Verification and Validation Report: REVITALIZE

Author Name

March 8, 2023

1 Revision History

Date	Version	Notes
March 5th, 2023	Bill Nguyen	Adding Unit Tests for Workout and Rest Section

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test

[symbols, abbreviations or acronyms – you can reference the SRS tables if needed —SS]

Contents

List of Tables

List of Figures

This document ...

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 textbox
Type:	Manual
Initial State:	Login page is displayed with username textbox
Input:	Enter username information in textbox
Output:	Username information entered is displayed in textbox
Expected:	
Result:	PASS

Test #3:	**FR-LP-3**
Description:	Testing that login page displays fillable password textbox
Type:	Manual
Initial State:	Login page is displayed with password textbox
Input:	Enter password information in textbox
Output:	Password information entered is displayed in textbox via hidden text
Expected:	
Result:	PASS

Test #4:	**FR-LP-4**
Description:	Testing that login page displays login button
Type:	Manual
Initial State:	Login page is displayed with login button
Input:	Click the login button
Output:	User logs in after the system checks the validity of the input parameters in the login page
Expected:	Login button is displayed and user logged in successfully
Result:	PASS

Test #5:	**FR-LP-5**
Description:	Testing that login page displays forgot password button
Type:	Manual
Initial State:	Login page is displayed with forgot password button
Input:	Click forgot password button
Output:	Display forgot password screen with textbox to enter email
Expected:	
Result:	PASS

Test #6:	**FR-LP-6**
Description:	Testing that login page displays a stay looged in checkbox
Type:	Manual
Initial State:	Login page is displayed with stay logged in checkbox that is empty
Input:	Click stay logged in checkbox
Output:	Display a check-mark in the stay logged in checkbox if checkbox is empty. Else if checkbox contains check-mark already it will then display an empty checkbox
Expected:	
Result:	PASS

Test #7:	**FR-LP-7**
Description:	Testing that application saves prior login information if stay logged in checkbox is checked
Type:	Manual
Initial State:	Loading stage of REVITALIZE where previous state had stay logged in checkbox checked
Input:	An event that loads REVITALIZE
Output:	Display main page ,, with same data from previous state of main page
Expected:	
Result:	PASS

Test #8:	**FR-LP-8**
Description:	Testing that login page displays sign-up button that redirects to sign-up page
Type:	Manual
Initial State:	Login page is displayed with sign up button
Input:	Click sign up button
Output:	Loads and displays sign up page
Expected:	
Result:	PASS

Test #9:	**FR-LP-9**
Description:	Testing if application checks validity of input parameters in login page
Type:	Manual
Initial State:	Login page is displayed with inputted information in username and pass- word text-boxes
Input:	Click login button
Output:	If failure state,, display an invalid password or username banner. Else if success state,, load and display main page
Expected:	
Result:	PASS

3.2 Signup Page

Test #10:	**FR-SP-1**
Description:	Testing that signup page displays fillable username textbox
Type:	Manual
Initial State:	Signup page is displayed with username textbox
Input:	Enter username information in textbox
Output:	Username information entered is displayed in textbox
Expected:	
Result:	PASS

Test #11:	**FR-SP-2**
Description:	Testing that signup page displays fillable password textbox
Type:	Manual
Initial State:	Signup page is displayed with password textbox
Input:	Enter password information in textbox
Output:	Password information entered is displayed in textbox via hidden text
Expected:	
Result:	PASS

Test #12:	**FR-SP-3**
Description:	Testing that signup page displays fillable email textbox
Type:	Manual
Initial State:	Signup page is displayed with email textbox
Input:	Enter email information in textbox
Output:	Email information entered is displayed in textbox
Expected:	
Result:	PASS

Test #13:	**FR-SP-4**
Description:	Testing that signup page displays signup button
Type:	Manual
Initial State:	Signup page is displayed with signup button
Input:	Click the signup button
Output:	User signs up after the system checks the validity of the input parameters on the signup page
Expected:	
Result:	PASS

Test #14:	**FR-SP-5**
Description:	Testing if application checks validity of input parameters in signup page
Type:	Manual
Initial State:	Signup page is displayed with inputted information in username and password text-boxes
Input:	Click signup button
Output:	If failure state then display an invalid username/password or email banner. Else if success state then load and display login page
Expected:	
Result:	PASS

3.3 Main Page

Test #15:	**FR-MP-1**
Description:	Testing that the application displays a calendar with current date on successful login
Type:	Manual
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 #16:	**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
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 #17:	**FR-MP-3**
Description:	Testing that a back button is displayed on each user interface after a section is selected
Type:	Manual
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 #18:	**FR-MP-4**
Description:	Testing that the application displays the sections Diet,, Exercise,, and Rest on the current calendar day
Type:	Manual
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 #19:	**FR-DS-1**
Description:	Testing that application prompts the user to height,, input dietary,, weight,, calorie information on initial launch of Diet section
Type:	Manual
Initial State:	Diet section is initialized for the first time and an initial information dialog is launched
Input:	An event that loads the diet section for the first time
Output:	A fillable dialog box is launched with height,, dietary information,, weight and calorie information
Expected:	
Result:	PASS

Test #20:	**FR-DS-2**
Description:	Testing that the application saves initial user height,, dietary,, weight,, calorie information
Type:	Manual
Initial State:	Diet section is initialized for the first time and an initial information dialog is launched
Input:	Initial information dialog values are filled
Output:	Initial information values are saved to the database
Expected:	
Result:	PASS

Test #21:	**FR-DS-3**
Description:	Testing that the application initializes with a list of food logged on the current calendar day
Type:	Manual
Initial State:	Section is initialized with a list of food logged for the current calendar day
Input:	An event that loads the rest section
Output:	A list of inputted food is loaded for the current calendar day
Expected:	
Result:	PASS

Test #22:	**FR-DS-4**
Description:	Testing that Diet section displays add food button
Type:	Manual
Initial State:	Diet section is displayed with add food button
Input:	Click add food button
Output:	A user interface is launched that lets the user select between searching for food or adding a custom meal
Expected:	
Result:	PASS

Test #23:	**FR-DS-5**
Description:	Testing that Diet section displays search food button
Type:	Manual
Initial State:	Food adding user interface is displayed with search food button
Input:	Click the search food button
Output:	A recipe criteria user interface is launched that displays a list of modifiable criteria and a search button
Expected:	
Result:	PASS

Test #24:	**FR-DS-6**
Description:	Testing that search food button launches recipe criteria user interface
Type:	Manual
Initial State:	Recipe criteria user interface is launched
Input:	Search criteria is modi ed and search button is clicked
Output:	List of recipes are loaded correctly based on constraints of search criteria
Expected:	
Result:	PASS

Test #25:	**FR-DS-7**
Description:	Testing that recipe search displays correct recipe values based on input constraints
Type:	Manual
Initial State:	Recipe list is loaded based on search constraints
Input:	Add recipe button is clicked
Output:	Selected recipe is added to the list of food logged on the current calendar day
Expected:	
Result:	PASS

Test #26:	**FR-DS-8**
Description:	Testing that add custom meal button adds meal to list of logged food for the current calendar day upon filling necessary recipe information textboxes
Type:	Manual
Initial State:	Food adding interface is displayed with add custom meal button
Input:	Click add custom meal button
Output:	A dialog box is launched that lets the user fill in custom meal information. The meal is added to the food log list of the current calendar day
Expected:	
Result:	PASS

3.5 Workout Section Page

Test #27:	**FR-WS-1**
Description:	Testing that the Workout section initializes with a preset list of exercises on the current calendar day
Type:	Manual
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 #28:	**FR-WS-2**
Description:	Testing that the Workout section has add exercise button
Type:	Manual
Initial State:	Workout section is displayed with add exercise button
Input:	Click add exercise button
Output:	A dialog box is launched that lets the user ll custom exercise information. The exercise is added to the exercise list of the current calendar day
Expected:	
Result:	PASS

Test #29:	**FR-WS-3**
Description:	Testing that the Workout section has delete exercise button
Type:	Manual
Initial State:	Each exercise in the workout section is displayed with a delete exercise button
Input:	Click delete exercise button
Output:	The exercise is deleted from the exercise list of the current calendar day
Expected:	
Result:	PASS

Test #30:	**FR-WS-4**
Description:	Testing that exercises display an edit exercise button that launches the changeable exercise information when clicked
Type:	Manual
Initial State:	Each exercise in the workout section is displayed with an edit exercise button
Input:	click edit exercise button
Output:	A fillable dialog box is launched with information of the exercise. Once the edit exercise button is clicked the dialog box will close and update the exercise information in the list of exercises for the current calendar day
Expected:	
Result:	PASS

Test #31:	**FR-WS-5**
Description:	Testing that the Workout section prompts the user to add repetitions and sets of each exercise logged in the current calendar day
Type:	Manual
Initial State:	Workout section is displayed with list of exercises for current calendar day
Input:	An event that loads the workout section
Output:	If repetition and sets for exercises not logged then dialog box for exercise is launched and the missing repetition and set values are highlighted
Expected:	
Result:	PASS

3.6 Rest Section Page

Test #32:	**FR-RS-1**
Description:	Testing that Rest section launches with sleep statistics of current calendar day
Type:	Manual
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 #33:	**FR-RS-2**
Description:	Testing that user can alter inaccurate sleep data
Type:	Manual
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

4 Nonfunctional Requirements Evaluation

4.1 Usability

4.2 Performance

4.3 etc.

5 Comparison to Existing Implementation

This section will not be appropriate for every project.

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: 'notfound@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

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

9 Trace to Requirements

10 Trace to Modules

11 Code Coverage Metrics

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.