bedMinute: 5, dateAdded: '2022-03-07'}			PASS
RS4	FR-RS-2	{email: 'test@gmail.com', dateAdded: '2022-01-01'}	{success: true, message: 'Success in deleting sleep data'}	{success: true, message: 'Success in deleting sleep data'}	PASS
RS5	FR-RS-2	{email: 'not-found@gmail.com', dateAdded: '2022-01-01'}	{success: false, message: 'Was not able to delete selected sleep data'}	{success: false, message: 'Was not able to delete selected sleep data'}	PASS

Table 3: Rest Section Unit Tests Part 1

Test ID	FR	Inputs	Expected Values	Actual Values	Result
RS6	FR-RS-2	params:{ email: 'example@gmail.com', dateAdded: '2022-01-01'}, body: { sleepHour: 12, bedHour: 11, sleepMinute: 57, bedMinute: 47}	{success: true, message: 'Success in editing sleep data'}	{success: true, message: 'Success in editing sleep data'}	PASS
RS7	FR-RS-2	params: { email: 'notfound@gmail.com', dateAdded: '2022-03-07'}, body: { sleepHour: 12, bedHour: 11, sleepMinute: 57, bedMinute: 47}	{success: false, message: "Was not able to find appropriate sleep data to edit" }	{success: false, message: "Was not able to find appropriate sleep data to edit" }	PASS

Table 4: Rest Section Unit Tests Part 2

6.3 Diet Section

Unit tests for the rest section: https://github.com/BillNguyen1999/REVITALIZE/blob/main/src/SERVER/backend/test/foodLog.test.js.

Test ID	FR	Some Inputs	Some Expected Values	Corresponding Actual Values	Result
DS1	FR-DS-3	{email: 'test@gmail.com', foodDate: 2022-03-08}	{success: true, message: 'Success in getting food log'}	{success: true, message: 'Success in getting food log'}	PASS
DS2	FR-DS-2	{email: 'test@gmail.com', foodDate: 2022-03-08, calories: 1}	{success: true, message: 'Meal successfully added', calories: 1}	{success: true, message: 'Meal successfully added', calories: 1}	PASS
DS3	FR-DS-3	{email: 'test@gmail.com', foodDate: 2022-03-08, foodName: 'name'}	{success: true, message: 'Success in deleting meal'}	{success: true, message: 'Success in deleting meal"}	PASS
DS4	FR-RS-8	{email: 'test@gmail.com', foodDate: 2022-03-08}	{success: true, message: 'Success in updating meal'}	{success: true, message: 'Success in updating meal'}	PASS

Table 5: Diet Section Unit Tests Part 1

6.4 User Section

Unit tests for the User section: https://github.com/BillNguyen1999/REVITALIZE/blob/main/src/SERVER/backend/test/user.test.js.

Test ID	FR	Inputs	Expected Values	Actual Values	Result
US1	FR-SP-1, FR-SP-2, FR-SP-3 and FR-SP-5	{name:'Test Name',email: 'test123@gmail.com', password: '12345'}	Status Code = 201	Status Code = 201	PASS

Table 6: User Section Unit Test

7 Changes Due to Testing

Formal testing did not reveal any necessary changes in terms of module interfacing, decomposition, or internal design. Changes made to code were to address bugs and logical errors revealed by the testing plan. User interface improvements were made throughout the development process in response to feedback from developers and informal testers.

8 Automated Testing

Jest was used to automate the unit tests

9 Trace to Requirements

Table 7: **Traceability Matrix for Login Page Functional Requirements**

		Requirements						
		FR1	FR2	FR3	FR4	FR5	FR6	FR7
Test Cases	FR-LP-1	X						
	FR-LP-2		X					X
	FR-LP-3			X				
	FR-LP-4				X			
	FR-LP-5					X		
	FR-LP-6						X	

Table 8: **Traceability Matrix for Signup Page Functional Requirements**

		Requirements			
		FR8	FR9	FR10	FR11
Test Cases	FR-SP-1	X			X
	FR-SP-2		X	X	
	FR-SP-3		X	X	
	FR-SP-4		X	X	
	FR-SP-5		X	X	

Table 9: **Traceability Matrix for Main Page Functional Requirements**

		Requirements			
		FR12	FR13	FR14	FR15
Test Cases	FR-MP-1	X			
	FR-MP-2		X		
	FR-MP-3			X	
	FR-MP-4				X

Table 10: **Traceability Matrix for Diet Page Functional Requirements**

		Requirements							
		FR16	FR17	FR18	FR19	FR20	FR21	FR22	FR23
Test Cases	FR-DS-1	X							
	FR-DS-2		X						
	FR-DS-3		X						
	FR-DS-4		X						
	FR-DS-5			X	X	X	X	X	X

Table 11: **Traceability Matrix for Workout Page Functional Requirements**

		Requirements		
		FR24	FR25	FR26
Test Cases	FR-WP-1	X		
	FR-WP-2		X	
	FR-WP-3		X	
	FR-WP-4		X	
	FR-WP-5	X		X

Table 12: **Traceability Matrix for Rest Section Functional Requirements**

		Requirements	
		FR27	FR28
Test Cases	FR-RS-1	X	
	FR-RS-2		X
	FR-RS-3		X
	FR-RS-4		X
	FR-RS-5		X
	FR-RS-6		X

Table 13: **Traceability Matrix for Look and Feel Nonfunctional Requirements**

		Requirements	
		LF1	LF2
Test Cases	NFR-LF1	X	
	NFR-LF22		X

Table 14: **Traceability Matrix for Usability and Humanity Nonfunctional Requirements**

		Requirements					
		UH1	UH2	UH3	UH4	UH5	UH6
Test Cases	NFR-UH1	X					
	NFR-UH2		X				
	NFR-UH3			X			
	NFR-UH4				X		
	NFR-UH5					X	

Table 15: **Traceability Matrix for Perfromance Nonfunctional Requirements**

		Requirements				
		PE1	PE2	PE3	PE4	PE5
Test Cases	NFR-PE1	X				
	NFR-PE2		X			
	NFR-PE3			X		
	NFR-PE4				X	
	NFR-PE5					X

Table 16: **Traceability Matrix for Operational Nonfunctional Requirements**

		Requirements	
		OE1	OE2
Test Cases	NFR-OE1	X	
	NFR-OE2		X

Table 17: **Traceability Matrix for Maintainability and Portability Nonfunctional Requirements**

		Requirements		
		MP1	MP2	MP3
Test Cases	NFR-MP1	X		
	NFR-MP2		X	

Table 18: **Traceability Matrix for Security Nonfunctional Requirements**

		Requirements	
		SE1	SE2
Test Cases	NFR-SE1	X	
	NFR-SE2		X

Table 19: **Traceability Matrix for Cultural and Political Nonfunctional Requirements**

		Requirements
		CU1
Test Cases	NFR-CU1	X

10 Trace to Modules

Req.	**Modules**
FR-LP-1	M3
FR-LP-2	M3
FR-LP-3	M3
FR-LP-4	M3
FR-LP-5	M3
FR-LP-6	M3
FR-LP-7	M3
FR-LP-8	M3
FR-LP-9	M3
FR-SP-1	M18
FR-SP-2	M18
FR-SP-3	M18
FR-SP-4	M18
FR-SP-5	M18
FR-MP-1	M1
FR-MP-2	M1
FR-MP-3	M1
FR-MP-4	M1
FR-DS-1	M7
FR-DS-2	M7
FR-DS-3	M7
FR-DS-4	M8
FR-DS-5	M8, M10
FR-DS-6	M11
FR-DS-7	M11
FR-DS-8	M9
FR-WP-1	M14
FR-WP-2	M14
FR-WP-3	M15
FR-WP-4	M15
FR-WP-5	M17
FR-RS-1	M5
FR-RS-2	M6

Table 20: Trace Between Requirements and Modules

11 Code Coverage Metrics

N/A

12 Reflection Appendix

Bill Nguyen: for the vnv plan, it was more formulation rather than implementation, we looked at how we were going to test our project rather than actually doing it. For the vnv report it was more the implementation of our formulation where we wrote actual unit/automated tests and tested our project fully and than compared it to our vnv plan to see what requirements etc. did we satisfy and maybe find things we need to improve on.

Hasan Kibria: In comparison to the vnv plan, the vnv report was more based on practicality an implementation. To complete it fully, there was real code and test cases that had to be thought of an implemented so that they could then be documented in the vnv report. In the vnv plan it was more of an outlook of what we envisioned our testing to look like.

Syed Bokhari: The VNV plan focuses on formulating the testing approach and strategies, while the VNV report is more concerned with the implementation and documentation of the actual testing process. The VNV report involves the creation and execution of test cases, which are then compared to the plan to identify any gaps or areas for improvement. The VNV plan provides a high-level view of the testing process, while the VNV report is a more detailed account of the actual testing activities.

Youssef Dahab: Both the VnV plan and VnV report take inspiration from the functional and non-functional requirements in the SRS document. The VnV plan described how we were going to test our functional and non-functional requirements while the VnV report described the results of performing those tests.

Logan Brown: The VnV plan was more abstract without knowledge of the implementation. The VnV report documents the more refined and directed tests that were performed which could now be completed due to the implementation being more concrete. I have a better idea of how VnV is carried out and the importance of "faking the design process" in the initial stages to make later VnV much easier.

Mahmoud Anklis: The VnV plan is designed to come up with a testing and verification methodology that would ensure that the software application adheres to the functional and non-functional requirements. On the other hand, the VnV report focuses on the actual execution of the tests which requires implementation steps.