Project Title: System Verification and Validation Plan for REVITALIZE

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

April 5, 2023

1 Revision History

Date	Version	Developer	Notes
October 31st, 2022	0.0	Bill Nguyen	Added Functional Requirements Tests for Login Page
October 31st, 2022	0.0	Bill Nguyen	Added Non Functional Requirements Tests for Look and Feel, Usability and Performance
October 31st, 2022	0.0	Bill Nguyen	Added 3 Questions to Usability Survey
November 1st, 2022	0.0	Youssef Dahab	Added VnV Team, SRS Verification Plan
November 1st, 2022	0.0	Bill Nguyen	Self Reflection
November 1st, 2022	0.0	Youssef Dahab	Added Design Verification Plan
November 1st, 2022	0.0	Syed Bokhari	Added Functional Requirement Tests for Signup Page, Main Page, Diet Section, Workout Section, Rest Section
November 1st, 2022	0.0	Youssef Dahab	Added VnV Verification Plan
November 1st, 2022	0.0	Mahmoud Anklis	Added Non Functional Requirements Tests for Operational, Maintainability and Portability, Security and Cultural and Political
November 1st, 2022	0.0	Mahmoud Anklis	Added Self Reflection as well as Symbols, Abbreviations and Acronyms
November 1st, 2022	0.0	Youssef Dahab	Added Software Validation Plan
November 1st, 2022	0.0	Logan Brown	Added General Information and Self Reflection
November 1st, 2022	0.0	Logan Brown	Added Symbolic Parameters
November 1st, 2022	0.0	Youssef Dahab	Added Implementation Verification Plan, Self Reflection
April 1st, 2023	1.0	Bill Nguyen	Made Suggested Changes to Functional Requirements

Contents

List of Tables

2 Symbols, Abbreviations and Acronyms

symbol	description
REVITALIZE	Name of application
UAT	User Acceptance Testing
UI/UX	User Interface/User Experience
HCI	Human-Computer Interface
MG	Module Guide
MIS	Module Interface Specification
SRS	Software Requirements Specification
VnV	Verification and Validation
FR	Functional Requirement
NFR	Non Functional Requirement
SE	Security 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

This document outlines the Verification & Validation Plan of the REVITALIZE application.

3 General Information

3.1 Summary

This report describes the tests and validation that will be conducted on the REVITALIZE app. The REVITALIZE app is an all-in-one health and wellness app, comprised of 1 main section and 3 major subsections. The main section is a calendar which organizes and documents the contents of the 3 subsections. The 3 subsections are the diet section, workout section, and sleep section.

3.2 Objectives

This document outlines testing plans and testing descriptions in order to determine the following objectives:

- Demonstrate satisfactory usability based on laid out criteria
- Satisfy requirements outlined in SRS
- Build confidence that implementation behaves as intended

3.3 Relevant Documentation

This document references the SRS Author (2019), the Module Guide **?**, and the Module Interface Specification **?**

4 Plan

This section outlines REVITALIZE's verification and validation plan. It begins with identifying the verification and validation team. It then moves to discussing the SRS, design, verification and validation, and implementation verification plans. This section then concludes with REVITALIZE's software validation plan.

4.1 Verification and Validation Team

Table 1: **Verification and Validation Team Member Roles**

Name	Role	Description
Hasan Kibria	~~Front-end Application Tester~~ Back-end and Database Application Tester	~~Run tests to verify and validate front-end design of the application~~ Run tests to verify and validate back-end and database of Diet Section and Login.
Bill Nguyen	Back-end and Database Application Tester	~~Run tests to verify and validate back-end design of the application~~ Run tests to verify and validate back-end and database of Workout , Rest Section and Sign-up.
Syed Bokhari	~~Database Application Tester~~ Front-end and UI / UX Application Tester	~~Run tests to verify and validate database schema, tables, relationships, and data mappings.~~ Run tests to verify and validate front-end design of the Diet Section and Main Menu. Run tests to verify and validate application's accessibility, usability, efficiency, and safety to measure HCI aspects for the Diet Section and Main Menu.
Logan Brown	~~Performance Tester~~ Front-end and UI / UX Application Tester	~~Run tests to verify and validate application speed, responsiveness, and stability.~~ Run tests to verify and validate front-end design of the Workout Section and Calendar. Run tests to verify and validate application's accessibility, usability, efficiency, and safety to measure HCI aspects for the Workout Section and Calendar.
Youssef Dahab	~~UI/UX Tester~~ Front-end and UI / UX Application Tester	Run tests to verify and validate front-end design of the Rest Section, Login and Sign-up. Run tests to verify and validate application's accessibility, usability, efficiency, and safety to measure HCI aspects for Rest Section, Login and Sign-up..
Mahmoud Anklis	System Integration Tester	Run tests to verify and validate front-end, back-end, and database components integration with each other.
Primary & Secondary Stakeholders	UAT	Perform user acceptance testing to verify and validate that the application meets its purpose.
Dr. Smith	Course Instructor	Provide high level project feedback.
Ting-Yu Wu	Course TA	Provide low level project feedback.

~~REVITALIZE team members are aware that they might have to put on multiple hats or perform multiple roles.~~ In the above table, mentions the specific roles and sections each team member is responsible for, but if needed, multiple team members can perform other roles for example back-end testers for the Diet Section can coordinate with front-end testers if problems arise, to make sure both components work in unison.

4.2 SRS Verification Plan

The REVITALIZE team's SRS verification plan is to have the SRS reviewed by classmates, group members, and course TA. Classmates already adhocly reviewed REVITALIZE's SRS and provided their feedback. Furthermore, the course TA will review REVITALIZE's SRS to provide more detailed and structured feedback. REVITALIZE team members will meet together to gather all feedback from classmates, other team members and course TA. The team will then assess the feedback and analyze what can be incorporated. After that, the SRS will be refined and updated as part of the SRS verification process.

4.3 Design Verification Plan

The REVITALIZE team's design verification plan is to have its design reviewed by classmates, group members, and course TA. The team will gather all feedback from classmates, other group members, and the course TA and incorporate it in the design document. After completing the MG and MIS, the team will ensure that their modules match the system requirements in the SRS. Finally, a design walk-through will be conducted by the team to verify the design.

4.4 Verification and Validation Plan Verification Plan

The REVITALIZE teams plan to verify its VnV plan is to have it reviewed by classmates, group members, and course TA. The team will gather all feedback and incorporate it in the verification and validation plan. The REVITALIZE team will also conduct a walk-through after feedback has been incorporated to verify that every functional and nonfunctional requirement in the SRS has at least one test that tests it in the verification and validation plan. Team members will then run the tests in the VnV plan to determine if the system requirements stated in the SRS have been met. This way, the team verifies the completeness of the verification and validation plan in both directions.
The same process will be followed regarding the unit tests. Each module in the MG and MIS will be matched to the tests written in the VnV plan to verify it.

4.5 Implementation Verification Plan

REVITALIZE team members will use both a static and dynamic approach to verify the application's implementation. In terms of dynamic methods, automated unit tests will be performed as part of testing the unit system tests. Furthermore, test cases will be monitored to ensure its up to date such that full testing coverage of the code can be done. This includes edge case testing.

A prototype testing method will also be used. REVITALIZE will not be implemented in one go. Rather, features and functionality will constantly be added throughout the course of the project. Therefore, REVITALIZE will go through numerous prototypes or phases. This prototype testing will act as an ongoing static and dynamic verification of the REVITALIZE application.

Static verification of REVITALIZE's implementation will be done by performing code inspections. Team members will perform code reviews as part of static analysis where each team member will follow the guidelines stated in REVITALIZE's Development Plan. Team members will review each other's code and raise issues to note any concerns. The purpose of conducting static verification of the implementation in this way is to ensure code walk-throughs are completed, concerns are raised, and best implementation practices are followed.

4.6 Automated Testing and Verification Tools

Automated testing and verification tools have been previously stated in sections 6 and 7 of REVITALIZE's Development Plan.

4.7 Software Validation Plan

The REVITALIZE team's software validation plan includes having a review session with primary and secondary stakeholders to check that the SRS requirements document captures the right requirements. After REVITALIZE's initial release, another session with the stakeholders will be held to validate the implementation. The user will use REVITALIZE for the first time, answer team member questions, and give feedback on the application. After that, as per the feedback, the SRS and the implementation may require modifications such as the addition or removal of particular functionality.

5 System Test Description

5.1 Tests for Functional Requirements

Subsections of the requirements will be divided into the events from our SRS, which are Login Page, Sign up Page, Main Page, Diet Section, Workout Section and Rest Section. There will be 1 test per functional requirement, and will follow the same order as functional requirements in SRS (~~ex.~~ eg. FR1 in VnV plan is the test for FR1 in SRS).

5.1.1 Login Page Testing

Testing all functional requirements for login page of REVITALIZE. (Refer to BE1 in SRS)

1. FR-LP-1

   Description: Verifies that the login page loads correctly when requested, and that all necessary components are present on the page

   Control: Manual, Functional, Dynamic

   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

   Test Case Derivation: Request is made to load login page

   How test will be performed: ~~Tester will open REVITALIZE application and login page should be displayed~~ The tester will open the REVITALIZE application. The tester will initiate the event that loads the login page. The tester will verify that the login page is displayed with all necessary components, such as the username and password fields and a "Login" button. If the login page is not displayed as expected, the test will fail. If the login page is displayed as expected, the test will pass.

2. FR-LP-2

   Description: Verifies that a user can successfully log in by providing valid username and password information

   Control: ~~Manual~~ Automatic, Functional, Dynamic

   Initial State: ~~Login page is displayed with username textbox~~ 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

   Test Case Derivation: ~~User can enter information in username textbox~~ User should be able to log in by providing valid username/email and password information

   How test will be performed: ~~Tester will enter information in username textbox and checks if textbox displays what the tester entered.~~ Open the login page and ensure that the username textbox is displayed. Enter valid username and password information in their respective textboxes. Click on the "Login" button. Verify that the user is successfully logged in and redirected to the home page

3. FR-LP-3

Description: Verifies that a user fails to log when providing invalid username/email and password information

Control: 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")

Test Case Derivation: User should not be able to log in by providing invalid username/email and password information

Open the login page and ensure that the username textbox is displayed. Enter invalid username and password information in their respective textboxes. Click on the "Login" button. Verify that the user is unsuccessfully logged in and appropirate error message is returned

4. ~~FR-LP-3~~

   ~~Control: 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~~

   ~~Test Case Derivation: User can enter information in password textbox~~

   ~~How test will be performed: Tester will enter information in password textbox and checks if textbox displays what the tester entered via hidden text.~~

5. ~~FR-LP-4~~

   ~~Control: Manual~~

   ~~Initial State: Login page is displayed with login button~~

   ~~Input: Click login button~~

   ~~Output: Intended events occurs. Refer to FR9~~

   ~~Test Case Derivation: User clicks login button and a request is made based on username and password text-boxes~~

   ~~How test will be performed: Tester will click on login button and check if request is made correctly.~~

6. FR-LP-4

   Description: Verifies that the forget password page loads correctly when requested, and that all necessary components are present on the page

   Control: 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.

   Test Case Derivation: ~~User clicks forgot password button and request is made to display forgot password screen with textbox to enter email~~ By clicking on the "forgot password" button, the system should display the "forgot password" screen with a textbox to enter the email.

   How test will be performed: Tester will click on forgot password button and checks if forgot password screen is displayed with textbox to enter email

7. FR-LP-5

   Description: Verifies the system must allow users to reset their password if they forget it

   Control: 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

   Test Case Derivation: This test case is derived from the functional requirement that the system must allow users to reset their password if they forget it

   How test will be performed: [Enter a valid email address registered with the account. Expected Output: An email will be sent to the user's registered email address with a link to reset their password] [Enter an invalid email address. Expected Output: An error message should be displayed informing the user that the email address is invalid]

8. ~~FR-LP-6~~

   ~~Control: 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~~

~~Test Case Derivation: User clicks on stay logged in checkbox and displays appropriate action~~

~~How test will be performed: Tester will click on checkbox and checks to see if check-mark is displayed if checkbox was empty and if an empty checkbox appears if checkbox contained a check-mark~~

9. ~~FR-LP-7~~

~~Control: 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~~

~~Test Case Derivation: User can load REVITALIZE and main page is displayed with same data as the previous time user opened REVITALIZE main page~~

~~How test will be performed: Tester will check stay logged in checkbox go to main page, leave REVITALIZE, reopen REVITALIZE and check whether same data from main page is the same from the last time tester opened main page~~

10. FR-LP-6

Description: Verifies that the login page displays a sign up button that redirects to the sign up page when clicked

Control: 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.

Test Case Derivation: User can click sign up button which loads and displays sign up page

How test will be performed: ~~Tester will click on sign up button and checks if sign up page is displayed~~ Navigate to the login page. Click on the "Sign up" button. Verify that the system redirects the user to the sign up page

11. ~~FR-LP-9~~

~~Control: Manual~~

~~Initial State: Login page is displayed with inputted information in username and password 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~~

~~Test Case Derivation: User clicks login button and request is made based on username and password, and will proceed to main page only if username and password are valid~~

~~How test will be performed: Tester will click on login button and test for scenarios when login should be successful and when login should fail~~

5.1.2 Signup Page Testing

Testing all functional requirements for sign page of REVITALIZE. (Refer to BE2 in SRS)

1. FR-SP-1

   Description: Verifies that a new user can sign up to REVITALIZE successfully

   Control: ~~Manual~~ Automatic, Functional, Dynamic

   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)

   Test Case Derivation: ~~User can enter information in username textbox~~ The user can successfully sign up for the system with a unique username, password, and email.

   How test will be performed: ~~Tester will enter information in username textbox and checks if textbox displays what the tester entered~~ Tester will navigate to the signup page and verify that the username, password, email and confirm password textboxes are present and blank. Tester will enter a valid username, password, and email in the textboxes and click the "sign up" button. Tester will verify that the user is added to the database and that their username is displayed.

2. FR-SP-2
   Description: Verifies that a new user can not sign up to REVITALIZE when there is invalid information

   Control: ~~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")

Test Case Derivation: ~~User can enter information in password textbox~~ The user unsuccessfully sign ups.

How test will be performed: ~~Tester will enter information in password textbox and checks if textbox displays what the tester entered via hidden text~~ Tester will navigate to the signup page and verify that the username, password, email and confirm password textboxes are present and blank. Tester will enter a invalid username, password, and email in the textboxes and click the "sign up" button. Tester will verify that failure message appears and user is not added into the database.

3. FR-SP-3

Description: Verifies that a new user can not sign up to REVITALIZE when email already exists in database

Control: 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")

Test Case Derivation: The user unsuccessfully sign ups.

Tester will navigate to the signup page and verify that the username, password, email and confirm password textboxes are present and blank. Tester will enter an existing username, password, and email in the textboxes and click the "sign up" button. Tester will verify that failure message appears and user is not added into the database.

4. FR-SP-4

   Description: Verifies that a new user can not sign up to REVITALIZE when there are empty fields

   Control: 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")

   Test Case Derivation: The user unsuccessfully sign ups.

   Tester will navigate to the signup page and verify that the username, password, email and confirm password textboxes are present and blank. Tester not enter a username, password, and email in the textboxes and click the "sign up" button. Tester will verify that failure message appears and user is not added into the database.

5. FR-SP-5

   Description: Verifies that a new user can not sign up to REVITALIZE when password and confirm password fields do not match

   Control: 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")

   Test Case Derivation: The user unsuccessfully sign ups.

   Tester will navigate to the signup page and verify that the username, password, email and confirm password textboxes are present and blank. Tester will enter a username, email and mismatching passwords in the textboxes and click the "sign up" button. Tester will verify that failure message appears and user is not added into the database.

6. ~~FR-SP-3~~

   ~~Control: Manual~~

~~Initial State: signup page is displayed with email textbox~~

~~Input: Enter email information in textbox~~

~~Output: Email information entered is displayed in textbox~~

~~Test Case Derivation: User can enter information in email textbox~~

~~How test will be performed: Tester will enter information in email textbox and checks if textbox displays what the tester entered~~

7. ~~FR-SP-4~~

   ~~Control: Manual~~

   ~~Initial State: Signup page is displayed with signup button~~

   ~~Input: Click signup button~~

   ~~Output: Intended events occurs. Refer to FR14~~

   ~~Test Case Derivation: User clicks signup button and a request is made based on username, password and email text-boxes~~

   ~~How test will be performed: Tester will click on signup button and check if request is made correctly~~

8. ~~FR-SP-5~~

   ~~Control: Manual~~

   ~~Initial State: Signup page is displayed with inputted information in username and password text-boxes~~

   ~~Input: Click signup button~~

   ~~Output: if failure state, display an invalid username, password or email banner, else if success state, load and display login page~~

   ~~Test Case Derivation: User clicks signup button and request is made based on username, password and email, and will proceed to login page only if username, password and email are valid~~

   ~~How test will be performed: Tester will click on signup button and test for scenarios when signup should be successful and when signup should fail~~

5.1.3 Main Page Testing

Testing all functional requirements for main page of REVITALIZE. (Refer to BE3 in SRS)

1. FR-MP-1

   Description: Verifies the system should allow a calendar to be launched to select dates

Control: 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

Test Case Derivation: Request is made to load main page

How test will be performed: Tester will successfully login to the REVITALIZE application and will visually check if the correct calender data is loaded

2. FR-MP-2

   Description: Verifies the system should allow the user to navigate to the previous or next day on the calendar

   Control: 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 calender refreshes the calender information for the next day. Once the preivous day button is clicked, the calender refreshes the calender information for the previous day.

   Test Case Derivation: Request is made to load main page, Diet, Workout, Rest sections and the previous day and next day buttons are clicked

   How test will be performed: Tester will successfully login to the REVITALIZE application and will visually check if the previous day and next day buttons are visible. The tester will click the previous day button and visually check if the calender is updated to the date of the previous day. The tester will click the next day button and visually check if the calender is updated to the date of the next day. The tester will enter the Diet, Workout and Rest sections and will visually check if the previous day and next day buttons are visible. The tester will click the previous day button and visually check if the calender is updated to the date of the previous day. The tester will click the next day button and visually check if the calender is updated to the date of the next day

3. FR-MP-3

   Description: Verifies the system should allow the user to go back to the previous screen or section using a back button

   Control: 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

Test Case Derivation: Request is to leave the main page and the back button is clicked

How test will be performed: Tester will leave the main page by selecting any of the options on the page. The tester will visually check if a back button is visible on every page that is entered through the main page interaction. The tester will click the back button and visually check if the current page is closed and the main page is loaded. The tester will repeat this process with every page that is loaded from clicking an interaction from the main page

4. FR-MP-4

   Description: Verifies the system should allow the user to select and view the Diet, Exercise, and Rest sections for the current calendar day

   Control: 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

   Test Case Derivation: Request is made to load main page and the Diet, Exercise and Rest buttons are clicked

   How test will be performed: Tester will successfully login to the REVITALIZE application and will visually check if the Diet, Exercise and Rest buttons are visible. The tester will click the Diet button and visually check if the Diet interface is loaded. the tester will click the Exercise button and visually check if the Exercise interface is loaded. The tester will click the Rest button and visually check if the Rest interface is loaded

5.1.4 Diet Section Testing

Testing all functional requirements for rest section of REVITALIZE. (Refer to BE4 in SRS)

1. ~~FR-DS-1~~

   ~~Control: 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~~

~~Test Case Derivation: Request is made to enter diet section for the first time~~

~~How test will be performed: Tester will enter rest section for the first time and visually check if the dialog box with correct input information is launched~~

2. ~~FR-DS-2~~

~~Control: 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~~

~~Test Case Derivation: Initial information dialog is filled~~

~~How test will be performed: Tester will fill the initial information dialog after entering the diet section. The tester will visually check the user data in the database to see if the initial information is saved~~

3. FR-DS-1

Description: Verifies that the Diet section must initialize with a list of food logged on the current calendar day

Control: 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]

Test Case Derivation: ~~Request is made to enter rest section~~ Request is made to get/load the food log for email = test@gmail.com and foodDate = 2022-10-25 and request is successful

How test will be performed: ~~Tester will enter the rest section and will visually check if a list of logged food is loaded for the current calender day~~ Ensure that the current calendar day has at least one food item logged in the database. Enter the diet section by selecting it from the REVITALIZE main page or by clicking on the diet section button. Observe the screen and check that a list of the food items logged for the current calendar day is displayed. Verify that the list contains all the food items logged for the current calendar day and user and that the items matches the information in the database

4. FR-DS-2

   Description: Verifies that a new meal can be added to database successfully

   Control: ~~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"

   Test Case Derivation: ~~User clicks add food button~~ User successfully adds meal to database

   How test will be performed: ~~Tester will click on add food button and will visuallly check if the user interface with options for adding a custom meal and searching for food are available.~~ Tester will fill in the necessary fields for adding a meal. Tester will call the appropriate function that adds meal to the database. Tester will then verify the database and see whether the meal was successfully added into the database

5. FR-DS-3

   Description: Verifies that an existing meal can be updated to database successfully

   Control: 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")

Test Case Derivation: The user successfully updates existing meal

Tester will fill in the necessary fields they want for updating a meal. Tester will call the appropriate function that updates meal to the database. Tester will then verify the database and see whether the meal was successfully updated into the database

6. FR-DS-4

   Description: Verifies that an existing meal can be deleted from database successfully

   Control: 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"

   Test Case Derivation: The user successfully deletes existing meal

   Tester checks database for an existing meal data. Tester will fill in the necessary fields that are needed to delete specific meal data. Tester will call the appropriate function that deletes the meal from the database. Tester will then verify the database and see whether the meal was successfully deleted from the database

7. FR-DS-5

   Description: Verifies that searching for a recipe/meal is successful

   Control: 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

   Test Case Derivation: The user successfully searches for a recipe

   Enter the Diet Recipe Search Screen. Fill in the required inputs. Click the search recipe button. Verify to see if a large array of recipes are displayed, matching the information provided in the inputs

8. FR-DS-5}

   Control: Manual

   Initial State: Food adding user interface is displayed with search food button

   Input: Click search food button

   Output: A recipe criteria user interface is launched that displays a list of modifiable criteria and a search button

   Test Case Derivation: User clicks search food button

   How test will be performed: Tester will click on search food exercise button and will visuallly check if a recipe criteria user interface is launched that displays a list of modifiable criteria and a search button

9. FR-DS-6

   Control: Manual

   Initial State: Recipe criteria user interface is launched

   Input: Search criteria is modified and search button is clicked

   Output: List of recipes are loaded correctly based on constraints of search criteria

   Test Case Derivation: User modifies the search criteria and clicks the search button

   How test will be performed: Tester will modify the search criteria and click the search button. The tester will visually check if a recipe list is loaded. The tester will visually check the correctness of the list based on selected constraints.

10. FR-DS-7

    Control: 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

    Test Case Derivation: User selects the recipe from the recipe list and clicks the add recipe button

    How test will be performed: Tester will select a recipe and click the add recipe button. The tester will visually check if the selected recipe has been added to the list of food logged on the current calendar day

11. FR-DS-8

    Control: 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 custom meal information. The meal is added to the food log list of the current calendar day~~

~~Test Case Derivation: User clicks add custom meal button~~

5.1.5 Workout Section Testing

Testing all functional requirements for workout section of REVITALIZE. (Refer to BE5 in SRS)

1. FR-WS-1

   Verifies that a preset list of exercises are initialized when entering workout section

   Control: ~~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

   Test Case Derivation: Request is made to enter workout section

   How test will be performed: Tester will enter the workout section and will visually check if a list of preset exercises are loaded for the current calendar day

2. FR-WS-2

   Description: Verifies that a new workout can be added to database successfully

   Control: 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"

   Test Case Derivation: The user successfully adds workout

   Tester will fill in the necessary fields for adding a workout. Tester will call the appropriate function that adds workout to the database. Tester will then verify the database and see whether the workout was successfully added into the database

3. FR-WS-3

   Description: Verifies that an existing workout can be updated to database successfully

Control: 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")

Test Case Derivation: The user successfully updates existing workout

Tester will fill in the necessary fields they want for updating a workout. Tester will call the appropriate function that updates workout to the database. Tester will then verify the database and see whether the workout was successfully updated into the database

4. FR-WS-4

   Description: Verifies that an existing workout can be deleted from database successfully

   Control: 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 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"

   Test Case Derivation: The user successfully deletes existing workout

   Tester checks database for an existing workout data. Tester will fill in the necessary fields that are needed to delete specific workout data. Tester will call the appropriate function that deletes the workout from the database. Tester will then verify the database and see whether the workout was successfully deleted from the database

5. FR-WS-5

   Description: Verifies that the Workout section must initialize with a list of workouts on the current calendar day

   Control: Automatic, Functional, Dynamic

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")

Test Case Derivation: Request is made to get/load the exercise list for email = test@gmail.com and dateAdded = 2022-10-25 and request is successful

Ensure that the current calendar day has at least one workout data logged in the database. Enter the workout section by selecting it from the REVITALIZE main page or by clicking on the workout section button. Observe the screen and check that a list of the workout items logged for the current calendar day is displayed. Verify that the list contains all the workout items logged for the current calendar day and user, and that the items matches the information in the database

6. ~~FR-WS-2~~

   ~~Control: 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 fill custom exercise information. The exercise is added to the exercise list of the current calendar day~~

   ~~Test Case Derivation: User clicks add exercise button~~

   ~~How test will be performed: Tester will click on add exercise button and will visually check if a fillable dialog box is launched. The tester will fill the dialog box information and click the add exercise button. The tester will visually check if the exercise has been added to the list of exercises for the current calendar day~~

7. ~~FR-WS-3~~

   ~~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~~

~~Test Case Derivation: User clicks delete exercise button~~

~~How test will be performed: Tester will click on delete exercise button and will visually check if the exercise is deleted from the list of exercises for the current calendar day.~~

8. ~~FR-WS-4~~

   ~~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~~

   ~~Test Case Derivation: User clicks edit exercise button~~

   ~~How test will be performed: Tester will click on edit exercise button and will change the information on the fillable dialog box. The tester will click the edit exercise button and will visually check if the information is changed on the exercise list for the current calendar day.~~

9. ~~FR-WS-5~~

   ~~Control: 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, dialog box for exercise is launched and the missing repetition and set values are highlighted~~

   ~~Test Case Derivation: User launches the workout section. User adds an exercise and does not input repetition and set values.~~

5.1.6 Rest Section Testing

Testing all functional requirements for rest section of REVITALIZE. (Refer to BE6 in SRS)

1. FR-RS-1

   Description: Verifies that Rest section launches with sleep statistics of current calendar day

   Control: 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

Test Case Derivation: Request is made to enter rest section

How test will be performed: Tester will enter the rest section and will visually check if sleep statistics are loaded for the current calendar day

2. FR-RS-2

   Description: Verifies that user can alter inaccurate sleep data

   Control: 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

   Test Case Derivation: User alters sleep data

   How test will be performed: Tester will alter the sleep data and will visually check if the new values are correctly displayed in the sleep statistics

3. FR-RS-3

   Description: Verifies that a new sleep data can be added to database successfully

   Control: Automatic, Functional, Dynamic

   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"

   Test Case Derivation: The user successfully adds sleep data

   Tester will fill in the necessary values for adding new sleep data. Tester will call the appropriate function that adds sleep data to the database. Tester will then verify the database and see whether the sleep data was successfully added into the database

4. FR-RS-4

   Description: Verifies that an existing sleep data can be updated to database successfully

   Control: Automatic, Functional, Dynamic

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)

Test Case Derivation: The user successfully updates existing sleep data

Tester will fill in the necessary fields they want for updating a sleep data. Tester will call the appropriate function that updates sleep data to the database. Tester will then verify the database and see whether the sleep data was successfully updated into the database

5. FR-RS-5

Description: Verifies that an existing sleep data can be deleted from database successfully

Control: Automatic, Functional, Dynamic

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"

Test Case Derivation: The user successfully deletes existing sleep

Tester checks database for an existing sleep data. Tester will fill in the necessary fields that are needed to delete specific sleep data. Tester will call the appropriate function that deletes the sleep data from the database. Tester will then verify the database and see whether the sleep data was successfully deleted from the database

6. FR-RS-6

Description: Verifies that the Rest section must initialize with last saved sleep data on the current calendar day

Control: Automatic, Functional, Dynamic

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"

Test Case Derivation: Request is made to get/load the last saved sleep data for email = test@gmail.com and dateAdded = 2022-10-25 and request is successful

Ensure that the current calendar day has one sleep data logged in the database. Enter the rest section by selecting it from the REVITALIZE main page or by clicking on the rest section button. Observe the screen and check that the last saved sleep data for the current calendar day is displayed. Verify that the last saved sleep data for the current calendar day and user, and that the item matches the information in the database

...

5.2 Tests for Nonfunctional Requirements

5.2.1 Look and Feel Testing

1. NFR-LF1

   Type: Dynamic, Functional, Manual

   Initial State: User is using REVITALIZE features

   Input/Condition: REVITALIZE features are in use

   Output/Result: All UI/UX design and elements matches original design and are displayed correctly and neatly

   How test will be performed: All related stakeholders will test application with the focus on neatness and attractiveness of UI/UX design of REVITALIZE and answer question 1 of the Usability Survey. Would need an average rating of MINIMUM_TEST_SCORE or above out of 10 and assess all stakeholders responses to make improvements

2. NFR-LF2

   Type: Static, Manual

   Initial State: A display of all pages in REVITALIZE

   Input/Condition: All displays for all pages in REVITALIZE are at a common point during a user session

Output/Result: All colours are considered appealing, contrasting and non-intrusive

How test will be performed: All related stakeholders will test application with the focus on colour and answer question 2 of the Usability Survey. Would need an average rating of MINIMUM_TEST_SCORE or above out of 10 for each factor and assess all stakeholders responses to make improvements

5.2.2 Usability and Humanity Testing

1. NFR-UH1

   Type: Dynamic, Functional, Manual

   Initial State: User is using REVITALIZE features

   Input/Condition: REVITALIZE features are in use, using one hand/one finger

   Output/Result: REVITALIZE features are displaying correct outputs and results

   How test will be performed: All related stakeholders with varying size hands/fingers can use all aspects of REVITALIZE using one hand/finger and have an average rating of MINIMUM_TEST_SCORE_2 or above for question 3 of the Usability Survey

2. NFR-UH2

   Type: Dynamic, Functional, Manual

   Initial State: Main page of application is displayed

   Input/Condition: User uses main page to access features (Diet, Workout and Rest Section)

   Output/Result: All features take less than MINIMUM_ACCESS_TIME seconds to access

   How test will be performed: Stakeholders will navigate to the Diet, Workout and/or Rest section from the main page in MAXIMUM_ACCESS_TIME seconds or less. 90% of stakeholders need to be able to navigate to any of the sections in MAXIMUM_ACCESS_TIME seconds or less

3. NFR-UH3

   Type: Dynamic, Functional, Manual

   Initial State: REVITALIZE is loaded but not in use

   Input/Condition: Users in targeted demographic will use all features of REVITALIZE

   Output/Result: Results gathered from survey

   How test will be performed: Primary stakeholders such as teenagers and young adults 14 years or older will test application and fill in survey, with the goal of an approval rating of MIN_APPROVAL_RATING or above

4. NFR-UH4

   Type: Dynamic, Functional, Manual

   Initial State: REVITALIZE is loaded but not in use

   Input/Condition: User will use all features of REVITALIZE

   Output/Result: User can use and understand basic/common aspects of all features after 3rd iteration

   How test will be performed: Stakeholders will use all features/aspects of REVITALIZE and MIN_APPROVAL_RATING of stakeholders should be able to use and understand basic/common aspects of all features in 3 iterations or less.

5. NFR-UH5

   Type: Dynamic, Functional, Manual

   Initial State: REVITALIZE is loaded but not in use

   Input/Condition: User will use all features of REVITALIZE

   Output/Result: Results gathered from survey based on consistency of UI aspects such as buttons, drop-downs, words etc.

   How test will be performed: Stakeholders will test application with focus on consistency of UI aspects and fill in survey, with the goal of an approval rating of MIN_APPROVAL_RATING or above

5.2.3 Performance Testing

1. NFR-PE1

   Type: Dynamic, Functional, Manual

   Initial State: REVITALIZE is loaded but not in use

   Input/Condition: All REVITALIZE features are loaded with appropriate data

   Output/Result: loading time of all REVITALIZE features

   How test will be performed: Developers will run performance tests and ensure that all output data loads within 5 seconds or less for MIN_APPROVAL_RATING_2 of all API responses and outputs

2. NFR-PE2

   Type: Dynamic, Functional, Automatic

   Initial State: REVITALIZE is loaded but not in use

   Input/Condition: User uses all features of REVITALIZE where output contain data/numbers

Output/Result: data/numbers of used features in floating points

How test will be performed: Developers will run accuracy tests to ensure output data/numbers are accurate for double precision floating points and pass all test cases

3. NFR-PE3

   Type: Dynamic, Functional, Manual

   Initial State: REVITALIZE is loaded but not in use

   Input/Condition: Multiple (More than MIN_USER_LOAD) users using REVITALIZE

   Output/Result: Performance metrics (ex. loading time, frames per second etc.)

   How test will be performed: MIN_USER_LOAD or more users (does not have to be actual people) use/send requests to REVITALIZE application simultaneously and analyze performance trends based on the number of users

4. NFR-PE4

   Type: Dynamic, Functional, Manual

   Initial State: REVITALIZE is loaded but not in use

   Input/Condition: User will use all features of REVITALIZE

   Output/Result: Storage percentage of data

   How test will be performed: Developers will try to add as much data as possible to user account and application should be able to store MIN_DATA_POINTS or more data points for all users

5.2.4 Operational Testing

1. NFR-OE1

   Type: Dynamic, Functional, Manual

   Initial State: Internet connection is established on the device

   Input/Condition: REVITALIZE application is launched

   Output/Result: REVITALIZE features are all loaded

   How test will be performed: Users will connect to the internet on their devices and launch the application. The application should load all features.

5.2.5 Maintainability and Portability Testing

1. NFR-MP1

   Type: Static, Manual

Initial State: Existing source code

Input/Condition: New source code is added

Output/Result: New source code is commented

How test will be performed: Developers will cross-check each other's comments each time new source code is pushed and before it is merged to the main repository

2. NFR-MP2

   Type: Static, Automatic

   Initial State: Existing source code

   Input/Condition: New source code is added

   Output/Result: New source code that is added adheres to Google JS Style Guide (refer to SRS)

   How test will be performed: The ESLint linter tool will scan the source code and ensure it adheres to the Google JS Style Guide for newly developed source code.

5.2.6 Security Testing

1. NFR-SE1

   Type: Dynamic, Functional, Automatic

   Initial State: REVITALIZE is launched

   Input/Condition: User creates a new account

   Output/Result: REVITALIZE sign-up/account features are loaded and secured

   How test will be performed: Developers will execute security tests to ensure account information is private and secure. 100% of the test cases must pass

2. NFR-SE2

   Type: Dynamic, Functional, Automatic

   Initial State: REVITALIZE sign-up feature is loaded

   Input/Condition: User inputs an email already associated with an existing account

   Output/Result: Message alerts the user that user already exists with this email.

   How test will be performed: Developers will run tests on the Database that stores the users that check to make sure there is no duplication. 100% of the tests must pass

5.2.7 Cultural and Political Testing

1. NFR-CU1

   Type: Dynamic, Functional, Manual

   Initial State: REVITALIZE is launched

   Input/Condition: REVITALIZE features are all loaded

   Output/Result: English is displayed in all REVITALIZE features

   How test will be performed: Developers will test the Front-End to ensure English is the language displayed.

5.3 Traceability Between Test Cases and Requirements

Table 2: **Traceability Matrix for Login Page Functional Requirements**

		Requirements								
		FR1	FR2	FR3	FR4	FR5	FR6	FR7	FR8	FR9
Test Cases	FR-LP-1	X								
	FR-LP-2		X							
	FR-LP-3			X						
	FR-LP-4				X					
	FR-LP-5					X				
	FR-LP-6						X			
	FR-LP-7							X		
	FR-LP-8								X	
	FR-LP-9									X

Table 3: **Traceability Matrix for Signup Page Functional Requirements**

		Requirements				
		FR10	FR11	FR12	FR13	FR14
Test Cases	FR-SP-1	X				
	FR-SP-2		X			
	FR-SP-3			X		
	FR-SP-4				X	
	FR-SP-5					X

Table 4: **Traceability Matrix for Main Page Functional Requirements**

		Requirements				
		FR15	FR16	FR17	FR18	FR30
Test Cases	FR-MP-1	X				
	FR-MP-2		X			X
	FR-MP-3			X		
	FR-MP-4				X	

Table 5: **Traceability Matrix for Diet Page Functional Requirements**

		Requirements									
		FR19	FR20	FR21	FR22	FR23-25	FR26	FR27	FR28	FR29	F30
Test Cases	FR-DS-1	X									
	FR-DS-2		X								
	FR-DS-3			X							
	FR-DS-4				X						
	FR-DS-5					X					
	FR-DS-6						X				
	FR-DS-7							X			
	FR-DS-8								X	X	

Table 6: **Traceability Matrix for Workout Page Functional Requirements**

		Requirements				
		FR31	FR32	FR33	FR34	FR35
Test Cases	FR-WP-1	X				
	FR-WP-2		X			
	FR-WP-3			X		
	FR-WP-4				X	
	FR-WP-5					X

Table 7: **Traceability Matrix for Rest Section Functional Requirements**

		Requirements	
		FR36	FR37
Test Cases	FR-RS-1	X	
	FR-RS-2		X

Table 8: **Traceability Matrix for Look and Feel Nonfunctional Requirements**

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

Table 9: **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 10: **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

Table 11: **Traceability Matrix for Operational Nonfunctional Requirements**

		Requirements	
		OE1	OE2
Test Cases	NFR-OE1	X	

Table 12: **Traceability Matrix for Maintainability and Portability Nonfunctional Requirements**

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

Table 13: **Traceability Matrix for Security Nonfunctional Requirements**

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

Table 14: **Traceability Matrix for Cultural and Political Nonfunctional Requirements**

		Requirements
		CU1
Test Cases	NFR-CU1	X

6 Unit Test Description

6.1 Unit Testing Scope

All modules mentioned below were deemed in the unit testing scope. Calendar, Recipe Results and Recipe Details Modules were out of scope since mostly involves third party data. Main Menu, Label, Search or Add Food and Workout Log Modules were out of scope since mostly modules for UI design and deemed unnecessary to create unit tests. FoodT and ExerciseT Modules were out of scope since these are mainly data types and these data types were tested in the other modules mentioned below.

6.2 Tests for Functional Requirements

6.2.1 Login Module

The upcoming section will feature assessments that pertain to the "Login Module" as defined in the MIS. The selection of these tests was made in consideration of typical user workflows.

1. UT-M3-1

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: A user exists in database

   Input: Create a login request with valid inputs

   Output: Request is successful and will display success message

   Test Case Derivation: User wants to login into REVITALIZE

   How test will be performed: Jest will create login request with valid inputs

2. UT-M3-2

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: A user exists in database

   Input: Create a login request with invalid inputs

   Output: Request fails and will display a failure message

   Test Case Derivation: User wants to login into REVITALIZE, but have invalid inputs

   How test will be performed: Jest will create login request with invalid inputs

6.2.2 Sign-up Module

The upcoming section will feature assessments that pertain to the "Sign-up Module" as defined in the MIS. The selection of these tests was made in consideration of typical user workflows.

1. UT-M18-1

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: N/A

   Input: Create a sign-up request with valid inputs

   Output: Request is successful, will display success message and new user will be added into database

   Test Case Derivation: User wants to sign-up into REVITALIZE

   How test will be performed: Jest will create sign-up request with valid inputs

2. UT-M18-2

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: N/A

   Input: Create a sign-up request with invalid inputs

   Output: Request fails and will display a failure message

   Test Case Derivation: User wants to sign-up into REVITALIZE, but have invalid inputs

   How test will be performed: Jest will create sign-up request with invalid inputs

6.2.3 Container Module

The upcoming section will feature assessments that pertain to the "Container Module" (For the MIS of Sleep) as defined in the MIS. The selection of these tests was made in consideration of typical user workflows.

1. UT-M4-1

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: Sleep data exists in database

   Input: Create a get request for existing sleep data with valid inputs

   Output: Request is successful, will display success message and return desired sleep request

   Test Case Derivation: User wants to get a sleep data

   How test will be performed: Jest will create get request for sleep data with valid inputs

2. UT-M4-2

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: Sleep data exists in database

   Input: Create a get request for existing sleep data with invalid inputs

   Output: Request fails, will display failure message

   Test Case Derivation: User wants to get a sleep data with invalid inputs

   How test will be performed: Jest will create get request for sleep data with invalid inputs

6.2.4 Circular Slider Module

The upcoming section will feature assessments that pertain to the "Circular Slider Module" (For the MIS of Sleep) as defined in the MIS. The selection of these tests was made in consideration of typical user workflows.

1. UT-M6-1

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: N/A

   Input: Create a request to add new sleep data with valid inputs

   Output: Request is successful, will display success message and new sleep data added into database

   Test Case Derivation: User wants to add new sleep data

   How test will be performed: Jest will create a request for sleep data with valid inputs

2. UT-M6-2

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: N/A

   Input: Create a request to add new sleep data with invalid inputs

   Output: Request fails, will display failure message

   Test Case Derivation: User wants to add new sleep data with invalid inputs

   How test will be performed: Jest will create a request for sleep data with invalid inputs

3. UT-M6-3

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: Sleep data exists in database

   Input: Create a request to update sleep data with valid inputs

   Output: Request is successful, will display success message and existing sleep data is updated into database

   Test Case Derivation: User wants to update sleep data with valid inputs

   How test will be performed: Jest will create a request to update sleep data with valid inputs

4. UT-M6-4

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: Sleep data exists in database

   Input: Create a request to update sleep data with invalid inputs

   Output: Request is fails, will display failure message

   Test Case Derivation: User wants to update sleep data with invalid inputs

   How test will be performed: Jest will create a request to update sleep data with invalid inputs

5. UT-M6-5

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: Sleep data exists in database

   Input: Create a request to delete sleep data with valid inputs

   Output: Request is successful, will display success message and existing sleep data is deleted from database

   Test Case Derivation: User wants to delete sleep data with valid inputs

How test will be performed: Jest will create a request to delete sleep data with valid inputs

6. UT-M6-6

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: Sleep data exists in database

   Input: Create a request to delete sleep data with invalid inputs

   Output: Request is fails, will display failure message

   Test Case Derivation: User wants to delete sleep data with invalid inputs

   How test will be performed: Jest will create a request to delete sleep data with invalid inputs

6.2.5 Diet Log Module

The upcoming section will feature assessments that pertain to the "Diet Log Module" as defined in the MIS. The selection of these tests was made in consideration of typical user workflows.

1. UT-M7-1

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: N/A

   Input: Create a request to add new food data with valid inputs

   Output: Request is successful, will display success message and new food data added into database

   Test Case Derivation: User wants to add new food data

   How test will be performed: Jest will create a request for food data with valid inputs

2. UT-M7-2

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: N/A

   Input: Create a request to add new food data with invalid inputs

   Output: Request fails, will display failure message

   Test Case Derivation: User wants to add new food data with invalid inputs

   How test will be performed: Jest will create a request for food data with invalid inputs

3. UT-M7-3

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: Food data exists in database

   Input: Create a request to update food data with valid inputs

   Output: Request is successful, will display success message and existing food data is updated into database

   Test Case Derivation: User wants to update food data with valid inputs

   How test will be performed: Jest will create a request to update food data with valid inputs

4. UT-M7-4

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: food data exists in database

   Input: Create a request to update food data with invalid inputs

   Output: Request is fails, will display failure message

   Test Case Derivation: User wants to update food data with invalid inputs

   How test will be performed: Jest will create a request to update food data with invalid inputs

5. UT-M7-5

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: Food data exists in database

   Input: Create a request to delete food data with valid inputs

   Output: Request is successful, will display success message and existing food data is deleted from database

   Test Case Derivation: User wants to delete food data with valid inputs

   How test will be performed: Jest will create a request to delete food data with valid inputs

6. UT-M7-6

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: food data exists in database

   Input: Create a request to delete food data with invalid inputs

   Output: Request is fails, will display failure message

Test Case Derivation: User wants to delete food data with invalid inputs

How test will be performed: Jest will create a request to delete food data with invalid inputs

7. UT-M7-7

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: Food data exists in database

   Input: Create a get request for existing food data with valid inputs

   Output: Request is successful, will display success message and return desired food request

   Test Case Derivation: User wants to get a food data

   How test will be performed: Jest will create get request for food data with valid inputs

8. UT-M7-8

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: Food data exists in database

   Input: Create a get request for existing food data with invalid inputs

   Output: Request fails, will display failure message

   Test Case Derivation: User wants to get a food data with invalid inputs

   How test will be performed: Jest will create get request for food data with invalid inputs

6.2.6 Custom Meal Module

The upcoming section will feature assessments that pertain to the "Custom Meal Module" as defined in the MIS. The selection of these tests was made in consideration of typical user workflows.

1. UT-M9-1

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: N/A

   Input: Create a request to add new custom meal data with valid inputs

   Output: Request is successful, will display success message and new custom meal data added into database

   Test Case Derivation: User wants to add new custom meal data

   How test will be performed: Jest will create a request for custom meal data with valid inputs

2. UT-M9-2

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: N/A

   Input: Create a request to add new custom meal data with invalid inputs

   Output: Request fails, will display failure message

   Test Case Derivation: User wants to add new custom meal data with invalid inputs

   How test will be performed: Jest will create a request for customer meal with invalid inputs

6.2.7 Search Recipe Module

The upcoming section will feature assessments that pertain to the "Search Recipe Module" as defined in the MIS. The selection of these tests was made in consideration of typical user workflows.

1. UT-M10-1

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: N/A

   Input: Create a request to search recipe with valid inputs

   Output: Request is successful, will display success message

   Test Case Derivation: User wants to search recipe

   How test will be performed: Jest will create a request to search recipe with valid inputs

2. UT-M10-2

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: N/A

   Input: Create a request to search recipe with invalid inputs

   Output: Request fails, will display failure message

   Test Case Derivation: User wants to search recipe

   How test will be performed: Jest will create a request to search recipe with invalid inputs

6.2.8 Workout Display Module

The upcoming section will feature assessments that pertain to the "Workout Display Module" as defined in the MIS. The selection of these tests was made in consideration of typical user workflows.

1. UT-M14-1

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: Workout data exists in database

   Input: Create a get request for existing workout data with valid inputs

   Output: Request is successful, will display success message and return desired workout request

   Test Case Derivation: User wants to get a workout data

   How test will be performed: Jest will create get request for workout data with valid inputs

2. UT-M14-2

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: Workout data exists in database

   Input: Create a get request for existing workout data with invalid inputs

   Output: Request fails, will display failure message

   Test Case Derivation: User wants to get a workout data with invalid inputs

   How test will be performed: Jest will create get request for workout data with invalid inputs

6.2.9 Workout Edit Module

The upcoming section will feature assessments that pertain to the "Workout Edit Module" as defined in the MIS. The selection of these tests was made in consideration of typical user workflows.

1. UT-M15-1

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: N/A

   Input: Create a request to add new workout data with valid inputs

   Output: Request is successful, will display success message and new workout data added into database

Test Case Derivation: User wants to add new workout data

How test will be performed: Jest will create a request for workout data with valid inputs

2. UT-M15-2

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: N/A

   Input: Create a request to add new workout data with invalid inputs

   Output: Request fails, will display failure message

   Test Case Derivation: User wants to add new workout data with invalid inputs

   How test will be performed: Jest will create a request for workout data with invalid inputs

3. UT-M15-3

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: Workout data exists in database

   Input: Create a request to update workout data with valid inputs

   Output: Request is successful, will display success message and existing workout data is updated into database

   Test Case Derivation: User wants to update workout data with valid inputs

   How test will be performed: Jest will create a request to update workout data with valid inputs

4. UT-M15-4

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: Workout data exists in database

   Input: Create a request to update workout data with invalid inputs

   Output: Request is fails, will display failure message

   Test Case Derivation: User wants to update workout data with invalid inputs

   How test will be performed: Jest will create a request to update workout data with invalid inputs

5. UT-M15-5

   Type: Functional, Dynamic, Automatic, Unit

   Initial State: Workout data exists in database

Input: Create a request to delete workout data with valid inputs

Output: Request is successful, will display success message and existing workout data is deleted from database

Test Case Derivation: User wants to delete workout data with valid inputs

How test will be performed: Jest will create a request to delete workout data with valid inputs

6. UT-M15-6

Type: Functional, Dynamic, Automatic, Unit

Initial State: Workout data exists in database

Input: Create a request to delete workout data with invalid inputs

Output: Request is fails, will display failure message

Test Case Derivation: User wants to delete workout data with invalid inputs

How test will be performed: Jest will create a request to delete workout data with invalid inputs

6.3 Traceability Between Test Cases and Modules

Test Cases	Modules
UT-M3-1	M3
UT-M3-2	M3
UT-M18-1	M18
UT-M18-2	M18
UT-M4-1	M4
UT-M4-2	M4
UT-M6-1	M6
UT-M6-2	M6
UT-M6-3	M6
UT-M6-4	M6
UT-M6-5	M6
UT-M6-6	M6
UT-M7-1	M7
UT-M7-2	M7
UT-M7-3	M7
UT-M7-4	M7
UT-M7-5	M7
UT-M7-6	M7
UT-M7-7	M7
UT-M7-8	M7
UT-M9-1	M9
UT-M9-2	M9
UT-M10-1	M10
UT-M10-2	M10
UT-M14-1	M14
UT-M14-2	M14
UT-M15-1	M15
UT-M15-2	M15
UT-M15-3	M15
UT-M15-4	M15
UT-M15-5	M15
UT-M15-6	M15

Table 15: Trace Between Test Cases and Modules

References

Author Author. System requirements specification. https://github.com/..., 2019.

7 Appendix

7.1 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

7.2 Usability Survey Questions

1. How would you rate overall neatness of UI/UX design of REVITALIZE out of 10? What are elements you like related to neatness? What are elements you would improve related to neatness?

2. Are colours used in REVITALIZE non-intrusive, appealing and/or contrasting? Rate each factor of non-intrusive, appealing and contrasting out of 10? What are elements you like that elevate these factors? What are elements you would improve to elevate these factors?

3. How would you rate the overall ability to use REVITALIZE with only one hand/one finger out of 10? What are elements that help with this? What are elements you would improve?

8 Reflection Appendix

8.1 Knowledge and Skills Needed

Bill Nguyen: My primary role for VnV plan is to be a back-end developer tester, so some skills needed would be how to write proper and effective unit tests for back-end implementation of project, integration tests that tests the project end to end and with the front-end, and also maybe some tests for performance such as load, stress test etc. for back-end code.

Syed Bokhari: My primary role for VnV plan is to be a Database Application tester. Some skills needed would be how to configure API testing environments and database query accuracy. I must know how to prepare the testing environment, writeunit tests for SQL queries and configure test data to ensure the accuracy of the results.

Mahmoud Anklis: My primary role for the VnV plan is to be a System Integration tester. Some of the skills needed would be how to setup system tests that test the integration of all modules. I would also have to know how to write scripts that test the interfaces of each component and ensure the entire system is behaving properly.

Hasan Kibria: My primary role for the VnV plan is to be a Frontend Tester. Some of the skills needed would be knowledge of frontend testing frameworks for our language of choice, as well as knowing how to properly mock unnecessary resources (i.e. the backend) to truly test targeted frontend functionality in the manner we aim to.

Logan Brown: My primary role for the VnV plan is to be a Performance Tester. First I would need to know what tools are available for monitoring the performance of our app and then the skill I would need to learn is how to properly analyze the measured metrics. I would also have to know how to simulate performance on a variety of devices with different capabilities.

Youssef Dahab: To contribute to this VnV, I needed to learn how to design a verification plan for the SRS, Design, VnV, and implementation. Furthermore, as a UI/UX tester, I will have to know how to evaulate user experience and have a good understanding of human computer interfaces.

8.2 How Knowledge and Skills Will Be Acquired

Bill Nguyen: Knowledge and skills will be acquired by researching techniques on how to write effective unit tests for the back-end for example using the "arrange act assert" and potentially setting minimum thresholds for code/branch coverage to improve consistency and organization of tests. For integration and performance tests also perform research to find effective ways to test back-end with all aspects of project and applying it often.

Syed Bokhari: The required knowledge and skills will be acquired by setting up practice environments via virtual machines and testing the API implementation with smaller scale applications. Once I can confirm the accuracy of the database queries and API requests with unit tests, I can configure a larger portion of the code to ingest the data from the API and store in the database.

Mahmoud Anklis: The knowledge and skills required for system integration testing will be acquired by understanding the software architecture and how each component communicates with the other. I will also have to research methods on how to build scripts the can test the entire application's components (Front-End, Back-End, Database etc), where the script is able to pin point the component that led to a failure if there is a failure in when testing.

Hasan Kibria: I hope to acquire this knowledge by reading online docs, looking at code examples, and trying my best to re-emulate tests I find on the internet to fit our specific needs. Once I find myself to be familiar with the technologies and infrastructure at hand, then I would continue to learn by diving deeper into testing principles and following through the targeted test implementation.

Logan Brown: To learn the skills I outlined above I would need to research different performance measuring tools and what they measure using online documentation. Most likely

I could use a tool that is a plugin for the development platform we will be using. I will also learn these skills by measuring the performance of simulated versions of the application, making changes to the implementation and seeing the effect these changes have on the performance.

Youssef Dahab: I had two options to acquire these skills and I used both of them. Firstly, I referenced the lecture notes and material to be able to complete the VnV plan. Secondly, I used the internet to gain further knowledge on topics that I needed more clarification on with regards to design, software validation, and implementation. I will also have to review material from SFWRENG 4HC3 to be able to use HCI concepts to better test REVITALIZE's UI and UX.