

Test Plan: BitOHealth

The 6 Bits:

Angel Cueva

Dante Secundino

Emily Sahyoun

Jacob Munoz

Rami Iskender

Sebastian Vasquez (Team Leader)

10/30/2021

Table of Contents

Document Purpose	3
1. Medication/Health Recorder:	5
2. Reminders:	15
3- Health Locator	22
4- Medication Lookup	25
5- Weight Management/Calorie Counter:	35
6. Track Records	54
7- Diet Recommendations	59
8- Hot Topics	64
9. Registration:	66
10 Login:	69
11. Authorization/Security	73
Cost, Effort, and Resources	74
Summary of Test Features and Resources:	74
Summary of Efforts and Cost:	75

Document Purpose

The Test Plan for the BitOHealth progressive web application ensures that essential testing of every feature is defined and specified clearly. This document is to be developed based on the requirements the developer team created and the client's feedback. This document is intended to include all relevant information regarding the testing of the web application's features and should be referred to in relation with the project plan and network diagram.

a. Unit Testing

- This is the most basic form of testing. We will isolate each feature and subfeature of the product to test. We will test for the correct output of all functionalities. It is essential to test like this before the product as a whole because a system needs all working parts before working together as a system. We will be able to automate this type of testing.

b. Negative Testing

- The use of negative testing will guarantee that failure scenarios are handled properly within each feature of our code. Essentially, invalid user input will be entered into the system. The expected outcome during this testing is for all appropriate error messages and warnings to appear. However, if error handling was not implemented properly, it could potentially cause the entire system to crash. The use of this testing will identify these issues, allowing us to fix them prior to launch.

c. API Testing

- Throughout our product, various APIs will be used to access information vital to feature success. We will be able to validate that information provided by the APIs is secure, reliant, performant, and functional. If an API function fails our tests, it can be addressed appropriately

d. UI Testing

- Performing user interface testing is important to the overall experience of our users. It will test any controls the user can interact with to ensure all are working properly and that visual elements are appearing as implemented. This testing will guarantee that the UI is optimal for user experience and product flow.

e. Integration Testing

- As mentioned in unit testing, a system needs all of its parts to work together to make a whole. Integration testing would test the system on how it works as a whole rather than individual parts. The expected

outcome is that all parts of the product communicate accurately and efficiently, from user input to being stored in the database. If a particular area of our product fails this testing, we can identify where the communication between parts failed so it can be fixed and addressed.

f. Regression Testing

- This type of testing is necessary for our product because of its scalability support. It ensures that features are working as expected despite feature changes or additions. It will allow us to have a consistent product while still expanding.

g. Security Testing

- Security testing entails assessing sufficient product protection of sensitive data like user passwords. Throughout testing, we can identify system vulnerabilities and set various methods to eliminate them.

h. Stress Testing

- Various stress tests will allow us to confidently specify the highest load our product can sustain. Essentially, our product features will be pushed to their limits until they fail. We want to be able to guarantee that the product can still operate in some sort of capacity like supplying error messages despite being pushed to the limit.

i. Performance Testing

- A performance test will test our system's speed, stability, and scalability. We want to ensure that the system executes and outputs within our specified time constraints. Also, we must identify the maximum load each unique feature can handle. Lastly, that the system is consistent under various loads within our maximum load.

j. System/End-to-End Testing

- This type of testing will test the flow of our product from beginning to end. It is the replication of user behaviors and common scenarios to make sure the product works as a whole. If the test results in a failure, the specific component that caused the failure can be fixed. It is also important to check user flow to guarantee that each feature is easy to get to.

Test Scope

- The testing mentioned above will be very beneficial to the success of our product. Realistically, all of this testing cannot be completed due to resource constraints. Many of our limited available resources should be focused on testing set functional and nonfunctional requirements to ensure the best user experience. In-scope testing refers to a test that will be completed prior to the launch of our product. Out-of-scope testing is the opposite, so testing that is not planned to be completed. Most of the testing that was defined will be in-scope, but regression testing is classified as out-of-scope. Scalability is an important

aspect that our product holds, but our focus is to prioritize pushing core requirements that make BitOHealth unique before focussing on different features. This is summarized below.

- In-Scope
 - Unit Testing
 - Negative Testing
 - API Testing
 - UI Testing
 - Integration Testing
 - Security Testing
 - Stress Testing
 - Performance Testing
 - End-to-End Testing
- Out-of-Scope
 - Regression Testing

Note:

For a success scenario to occur, all its cases must pass.

For a success scenario to fail, at least one of its cases must fail.

1. Medication/Health Recorder:

Type of Testing:

- Unit, Negative, UI, Integration, Security, Stress, Performance, and End-to-End Testing.

Assumptions:

- User is logged in.
- User is at the Medication/Health Recorder View.
- User information is retrieved from the Profile.

1.1 User selects Create Record Option

User selects a category and inputs a **valid** name for the record.

Success Scenarios:

- Create Record submission request is successful, if the user has not made a record in the last 30 seconds or is not on the record limit.
- Record is created and saved under the selected category.
- Timing of record creation is saved and associated with the record.

- Medication/Health Recorder View is updated to display newly created record.
- User is redirected to the Record View.

Failure Scenarios:

- Create Record submission request is unsuccessful.
- Create Record submission request is successful if the user made a record in last 30 seconds
- Record limit is exceeded
- Record is not created.
- Record is not saved under the selected category.
- Medication/Health Recorder View is not updated to display the newly created record.
- User is not redirected to the Record View.

User selects a category and inputs an **invalid** name for the record.

Success Scenarios:

- Create Record submission is unsuccessful.
- An error message is displayed stating to follow the proper guidelines of the specific invalid input
- Record is not created.

Failure Scenarios:

- Create record submission is successful.
- An error message is not displayed.
- Record is created.
- Medication/Health Recorder View is updated to display the newly created record.
- User is redirected to the Record View.

Testing plan:

- Execution:
 - User selects Create Record
- Test Cases
 - Test 1: Input valid name
 - Unit Testing:
 - Input:

- "Record 1" is inputted for name
 - Success Case:
 - record 1 is made.
 - Failure Cases:
 - Record 1 is not made.
 - Record 1 is made against the record limit
 - Execution Time: 1 minute
- Test 2: Test 30 second upload limit
 - Unit Testing:
 - Input:
 - "Record 1" is inputted for name
 - "Record 2" is made within 30 seconds of "record 1"
 - Success Case:
 - record 1 is made.
 - Record 2 is not made and an error is given telling the user a record was made in the last 30 seconds.
 - Failure Cases:
 - Record 1 is not made.
 - Record 2 is made against the 30 second block limit
 - Execution Time: 2 minutes
 - Test 3: Test Record Limit
 - Unit testing
 - Input:
 - "Record max" is inputted for name while at record limit
 - Success:
 - Record max isn't made and user is informed they're on limit
 - Failure:
 - Record max is made
 - Execution Time: 1 minute

- [illegible]

1.2 User selects Input Data on Record Option:

Assumptions:

- User must be logged in.
- Use must be in the Record View.

Success Scenarios:

- Input Data on Record request is successful.
- User is able to enter text to the record.

- User is able to upload a maximum of two records per day.
- User is able to set the date and time for the record.
- User is able to save changes made to the record.
- Record is saved under the selected category.

Failure Scenario:

- Input Data on Record request is unsuccessful.
- User is unable to enter text to the record.
- User is unable to upload files to the record.
- User is unable to set the date and time.
- User is unable to save changes made to the record.
- Record is saved under a different category.

User uploads **valid** File:

Success Scenarios:

- User's file is successfully uploaded.
- User's file is stored in the Record View.
- User can select and view contents of the file.
- A confirmation message is displayed to the user.

Failure Scenarios:

- User's file is not uploaded.
- User's file is not stored in the Record View.
- User can't select and display the file.
- No confirmation message is displayed to the user.

User uploads **invalid** File:

Success Scenarios:

- File upload request is unsuccessful.
- An error message is displayed stating to follow the proper guidelines.
- User is given the option to select a different file to upload.

Failure Scenarios:

- File upload request is successful.
- No error message is displayed to the user.

- Incorrect error message is displayed to the user.
- User is not given the option to select a different file to upload.

User reaches maximum file uploads for the day:

Success Scenarios:

- Once the last file is successfully uploaded and stored, a message is displayed to the user stating the limit has been reached.
- Upload file option will no longer be visible until the next day.

Failure Scenarios:

- No message is displayed to the user regarding reaching the maximum file upload limit.
- Upload file option is still visible to the User.
- User is able to successfully upload more files exceeding the file upload limit.
- File is stored and is displayed in the Weight Management View.

Test Cases:

Execution: User presses input file

- Test 1: valid file
 - Integration Testing
 - Input:
 - Valid.jpg is uploaded
 - Success:
 - Valid.jpg is on the record after the database is checked to make sure the limit isn't breach. Valid.jpg is also visible to user
 - Failure:
 - Valid.jpg does not show up on the record, even without limit breach or valid.jpg is not visible to user.
 - Execution time: 1 minute
- Test 2: User inputs invalid file
 - Negative Testing

- Input:
 - Invalid.jpg is uploaded
- Success:
 - Invalid.jpg is not uploaded
- Failure:
 - Invalid.jpg is accepted
- Execution time: 1 minute
- Test 3: Test 2 per day limit
 - Negative Testing
 - Input:
 - "inp1.jpg" , "inp2.jpg" , "inp3.jpg"
 - Success:
 - Inp1, inp2 are accepted, inp3 is rejected. Message is then shown informing user of limit
 - Failure
 - Inp3.jpg is accepted
 - Execution time: 3 minutes

1.3 User selects the Edit Record Option:

Assumptions:

- User is logged in.
- User goes to Medication/Health Recorder View.
- User picks a record and edits record.

Success Scenarios:

- User is able to edit the title and/or category of the record
- User is be able to edit the text, files, and date/time of the record.
- A confirmation message to save changes is displayed to the user.
- Record is updated with the new content and saved under the selected category.
- User will be given error if over storage limit

Failure Scenarios:

- User is unable to edit the title and/or category of the record.
- User is unable to edit the text, files, and date/time of the record.
- No confirmation message is displayed to the user.
- Record is not updated with the new content.
- Record changes are not saved.
- User successfully saves even if over limit

Test Cases:

- Test 1: Save
 - Integration test
 - Input:
 - “Testing record” and save
 - Success Case:
 - User is informed it is saved
 - Failure Case:
 - User not informed it is saved
 - Execution Time: 1 min
- Test 2: Database storage
 - Integration Testing
 - Input:
 - “Testing database” and save
 - Success Case:
 - The save is on the database
 - Fail Case:
 - Case is not on database
 - Execution Time: 1 min
- Test 3: Capacity Test:
 - Negative Test
 - Input
 - “Testing Limit” when on profile storage limit
 - Success Case
 - Data will not be saved and error will be given

- Fail Case
 - Data is saved and user goes over limit.
- Execution Time: 2 minutes.

1.4 User selects the View Record Option:

Assumptions:

- User is logged on.
- User is at the Record List View.
- At least one record exists.

User selects the record to be viewed.

Successful Scenarios:

- View Record request is successful.
- User is redirected from the Records List View to the Record View.
- User is able to view the contents of the Record.
- User is able to edit the contents of the Record.

Failure Scenarios:

- View Record request is unsuccessful.
- User is redirected to the wrong Records View.
- User is unable to view the contents of the Record.
- User is unable to edit the contents of the Record.

Test Cases:

- "Record 1" and "Record 2" will have been made.
- Test 1: Records are visible:
 - Integration test
 - Input:
 - No input required
 - Success Case:
 - User is able to see "Record 1" and "Record 2"
 - Fail Case:
 - User is unable to see "Record 1" or "Record 2"
 - Execution Time: 1 min
- Test 2: Access Record:

- Unit Test
- Input:
 - User chooses “record 2”
- Success case:
 - System retrieves record 2 and shows it to user with option to edit
- Fail case:
 - System fails to retrieve record 2
- Execution Time: 1 min

1.5 User selects the Delete Record Option:

- User is logged in.
- User is at the Medication/Health Recorder View.
- At least one record must exist.

Success Scenarios:

- Delete Record request is successful.
- A confirmation message is displayed to the user verifying whether they want to delete the record or not.
- Record is deleted.
- Medication/Health Recorder View is updated and reflects the removal of the meal item.

Failure Scenarios:

- Delete Record request is unsuccessful.
- No confirmation message is displayed to the user.
- Record is not deleted.
- The incorrect record is deleted.
- Medical/Health Recorder View does not reflect the removal of the record.

- Tests Cases

- Test 1- Delete A file:
 - Unit Test
 - Input:
 - Delete Record 1
 - Success Case:
 - User is told record 1 is deleted, record 1 no longer visible, record 1 is off the database
 - Fail Case:
 - User isn't informed record 1 is deleted, user can see Record 1 after deletion, record 1 appears on database.
 - Time Execution: 1 minute.

2. Reminders:

Type of Testing:

- Unit, UI, Negative, Integration, and Regression Testing.

Assumptions:

- User is logged in.
- User is at the Reminder List View.
- The user's current Reminder List has less than 99 reminders.

2.1 User selects the Create a Reminder Option.

User inputs a **valid** Name, Description, and selects a Date/Time for the reminder.

Success Scenarios:

- Create Reminder submission request is successful.
- Reminder is created and saved to the Reminder List.
- Reminder List is updated to display the newly created reminder.

Failure Scenarios:

- Create Reminder submission request is unsuccessful.
- Reminder is not created.
- Reminder is not saved to the Reminder List.
- Reminder List does not display the newly created reminder.

User inputs an **invalid** Name, Description, and selects a Date/Time for the reminder, as well as frequency.

Success Scenarios:

- Create Reminder submission is unsuccessful.
- An error message is displayed stating to follow the proper guidelines of the specific invalid input
- Reminder is not created.

Failure Scenarios:

- [illegible]

me1name1name1name1name1name1name1name1name1n
ame1name1name1name1”

- Description = “name1” 200 times
- Success Case:
 - An error is given on both to the user
- Failure Case:
 - The system accepts at least one of the inputs
- Execution time : 2 minutes

2.2 User selects Edit Reminder Option

Assumptions:

- User must be logged in.
- User must in the Reminders List View.
- At least one reminder must exist.
- At least one record must exist.

Success Scenarios:

- User is able to edit the title, description, and select a different date/time for the reminder.
- A confirmation message to save changes is displayed to the user.
- Reminder is updated with the new content and saved.
- User is notified of the reminder by the newly selected date/time.

Failure Scenarios:

- User is unable to edit the title, description, and select a different date/time for the reminder.
- No confirmation message is displayed to the user.
- Reminder changes are not saved.
- User is still notified of the reminder by the outdated date/time.

- Test Cases

- Integration

- Test 1: Reminder is saved

- Integration

- “Name1” changed to “name2”

- Success Case:
 - Name is successfully changed and stored in the database. User can see the change
- Fail Case:
 - Name isn't successfully changed, the name isn't stored in the database, and the user can't see the new name.
- Execution Time: 1 minute

2.3 User selects the View Reminder Option:

Assumptions:

- User is logged on.
- User is at the Reminders List View.
- At least one reminder exists.

User selects the reminder to be viewed.

Successful Scenarios:

- View Reminder request is successful.
- User is redirected from the Reminders List View to the Reminder View.
- User is able to view the contents of the Reminder.
- User is able to edit the contents of the Reminder.

Failure Scenarios:

- View Reminder request is unsuccessful.
- User is redirected to the wrong Reminder View.
- User is unable to view the contents of the Reminder
- User is unable to edit the contents of the Reminder.

- Test cases
 - "Remind1", "Remind2" is made
 - Test 1-view reminders:
 - Integration Test
 - Success Case:

- Both Reminders are visible
- Fail Case:
 - At least one reminder isn't visible
- Execution time: 1 minute
- Test 2- select reminder
 - Integration Test
 - Input:
 - "Remind1" is pressed
 - Success Case
 - Remind1 is retrieved from database and opened
 - Failure Case
 - Remind1 isn't selected from database or opened.
 - Execution time: 1 minute

2.4 User selects the Delete Reminder Option:

- User is logged in.
- User is at the Reminder List View.
- At least one reminder must exist.

Success Scenarios:

- Delete Reminder request is successful.
- A confirmation message is displayed to the user verifying whether they want to delete the reminder or not.
- Reminder is deleted.
- Reminder List View is updated and reflects the removal of the reminder.

Failure Scenarios:

- Delete Reminder request is unsuccessful.
- No confirmation message is displayed to the user.
- Reminder is not deleted.
- The incorrect reminder is deleted.

- Reminder List View does not reflect the removal of the reminder
- Test cases
 - “Remind1” is made
 - Test 1: test deletion
 - Integration Test
 - Input:
 - Delete Remind1
 - Success:
 - Remind1 is deleted off the database and no longer available for user
 - Failure
 - Remind1 is not deleted
 - Execution Time: 1 minute

2.5 User selects the Export Reminder Option

Assumptions:

- User is logged in.
- The reminder must exist.
- User is at the Reminder View.

Successful Scenarios:

- Export Reminder request is successful.
- A confirmation message is displayed to the user verifying whether they want to export the reminder or not.
- When exporting from a mobile device, the reminder is exported as an iCal and saved to the user's Files to manually be added to 3rd party services.
- When exporting from a desktop, the reminder is exported as a iCal and the user chooses where to save it.
- Reminder is successfully exported to their selected destination.
- Exporting the selected reminder does not affect the Reminder List.
- A message is displayed to the user stating that the reminder has been exported.

Failure Scenarios:

- Export Reminder request is unsuccessful.
- No confirmation message is displayed to the user verifying whether they want to export the reminder or not.
- No export of reminder occurs.
- Correct reminder is exported to the incorrect destination.
- Incorrect reminder is exported to the user's selected destination.
- Exporting a reminder affected the Reminder List.
- No message is displayed to the user stating the status of the export.

Test Cases

"Reminder 1" is made

- Test 1: Export Reminder is selected:
 - Unit testing
 - Input:
 - Export Reminder to desktop
 - Success case:
 - Reminder is successfully exported and found in desktop
 - Fail Case
 - Reminder is not exported or reminder is exported and is affected in the database.
 - Execution Time: 1 minute.

2.6 Opt-in Reminder Notifications Option

Assumptions:

- User is logged in.
- User is in the Reminder List View.
- A reminder exists

User selects the reminder preference (email or calendar).

Success Scenarios:

- The Opt-in reminder notification request is successful.
- A message is displayed to the user to confirm their selected preference.
- The user is notified of the reminder at their selected timing via their reminder preference.

Failure Scenarios:

- The Opt-in reminder notification request is not successful.
- No message is displayed to the user.
- The user is not notified at the correct timing or/and via their reminder preference.

Test cases

- "Reminder 1" exists
- "Reminder 1" exists with reminder occurring in 5 minutes
- Test Case 1 successfully reminded:
 - Unit Testing
 - Success case:
 - A reminder message displays showing the reminder's 1 description and name
 - Failure case:
 - Reminder fails to show or is missing information

3- Health Locator

Types of Testing:

- Unit, Negative, API, UI, Integration, Security, Stress, and End-to-End Testing.

Assumptions:

- User is logged in
- User is in Health Locator View

3.1 User searches by Zip Code

User inputs **valid** Zip Code and selects a Category for health type.:

Success Scenarios:

- Search request submission is successful.
- API request is made and returns correct results to the system.
- Health Locator View is updated to display the search results.
- User is notified when no results are generated.
- User is able to select a location item from the search list to view its details.

Failure Scenarios:

- Search request submission is unsuccessful.

- Search is not executed and results are not generated.
- No API request is made.
- API request is made and returns incorrect results to the system.
- Search displays wrong location items that do not correspond to the user's input.
- User is unable to select a location item from the search list to view it.

User inputs **invalid** Zip Code and selects a Category for health type:

Success Scenarios:

- User is unable to submit the Health Location Search request.
- No API request is made.
- An error message is displayed stating to follow the proper guidelines of the specific invalid input
- User is given the option to re-input the incorrect fields.

Failure Scenarios:

- User is able to submit the Health Location Search request.
- API request is made.
- Search is executed and results are generated.
- No error message is displayed to the user.
- Wrong error message is displayed to the user.
- User is not given the option to re-input the incorrect fields.

Search Results:

Success Scenarios

- Locations are sorted by proximity.
- User can interact with list of locations.
- User can select a location item to view additional information.
- User is notified when no locations match their request.

Failure Scenarios

- Locations are not sorted appropriately
- User is unable to interact with the results generated by the search.
- User is unable to view additional information on the selected item.
- User is not notified when no locations match their request.
-

User chooses to view a certain locations information:

Success Scenarios:

- The following information is displayed:
 - Location hours
 - Website

- Phone number

Failure Scenarios:

- No information is displayed.
- Incorrect information is displayed.

Testing Plan

Execution:

- User selects the Search Health Locator Option.
- User selects a category for health type.

Test Cases:

Integration Test

Test 1: User inputs valid Zip Code.

Input:

- 90840 is inputted for Zip Code.

Success Case:

- Search is conducted and displays results that correspond to the user's input.

Failure Case:

- Search is not conducted and does not display results that correspond to the user's input.

Execution Time: 1 minute.

Test 2: User inputs invalid Zip Code.

Unit Testing

Input Possibilities:

- 90342232 is inputted for Zip Code.
- 9x321 is inputted for Zip Code.
- 91 is inputted for Zip Code.
- 99999 is inputted for Zip Code.

Success Case:

- An error is displayed to the user stating the zip code is invalid.
- Search is not conducted and displayed to the user.

Failure Case:

- Search is conducted and displays results.

Execution Time: 1 minute.

4- Medication Lookup

Types of Testing:

- Unit, Negative, API, UI, Integration, Security, Stress, and End-to-End testing.

Assumptions:

- User is logged in
- User is at the Medication Lookup View

4.1 User searches by entering the Medication name

User enters a **valid** medication name using the search bar, zip code, and mile radius.

Successful Scenarios:

- Search request is successful
- API request is made and returns correct results to the system
- Medication Lookup view is updated to display the search results
- User is notified when no results are generated
- User is able to select the medication item from the search list to view its details

Failure Scenarios (at least one case must fail):

- Search request submission is unsuccessful
- Search is not executed and results are not generated
- No API request is made.
- API request is made and returns incorrect results to the system.
- Search displays wrong medication items that do not correspond to the user's input
- User is unable to selection a medication form the search list to view it

User enters a **invalid** medication name using the search bar, zip code, and mile radius

Success Scenarios:

- User is unable to submit the Medication Lookup Search request
- No API request is made
- An error message is displayed stating to follow the proper guidelines of the specific inclaid input
- User is given the option to re-input the incorrect fields.

Failure Scenarios:

- User is able to submit the Medication Lookup request
- API request is made.
- Search is executed and results are generated.
- No error message is displayed to the user.
- Wrong error message is displayed to the user. User is not given the option to re-input the incorrect fields.

Testing Plan:

- Execution Steps:
 - Go to the Medication Lookup View.
 - Click the search bar option
 - Input a valid Medication name
 - Click on the magnifying glass icon
- Test Cases
 - Test 1: Input valid medication name
 - Unit Testing:
 - Input:
 - User inputs on the search bar “Advil”
 - Success Case:
 - Search Request successful
 - User is notified if when no results are generated
 - User is able to select the Medication
 - Failure Case:
 - Search request unsuccessful
 - Execution Time: 1 minutes
 - Test 2: Testing invalid medication name
 - Negative Testing:
 - Input:
 - “12345abcd” or “ ” is imputed for Medication Name
 - Success Case:
 - An error is given to the user saying that the medication must be at least 1-100 characters long or the medication is spelled incorrectly.
 - Failure Case:
 - The search request is successful.
 - Execution Time: 1 minutes
 - Test 3: The results displays the Pharmacy providing the medication.
 - API Testing:
 - Input:
 - Advil is inputted for Medication name
 - Success Case:
 - A confirmation message notifies the user which displays the different vendors within the raidosu of the user that offer the medication.
 - If the user selects the medication, the API displays the list of prices associated with the medication
 - Note: Prices will be in US dollars
 - Search can return 0 results

- A maximum of 50 results will be displayed at a time.
- Failure Case:
 - The input displays a incorrect set of results
 - If the user selects the medication, the API displays the wrong prices associated with the medication.
- Execution Time: 2 minutes
- Test 4: Testing Database storage
- Unit Testing
 - Input:
 - “Favorite List” is inputted as name
 - Success Case:
 - The Favorite List is made and is stored successfully on the database include the date of the favorite list was made.
 - Failure Case:
 - The Favorite List is not stored in the database.
 - Execution Time: 2 minutes

4.2 User selects the View Medication on Medication Lookup.

Assumptions:

- User must be logged in
- User must be in the Medication Lookup view
- User successfully searched for a medication
- Search displays at least one medication

User selected the medication to be viewed

Success Scenarios:

- Medication view request is successful
- User is able to view the contents of Medication

Failure Scenarios:

- Medication view requisition is unsuccessful
- User is directed to the wrong Medication view
- User is unable to view the contents of Medication

Testing Plan:

- Execution Steps:
 - Go to the Medication Lookup View.
 - Click the search bar option
 - Input a valid Medication name

- Click on the magnifying glass icon
- User has already selected a valid medication.
 - For example, user types in Advil in the search bar.
- Test Cases
 - Test 1: Viewing a medication
 - Unit Testing:
 - Input:
 - User inputs on the search bar "Advil"
 - Success Case:
 - Search Request successful
 - User is notified if when no results are generated.
 - User is able to select the Medication
 - Failure Case:
 - Search request unsuccessful
 - Execution Time: 1 minute.
 - Test 2: The results displays the Pharmacy providing the medication.
 - API Testing:
 - Input:
 - User enters Advil on the search Bar
 - Success Case:
 - A confirmation message notifies the user which displays the different vendors within the radius of the user that offer the medication.
 - If the user selects the medication, the API displays the list of prices associated with the medication
 - Note: Prices will be in US dollars
 - Search can return 0 results
 - A maximum of 50 results will be displayed at a time.
 - Failure Case:
 - The input displays a incorrect set of results
 - If the user selects the medication, the API displays the wrong prices associated with the medication.
 - Execution Time: 1 minute.

4.3 User selects an Medication Item to put to Favorites List

Assumptions:

- User must be logged in
- User must be in the Medication View
- User successfully search for a medication
- User has less than 10 favorites in the favorite list

The user favorites the medication

Success Scenarios:

- The medication is added to the user's favorite List
- Medication vendor is favorited
- Favorite List incremented by 1.
- The action is updated on the database.
- Favorite must be added to the Favorite List in a maximum of 2 seconds.

Failure Scenarios:

- The medication is not added to the user's favorite list
- Medication vendor is not favorited
- Favorite List does not increment by 1.
- The action is not updated on the database.

The user unfavorites the medication

Success Scenarios:

- The medication is removed from the user's favorite list
- Medication vendor is unfavorited
- Favorite List decrements by 1.
- The action is updated on the database.
- Unfavoriting must be removed from the Favorite List in a maximum of 2 seconds.

Failure Scenarios:

- The medication is not removed from the user's favorite list
- Medication vendor is still favorited
- Favorite List does not decrease by 1.
- The action is not updated on database

Testing Plan:

- Execution Steps:
 - Go to the Medication Lookup View.
 - Click the search bar option
 - Input a valid Medication name
 - Click on the magnifying glass icon
 - User has already selected a valid medication.
 - For example, user types in Advil in the search bar.
- Test Cases
 - Test 1: Viewing a medication
 - Unit Testing:
 - Input:

- User inputs on the search bar "Advil"
- Success Case:
 - Search Request successful
 - User is notified if when no results are generated
 - User is able to select the Medication
- Failure Case:
 - Search request unsuccessful
- Execution Time: 1 minute.
- Test 2: The results displays the Pharmacy providing the medication.
- API Testing:
 - Input:
 - User enters Advil on the search Bar
 - Success Case:
 - A confirmation message notifies the user which displays the different vendors within the radius of the user that offer the medication.
 - If the user selects the medication, the API displays the list of prices associated with the medication
 - Note: Prices will be in US dollars
 - Search can return 0 results
 - A maximum of 50 results will be displayed at a time.
 - Failure Case:
 - The input displays a incorrect set of results
 - If the user selects the medication, the API displays the wrong prices associated with the medication.
 - Execution Time: 1 minutes
- Test 3: Testing the Database
- Integration Test
 - Input: User favorites Advil to their Favorite List
 - Success Case:
 - Medication is added to Favorite List and updated on the database including the date the favorite was made.
 - Failure Case:
 - Medication favorite is not stored in the database
 - Execution Time: 1 minute.
- Test 4: Favorite in Favorite List Limit
 - Unit Testing
 - Input:

- The user favorites another medication while at the favorite limit.
- Success Case:
 - The favorite item is not added and informed they have exceeded the limit of favorite per list.
- Failure Case:
 - User is able to favorite medication to their favorite list.
- Execution Time: 1 minute

4.4 User selects the View Favorite List Option:

Assumptions:

- User must be logged in
- User must be in the Medication Lookup View
- User successfully search for a medication
- User has less than 10 favorites in the favorite list

User selects the favorite list to be viewed

Success Scenarios:

- View Favorite List is successful
- User is redirected from the View Favorite List to the Favorite view
- User is able to view the contents of the Favorite List
- User is able to edit the contents of Favorite List

Failure Scenarios:

- View Favorite List is unsuccessful
- User is redirected to the wrong Favorite List View.
- User is unable to view the contents of the Favorite List.
- User is unable to edit the contents of Favorite List.

Testing Plan:

- Execution Steps:
 - Go to the Medication Lookup View.
 - User selects the Display Favorite List Option.
- Test Cases
 - Test 1: Viewing a Favorite Medication
 - Input Testing:
 - Input:
 - User selects an item off of favorite list
 - Success Case:

- User is taken to the Medication View which displays its different locations and information
- Failure Case:
 - Search request unsuccessful
- Execution Time: 1 minute.
- Test 2: Testing the Database
- Integration Test
 - Input: User favorites Advil to their Favorite List
 - Success Scenario:
 - Medication is added to Favorite List and updated on the database including the date the favorite was made.
 - Failure Scenario:
 - Medication favorite is not stored in the database
 - Execution Time: 1 minute.
- Test 3: Test Favorite in Favorite List Limit
 - Unit Testing
 - Input:
 - The user favorites another medication while at the favorite limit.
 - Success Case:
 - The favorite item is not favorited and use is informed they have exceeded the limit of favorite per list.
 - Failure Case:
 - User is able to favorite item to their favorite list.
 - Execution Time: 1 minute
- Test 4: Access Favorite in Favorite List Limit
 - Unit Test
 - Input:
 - User chooses "favorite medication in favorite list 1"
 - Success Case:
 - System retrieves Favorite List 1 and shows it to use with option to unfavorite/favorite
 - Fail Case:
 - System fails to retrieve favorite list 1
 - Execution Time: 1 minute

4.5 User selects the Refill Medication option.

Assumptions:

- User is logged in

- User must be in Medication View
- User must have at least one medication.

User inputs a **valid** refill date, time and recurring time.

Success Scenarios:

- Refill Medication submission request is successful
- Refill Medication is created and saved under Medication view
- Medication view is updated to display newly created Refill Medication
- User is redirected to Medication view
- Adding the refill reminder to the Reminder List takes less than 2 seconds.

Failure Scenarios:

- Refill Medication submission request is unsuccessful
- Refill Medication is not created and not saved under Medication view
- Medication view is not updated to display newly created Refill Medication
- User is not redirected to Medication view

User inputs a **invalid** refill date, time, and/or recurring time

Success Scenarios:

- Create Refill Medication submission is unsuccessful
- An error message is displayed stating to follow the proper guidelines of the specific invalid input
- Refill Medication is not created

Failure Scenarios:

- Create Refill Medication submission is successful
- An error message is not displayed
- Refill Medication is created
- Adding the refill reminder to the Reminder List takes less than 2 seconds.

Testing Plan:

- Execution Steps:
 - Go to the Medication View.
 - User has a valid Medication
 - User must select Refill Reminder Option
- Test Cases
 - Test 1: Input valid date and time
 - Unit Testing:
 - Input:
 - User inputs the date and time to be reminded of the refill.

- User inputs the recurring time. For example, every week, bi weekly, monthly, or yearly.
- Success Case:
 - A refill reminder is added to the Reminder List
 - User is notified that the reminder was created successfully via a confirmation message.
- Failure Case:
 - The refill reminder is not created
 - User receives an error message indicating that the date/time and/or recurring time was filled out incorrectly
- Execution Time: 2 minutes
- Test 2: Testing invalid date and time
- Negative Testing:
 - Input:
 - Date: ab/cd/ef32 or Date: ^45/12/3232 or Date" "/" "/" "
 - Success Case:
 - An error message is given to the user saying that the date must follow the correct format (MM/DD/YYYY) and use numerical values only.
 - The reminder is not created.
 - Failure Case:
 - The remainder is created.
 - Execution: 1 minute.
- Test 3: Testing
- Negative Testing:
 - Input:
 - User attempts to pick a past date and time to set the refill reminder.
For example, the System indicates the date and time to be 10/29/2021 6:18 pm
The user attempts to choose 10/15/2021 6:18 pm
 - Success Case:
 - An error message is given to the user saying that the user cannot select the past date from their current date.
 - Failure Case:
 - A confirmation message is given to the user that the refill reminder is created.
 - Execution Time: 1 minute.
- Test 4:

- Unit Testing
 - Input:
 - “Favorite List” is inputted as name
 - Success Case:
 - The Favorite List is made and is stored successfully on the database including the date of the favorite list was made.
 - Failure Case:
 - The Favorite List is not stored in the database.
 - Execution Time: 1 minute.
- Test 5: Access Refill Reminder in Reminder List
- Unit Test
 - Input:
 - User chooses “Reminder List”
 - Success Case:
 - System retrieves Reminder List and shows the user all of the user’s created refill reminders
 - Fail Case:
 - System fails to retrieve Reminder List
 - Execution Time: 1 minute

5- Weight Management/Calorie Counter:

Types of Testing:

- Unit, Negative, API, UI, Integration, Security, Stress, and End-to-End testing.

Assumptions:

- User is logged in.
- User is at the Weight Management View.
- User information is retrieved from Profile.

5.1 User selects the Create Goal Option:

User inputs **valid** Target Weight, and selects Exercise Schedule and Time Frame.

Success Scenarios (all cases must pass):

- Create Goal request submission is successful.
- User’s input is used to calculate the target weight by the given time frame.
- Weight Management View is updated to display the correct calculation results.

Failure Scenarios (at least one case must fail)::

- Create Goal request submission is unsuccessful.
- No input or wrong values are used to calculate the user's weight goal.
- No or wrong calculation results are displayed to the user in the Weight Management View.

User inputs **invalid** Target Weight, and selects Exercise Schedule and Time Frame:

Success Scenarios:

- Create Goal request submission is unsuccessful.
- An error message is displayed stating to follow the proper guidelines of the specific invalid input.
- Inputs are not stored in the system and no calculations are made.
- User is given the option to re-input the wrong fields.

Failure Scenarios:

- User is able to submit "Create Goal Option" request.
- No error message is displayed to the user.
- Incorrect error message is displayed to the user.
- User is not given the option to re-input the wrong fields.
- Invalid Inputs are stored and calculations are made.
- Weight Management View is updated to display results of invalid input calculations.

Test Plan:

- Execution:
 - User selects the Create Goal option.
 - Select Exercise Schedule.
 - Select Time Frame.
- Test Cases
 - Test 1: Input valid Target Weight.
 - Integration Test
 - Input:
 - 100 is inputted for target weight.
 - Success Cases:

- Correct calculation results are displayed to the user and stored in the system.
- Failure Cases:
 - No or wrong calculation results are displayed to the user in the Weight Management View.
- Execution Time: 1 minute.
- Test 2: Input invalid Target Weight.
- Negative Test
 - Input:
 - S100 is inputted for target weight.
 - Success Cases:
 - An error message is displayed to the user to only enter integers.
 - No calculations are displayed to the user.
 - Failure Cases:
 - No or wrong error message is displayed to the user.
 - Incorrect calculations are displayed to the user.
 - Execution Time: 1 minute.
- Test 3: Testing Database Storage
- Integration Testing:
 - Input:
 - 100 is inputted as Target Weight.
 - Success Case:
 - Input is stored successfully on the database.
 - Failure Case:
 - Input is not stored in the database.
 - Execution Time: 1 minute.

5.2 User selects the upload File Option:

Assumptions:

- User must be logged in.

- User must be at the Weight Management View.

User uploads **valid** File:

Success Scenarios:

- User's file is successfully uploaded.
- User's file is stored in the Weight Management View.
- User can select and view contents of the file.
- A confirmation message is displayed to the user.

Failure Scenarios:

- User's file is not uploaded.
- User's file is not stored in the Weight Management View.
- User can't select and display the file.
- No confirmation message is displayed to the user.

User uploads **invalid** File:

Success Scenarios:

- File upload request is unsuccessful.
- An error message is displayed stating to follow the proper guidelines.
- User is given the option to select a different file to upload.

Failure Scenarios:

- File upload request is successful.
- No error message is displayed to the user.
- Incorrect error message is displayed to the user.
- User is not given the option to select a different file to upload.

User reaches maximum file uploads for the day:

Success Scenarios:

- Once the last file is successfully uploaded and stored, a message is displayed to the user stating the limit has been reached.
- Upload file option will no longer be visible until the next day.

Failure Scenarios:

- No message is displayed to the user regarding reaching the maximum file upload limit.

- Upload file option is still visible to the User.
- User is able to successfully upload more files exceeding the file upload limit.
- File is stored and displayed in the Weight Management View.

Testing Plan:

- Execution:
 - User uploads a file.

Test Cases:

Execution: User presses input file

- Test 1: valid file
 - Unit Testing
 - Input:
 - Valid.jpg is uploaded
 - Success:
 - Valid.jpg is on the record after the database is checked to make sure the limit isn't breach. Valid.jpg is also visible to user
 - Failure:
 - Valid.jpg does not show up on the record, even without limit breach or valid.jpg is not visible to user.
 - Execution time: 1 minute
- Test 2: User inputs invalid file
 - Negative Testing
 - Input:
 - Invalid.jpg is uploaded
 - Success:
 - Invalid.jpg is not uploaded
 - Failure:
 - Invalid.jpg is accepted
 - Execution time: 1 minute
- Test 3: Test 2 per day limit

- Negative Testing
 - Input:
 - "inp1.jpg" , "inp2.jpg" , "inp3.jpg"
 - Success:
 - Inp1, inp2 are accepted, inp3 is rejected. Message is then shown informing user of limit
 - Failure
 - Inp3.jpg is accepted.
 - Execution time: 3 minutes.

5.3 User selects the Edit Weight Goal Option:

Assumptions:

- User is logged in.
- User is at the Weight Management View.

Type of Testing:

Success Scenarios:

- Fields are editable.
- User is able to refill the target weight, exercise schedule, and date.
- Recalculations are made with the new user input.
- Updated results are reflected in the Weight Management View.

Failure Scenarios:

- Fields are not editable.
- User is unable to refill the target weight, exercise schedule, and date.
- Recalculations are made with the wrong values (old input or garbage values)
- Previous calculation results are reflected in the Weight Management View.

Testing Plan:

- Execution:
 - User selects the Edit Weight Goal option.
 - User re-selects Target Weight, Exercise Schedule.
 - User re-inputs Time Frame.

Test Cases:

- Test 1: Save
 - Integration test
 - Input:
 - "120" and submit save.
 - Success Case:
 - User is informed that the input is saved.
 - Recalculations are made and correct results are displayed to the user.
 - Failure Case:
 - User is not informed that the input is saved.
 - No recalculations are made.
 - No correct results are displayed to the user.
 - Execution Time: 1 minute.
- Test 2: Database storage
 - Integration Testing
 - Input:
 - "120" and save
 - Success Case:
 - The input is saved to the database.
 - Fail Case:
 - The input is not saved to the database.
 - Execution Time: 1 minute.

5.4 User selects the Search Food Item Option:

Assumptions:

- User is logged in.
- User is at the Food Log View.

User selects the category to add food items under in their Food Log.

User inputs **valid** food item name in search bar

Successful Scenarios:

- Search request submission is successful.
- API request is made and returns correct results to the system.
- Food Log View is updated to display the search results.
- Search displays correct food items and calories that correspond to the user's input.
- User is notified when no results are generated.

Failure Scenarios:

- Search request submission is unsuccessful.
- Search is not executed and results are not generated.
- No API request is made.
- API request is made and returns incorrect results to the system.
- Search displays wrong food items and calories that do not correspond to the user's input.

User inputs **invalid** food item name in search bar

Successful Scenarios:

- Search request submission is unsuccessful.
- An error message is displayed to the user stating to follow the proper guidelines.
- User is given the option to re-input food item name in search bar.

Failure Scenarios:

- Search request submission is successful.
- Search is executed and wrong results are generated and displayed.

- No error message is displayed to the user.
- Wrong error message is displayed to the user.

Testing Plan:

- Execution:
 - User selects Search Food Item option.
 - Select a Food Category.
 - Click on search bar.
- Test Cases:
 - Test 1: Input valid Food Item Name.
 - Unit Testing
 - Input:
 - “Apples” is inputted for the food item name.
 - Success Case:
 - Search results that correspond to the user’s input are displayed.
 - User is notified if no results are generated.
 - Fail Cases:
 - Search results that do not correspond to the user’s input are displayed.
 - User is not notified if no results are generated.
 - Execution Time: 1 minute.

- Test 2: Input invalid Food Item Name.

Note: Invalid Food Item Name are item names that do not exist, contain special characters or numbers, or exceed more than 15 characters.

- Input:
 - “Enkmwd” is inputted for the food item name.
 - “2232” is inputted for the food item name.
 - “&apple” is inputted for the food item name.

- “strawberries and cream cake” is inputted for the food item name.
- Success Case:
 - Error message is displayed to the user.
 - Search is not conducted and displays results.
- Fail Case:
 - No error message is displayed to the user.
 - Search is conducted and displays results.
- Execution Time: 1 minute.
- Test 3: Validate API works properly
 - Input: “Apple”
 - Success Case:
 - System is returned information from the API that corresponds to the input. “Apple”, “Apple Pie”, etc.
 - System is returned 0 results from the API if no matches are found.
 - Fail Case:
 - System is returned incorrect information that does not correspond to the input. ‘Yogurt’ “Bananas” etc.
 - Execution Time: 1 minute.

5.5 User selects the Add Food Item Option

Assumptions:

- User is logged in.
- User is at the Food Log View.
- User must have successfully searched for an item.
- User has less than 20 food items added to the Food Log in the 24 hours.

User selects the Add option near the food item.

Successful Scenarios:

- Food Log is updated to reflect the newly added food item.
- Added Food item is the correct user selected item.
- Food item is added to the correct user selected category.
- A message is displayed to the user stating the food item has been successfully added to their Food Log.

Failure Scenarios:

- The user is unable to save a Food item to their Food Log.
- Wrong food item is added to the Food Log.
- Food item is added under the wrong user selected category.
- No message is displayed to the user.

Testing Plan:

- Execution:
 - User successfully searched for an item
 - User selected the Add Food Item Option.
- Test Cases:
 - Test 1:
 - Unit Testing
 - Input:
 - Add button near food item "Apple" is clicked.
 - Success Case:
 - Food item "Apple" is added to the Food Log.
 - Food item "Apple" is displayed to the user when viewing Food Log.
 - A confirmation message that the Item has been added to the Food Log is displayed.
 - Failure Case:
 - Food item "Apple" is not added to the Food Log.
 - A different food item is added to the Food Log.
 - Food item "Apple" is not displayed to the user when viewing Food Log.

- Execution Time: 1 minute.
- Test 2: Testing Database storage.
 - Integration testing.
 - Input:
 - Add button near food item “Apple” is clicked.
 - Success Case:
 - Food item “Apple” is successfully saved onto the Food Log.
 - Updated Food Log is successfully stored into the database.
 - Failure Case:
 - Food item “Apple” is not saved onto the Food Log.
 - Updated Food Log is not stored in the database. Test 3: Test Record Limit.
 - Execution Time: 1 minute.
 - Task 3: Test Food Log Max Limit
 - Negative Testing
 - Input:
 - “Apple” food item is added when Food is at its maximum limit.
 - Success Case:
 - “Apple” is not added to Food Log and user is informed they’re on limit
 - Failure Case:
 - “Apple” is added to Food Log and maximum limit is disregarded.
 - Execution Time: 1 minute

5.6 User selects the Add Custom Food Option:

User selects category to add food items under in their Food Log.

Assumptions:

- User must be logged in.

- User must be in the Food Log View.

User inputs **valid** custom food item Name, Description, and Calories:

Success Scenarios:

- Add Custom Food Item request is successful.
- Food Log is updated to reflect the newly added custom food item under the user selected category.
- A message is displayed to the user stating the food item has been successfully added to their Food Log.

Failure Scenarios:

- Add Custom Food Item request is unsuccessful.
- Custom food item is not added to the Food Log.
- Wrong food item is added to the Food Log.
- Custom food item is added under the wrong category.
- No message is displayed to the user stating the food item has been successfully added to their Food Log.

User inputs **invalid** custom food item Name, Description, and Calories:

Success Scenarios:

- Add Custom Food Item request is unsuccessful.
- An error message is displayed stating to follow the proper guidelines of the specific invalid input.
- User is given the option to re-input the wrong fields.

Failure Scenarios:

- "Add Custom Food Item" request is successful.
- Food Log is updated to reflect the newly added invalid custom food item.
- No error message is displayed to the user.
- Wrong error message is displayed to the user.
- User is not given the option to re-input the wrong fields.

Testing Plan:

- Execution:
 - User selects Add Custom Food
- Test Cases:
 - Test 1: Input valid Food Name, Description, and Calories.
 - Unit Testing:
 - Input:
 - Food Name: "LACNOR Banana Milk"
 - Food Description: "it is 180 ml, has 3 grams of fat, 0.1 grams of cholesterol, 177 mg of Calcium, etc."
 - Calories: 150 kcal.
 - Success Cases:
 - Food item is created and added to the Food Log.
 - Food item "Banana Milk" is displayed to the user when viewing Food Log.
 - A confirmation message that the Item has been added to the Food Log is displayed.
 - Able to select the food item and view its details.
 - Failure Cases:
 - Food item is not created and added to the Food Log.
 - Food item "Banana Milk" is not displayed to the user when viewing Food Log.
 - A different Food item is displayed to the user when viewing Food Log.
 - Unable to select the food item and view its details.
 - Execution Time: 1 minute.
 - Test 2: Input invalid Food Name, Description, and Calories.

Note: Invalid Food Item Name are item names that exceed 15 characters, contain special characters or numbers. Invalid Descriptions exceed the 1000 character limit. Invalid Calories include nonintegers.

- Unit Testing:
- Input:
 - Food Name: "LACNOR Banana Milk LACNOR Banana Milk LACNOR Banana Milk LACNOR Banana Milk LACNOR Banana Milk LACNOR Banana Milk LACNOR Banana Milk"
 - Food Description: exceeds 1000 characters.
 - Calories: "frfdsa" "%231"
- Success Cases:
 - Food item is not created and added to the Food Log.
- Failure Cases:
 - Food item is created and added to the Food Log.
 - Food item is displayed to the user when viewing Food Log.
 - A confirmation message that the Item has been added to the Food Log is displayed.
 - Able to select the food item and view its details.
- Execution Time: 1 minutes.
- Test 3: Test every other combination of invalid/valid cases for Test Name, Description, and calories.
 - Unit Testing.
 - Execution Time: 5 minutes.
- Test 4: Test Food Log Max Limit
 - Negative testing
 - Input:
 - "LACNOR Banana Milk" food item is added when Food is at its maximum limit.
 - Success Case:
 - "Apple" is not added to Food Log and user is informed they're on limit
 - Failure Case:

- “Apple” is added to Food Log and maximum limit is disregarded.
- Execution Time: 1 minute

5.7 User selects the Delete Food Item Option.

Assumptions:

- User must be logged in.
- User must be in the Food Log View.
- At least one food item must be in the Food Log.

User selects the Food Item to be deleted.

Success Scenarios:

- A confirmation message is displayed to the user verifying whether they want to delete the food item or not.
- The selected food item is deleted.
- Food Log is updated to reflect the changes.
- A message is displayed to the user once the food item is deleted.

Failure Scenarios:

- No confirmation message is displayed to the user.
- Wrong food item is deleted.
- No food item is deleted.
- Food Log view is not updated to reflect the deletion of the food item.
- No message is displayed to the user once the food item is deleted.

Testing Plan:

Execution:

- User selects the Food Item Delete option.
 - Tests Cases
 - Test 1- Delete a Food Item :
 - Unit Test
 - Input:
 - Delete Food Item “Apple” from Food Log.
 - Success Cases:

- "Apple" is deleted.
- A message is displayed to the user that "Apple" has been deleted.
- "Apple" is no longer visible on Food Log.
- Failure Cases:
 - "Apple" has not been deleted.
 - "LACNOR Banana Milk" has been deleted instead.
 - A message is not displayed to the user that "Apple has been deleted"
- Execution Time: 1 minute.
- Test 2: Testing Database storage.
 - Integration testing.
 - Input:
 - Delete Food Item "Apple" from Food Log.
 - Success Case:
 - Food Item "Apple" is removed from Food Log.
 - Updated Food Log is successfully stored into the database.
 - Failure Case:
 - Food item "Apple" is not removed from the Food Log.
 - Updated Food Log is not stored in the database.
 - Execution Time: 1 minute.

5.8 User selects the Counting Calories Option:

Assumptions:

- User is logged in.

- User is in the Calorie Counter View.
- System retrieves user inputted data from Weight Management and Food Log.
System retrieves the user's personal data from their profile.

Successful Scenarios:

- The following calculations are successfully made:
 - User's recommended daily calorie intake.
 - User's total calories eaten.
 - User's remaining available calories.
 - User's surplus or deficit in calories.
 - User's average amount of calories accross the week.
- Calculation results are correct.
- Calorie Counter View is updated to display these calculations.

Failure Scenarios:

- No calculations were made.
- System retrieves incorrect data to make the calculations.
- System retrieves no data to make the calculations.
- Incorrect results from calculations.
- Calorie Counter View does not display all calculation results to the user.

Testing Plan:

- Execution:
 - User selects count Calories Option.
- Test Cases:
 - Test 1:
 - Input:
 - System retrieves information from Weight Management and Food Log.

Success Cases:

- Correct information corresponding to the user is retrieved from the Weight Management and Food Log.
- Correct calculations are made.

- Correct results are displayed to the User.

Failure Cases:

- Incorrect information is retrieved from Weight Management and Food Log.
- Incorrect calculations are made.
- Incorrect results are displayed to the User.

Execution Time: 1 minute.

5.9 User selects the Export Food Log Option:

Assumptions:

- User is logged in.
- User is at the Food Log View.

User selects a destination to export Food Log to.

Successful Scenarios:

- Export food log request is successful.
- Correct Food Log is successfully exported to the user's selected destination.
- User can access the Food Log from their selected destination.
- Exporting the Food Log has no effect on the Food Log View in the application.
- A message is displayed to the user stating that the export was successful.

Failure Scenarios:

- Export food log request is successful.
- Correct Food Log is not exported.
- Correct Food Log is exported to the wrong destination.
- Wrong (old version or empty) Food Log is exported to the user's selected destination.
- User is unable to access the exported Food Log.
- Exporting the Food Log affects the Food Log View in the application.
- No message is displayed to the user.

Testing Plan:

- Execution:

- User selects the Export Food Log Option.

Test Cases

- Test 1: Testing Export Food Log option.
 - Unit testing
 - Input:
 - Export Reminder to desktop
 - Success case:
 - Food Log is successfully exported and found in desktop
 - A message is displayed to the user stating the export is successful
 - Failure Case
 - Food Log is exported to the wrong destination.
 - Incorrect food log is exported to the correct destination.
 - Incorrect food log is exported to the wrong destination.
 - Execution Time: 1 minute.

6. Track Records

Types of Testing:

- Unit, Negative, UI, Integration, Security, Stress, Performance, and End-to-End testing.

Assumptions:

- User is logged in.
- User is at the Track Records View.
- At least one record exists.

6.1 User selects the Search Records Option.

User inputs a **valid** (existing) name or date of a previously created record:

Successful Scenarios:

- Track Records request is successful.
- Track Records View is updated to display the results.
- Search displays correct record(s) that correspond to the user's input.
- User is notified when 0 results are generated.

- User is able to select a searched record and view its contents.

Failure Scenarios:

- Track Records request is unsuccessful.
- Search is not executed and results are not generated.
- Search is executed but Track Records View does not update to display the results.
- Search is executed and displays wrong records that do not correspond to the user's input.
- User is not notified when 0 results are generated.
- User is unable to select a record to view its contents.

User inputs an **invalid** name or date for a previously created record:

Successful Scenarios:

- Track Records request is unsuccessful.
- An error message is displayed to the user stating to follow the proper guidelines.
- User will be given the option to re-input name or date in the search bar.

Failure Scenarios:

- Track Records request is successful.
- No error message is displayed to the user.
- Wrong error message is displayed to the user.
- Search is executed and displays wrong records.

Testing plan:

- Execution: User presses search
- Test Cases
 - Test 1: Input valid name or date
 - Input Testing:
 - Input:
 - "Record 1" is inputted for name
 - "10/01/21" is inputted for date
 - Failure Cases:

- No record pops up that matches either name or date given
- Execution time: 1 minute
- Test 2: User chooses to view a specific record
- Unit Testing
 - Success Cases:
 - Record is shown
 - Failure Cases:
 - Incorrect record is shown
 - No record is shown
- Execution time: 1 minute

6.2 User selects the Categorize Records Option:

Assumptions:

- User is logged in.
- User is in the Track Records View.
- User successfully performed a record search.

Type of Testing: ———

User selects the Create Folder Option.

User inputs **valid** Folder Name:

Successful Scenarios:

- Create Folder request is successful.
- Folder is created with the correct user inputted name.
- Track Records View is updated to display the newly created folder.
- User is able to select and view the contents of the folder.

Failure Scenarios:

- User is unable to submit the Create Folder Request.
- Folder is not created.
- Track Records View is not updated to display the folder.
- User is unable to select and view the folder.

User inputs **invalid** Folder Name:

Successful Scenarios:

- Execution: User chooses to categorize a record
- Test Cases:
 - Test 1: User inputs valid folder name
 - Unit Testing:
 - Input:
 - “Folder a”
 - Success Cases:
 - Folder is created
 - Failure Cases:
 - Folder is not created
 - Execution time: 1 minute
 - Test 2: User inputs invalid folder name
 - Negative Testing:
 - Input:
 - “Aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.....”
 - Success Cases:
 - Folder is not created
 - Error given that folder name is too large
 - Failure Cases:

- Folder is created
- Execution time: 1 minute

6.3 View Folder

Assumptions:

- User is logged in.
- User is in the Track Records View.
- Folders must exist.

User selects the folder to be viewed.

Successful Scenarios:

- View Folder request is successful.
- User is able to view the contents of the folder.
- User is able to search for a record that was saved to the folder.
- User is able to select and view the records saved into the folder.

Failure Scenarios:

- View Folder request is unsuccessful.
- User is redirected to the wrong Folder View.
- User is unable to search for a record in the folder.
- User is unable to select and view the records in the folder.

Testing plan:

- Execution: User chooses to view a folder
- Test cases:
 - Test 1: User selects a folder
 - Unit Testing:
 - Success Case:
 - Folder contents are available to view
 - Failure Case:
 - User is directed to wrong folder
 - User is unable to view folder
 - User can not view specific contents of a folder

- Execution time: 1 minute.

7- Diet Recommendations

Types of Testing:

- Unit, Negative, API, UI, Integration, Security, and End-to-End Testing.

Assumptions:

- User is logged in.
- User information is retrieved from Profile.
- User is in the Diet Recommendations View.
- 10 dietary questions are displayed to the User.

7.1 User selects the Create Diet Recommendation option

User selects answers to all the questions. .

Successful Scenarios:

- Create Diet Recommendation request is successful.
- API request is made and returns correct, personalized results to the system.
- Diet Recommendations View is updated to display the results to the user.
- User is able to select meal items from the meal list.

Failure Scenarios:

- Create Diet Recommendation request is unsuccessful.
- Search is not executed and results are not generated.
- No API request is made.
- API request is made and returns incorrect results to the system.
- Diet Recommendations View is not updated to display results.

- User is unable to select items from the list.

Testing plan:

- Execution: User selects create diet recommendation
- Test Cases:
 - Test 1: User answers questionnaire
 - Unit Testing:
 - Input:
 - "5'8" inputted for height
 - "150" inputted for weight
 - "170" inputted for weight goals
 - Success Case:
 - Diet recommendation request is successful
 - API returns correct personalized data
 - User is able to view data and recommendations
 - Failure Case:
 - API makes no or wrong request
 - Diet recommendation is not displayed
 - Errors given
 - Time Execution: 2 minutes.

7.2 View Details of Recommended Meals.

User selects a meal from the generated list in the Diet Recommendations View.

Successful Scenarios:

- User is redirected to the Meal View.
- Correct details of the selected meal are displayed to the user.

Failure Scenarios:

- User is not redirected to the Meal View.
- User is redirected to the incorrect Meal View.
- Incorrect information is displayed to the user.
- No information is displayed to the user in the Meal View.

Testing Plan:

- Execution: User chooses to view specifics on recommended meals

- Test Cases:
 - Test 1: User chooses a specific meal from view
 - API Testing:
 - Input:
 - API filters through user questionnaire answers
 - Success Cases:
 - Accurate meal view is displayed
 - Failure Cases:
 - User is given wrong meal view
 - User is not shown anything
 - Execution Time: 1 minute.

7.3 Add Meal to Meal List

Assumptions:

- User must be logged in.
- User must be in the Diet Recommendations View.
- A list of meals has been successfully generated.
- Meal list contains less than 5 meals in 24 hours.

User selects the Add Meal Option

Successful Scenarios:

- Add Meal request is successful.
- A message is displayed to the user to confirm adding the meal item to the Meal List.
- Meal List View is updated to reflect the newly added meal item.
- Added meal item is the correct user selected item.

Failure Scenarios:

- Add Meal request is unsuccessful.
- No confirmation message is displayed to the user.
- Selected meal item is not added to the Meal List.
- Incorrect meal item is added to the Meal List.

Testing Plan:

- Execution: User adds a meal to meal list

- Test Cases:
 - Test 1: User chooses Add Meal
 - Unit Testing:
 - Success Cases:
 - Meal is added to meal list
 - Failure Cases:
 - Meal is not added to meal list
 - Incorrect meal is added to list
 - Time Execution: 1 minute.

7.4 View Meal Item from Meal List.

User selects a meal from the meal list.

Assumptions:

- User is logged in.
- User must be in the Meal List View.

Successful Scenarios:

- User is redirected to the Meal View.
- Meal View displays meal item's details.
- Correct meal item details are displayed to the user.

Failure Scenarios:

- User is not redirected to the Meal View.
- User is redirected to the wrong Meal View.
- Meal View does not display any information regarding the selected meal item.
- Meal View displays incorrect information regarding the selected meal item.

Testing Plan:

- Execution: User views a meal from meal list
- Test Cases:
 - Test 1: User chooses a specific meal from list
 - Unit Testing:
 - Success Cases:

- Meal details are displayed
- Failure Cases:
 - No meal details are displayed
 - Wrong meal details displayed
- Execution Time: 1 minute.

7.5 Delete Meal Item from Meal List.

User selects the delete option of a meal item from the Meal View.

Assumptions:

- User must be logged in.
- User must be in the Meal List View.
- Meal List has at least one meal added to it.

Successful Scenarios:

- Delete Meal request is successful.
- A confirmation message is displayed to the user verifying whether they want to delete the meal item or not.
- Meal is deleted from the Meal List.
- Meal List View is updated and reflects the removal of the meal item.

Failure Scenarios:

- Delete Meal request is unsuccessful.
- No confirmation message is displayed to the user.
- Meal item is not deleted from the Meal List.
- The wrong meal item is deleted from the Meal List.
- Meal List View does not reflect the removal of the meal item.

Testing Plan:

- Execution: User chooses to delete a meal from meal list
- Test Cases:

- Test 1: Uses chooses delete option when viewing a meal
- Unit Testing:
 - Success Cases:
 - Meal is no longer a part of meal list
 - Failure Cases:
 - Meal is still a part of meal list
 - Wrong meal is deleted
 - Execution Time: 1 minute.

8- Hot Topics

Types of Testing:

- Unit, Negative, UI, API, Integration, Security, and End-to-End testing.

Assumptions:

- User must be logged in.
- User must be in the Hot Topic View.

8.1 User selects the Hot Topic Search option.

User inputs **valid** Zip Code:

Successful Scenarios:

- Hot Topics Search request is successful.
- API request is made and returns correct results to the system.
- Hot Topics View is updated to display the results.
- User is notified when no results are generated.
- User is able to select an item from the search list to view its contents.

Unsuccessful Scenarios:

- Hot Topics Search request is unsuccessful.
- Search is not executed and results are not generated.
- No API request is made.
- API request is made and returns incorrect results to the system.
- Search displays incorrect information that does not correspond to the user's inputted zip code.

- User is user is unable to select an item and view its contents.

User inputs an **invalid** zip code input:

Successful Scenarios:

- Hot Topics Search request is unsuccessful.
- No API request is made.
- An error message is displayed to the user stating to follow the proper guidelines.
- User is not given the option to re-input the zip code in the search bar.

Failure Scenarios:

- Hot Topics Search request is successful.
- API request is made.
- Search is executed and results are generated.
- No error message is displayed to the user.
- Wrong error message is displayed to the user.
- User is not given the option to re-input the incorrect fields.

Testing Plan:

- Execution: User chooses to view hot topics
- Test Cases:
 - Test 1: Check user input with valid data.
 - API Testing:
 - Input:
 - "90840" for zipcode
 - Success Cases:
 - Hot topics from the area displayed
 - Failure Cases:
 - Hot topics from the wrong area displayed
 - Execution Time: 1 minute.
 - Test 2: Check user input with invalid data
 - Negative Testing:
 - Input:
 - "908a0" for zipcode
 - Success Cases:

- User is told to re-enter zipcode
- Failure Cases:
 - Search is made and hot topics displayed
- Execution Time: 1 minute.

8.2 Opt-In Notifications:

Assumptions:

- User is logged in.
- User is at the Hot Topics View.

User confirms zip code:

Success Scenarios:

- User receives emails on hot topics around them

Failure Scenarios:

- User does not receive any emails
- User receives emails on hot topics from a different area

Testing Plan:

- Execution: User verifies information
- Test Cases:
 - Test 1: Check user input with valid data.
 - Unit Testing:
 - Success Cases:
 - User starts receiving relevant hot topics through email
 - Failure Cases:
 - User receives no emails regarding hot topics
 - User receives the wrong hot topics
 - Execution Time: 1 minute.

9. Registration:

Assumptions:

- User is on registration page
- User does not have an existing account
- User is not logged in to existing account

9.1 User attempts to register and create an account.

User inputs valid email, username and password:

Success Scenarios:

- User's data is store
- User is taken back to homepage already logged in
- Welcome email is sent

Failure Scenarios:

- Data is not saved
- Data is seen as invalid and prompts user to enter valid data
- Incorrect data is saved
- User is not logged in once registering
- No email is sent

User inputs invalid email, username or password:

Success Scenarios:

- User is given one or more of the following errors:
 - Invalid username
 - Invalid email format
 - Invallid password
 - Missing special character
 - Too short
 - Repetitive
- Data is in a saved state so user does not have to re-enter valid data

Failure scenarios:

- No error is show
- Incorrect error shown
- Account created
- Valid data is not saved

User enters information already belonging to another account:

Success Scenarios:

- User is told an account with matching information already exists.
- User is give option to be taken to login page
- Data is in a saved state so user does not have to re-enter valid data

Failure Scenarios:

- Duplicate account is created
- No error is given
- Wrong error given
- Valid data is not saved
- Wrong error displayed

9.1 Test Plan:

- Example Input: "bobby@gmail.com:hks!ksmAA"

- Example Username: “DaBaby2”
- Execution steps:
 - Make sure you are logged out
 - Click Register
 - Enter valid email,username, and password in respective text boxes
- Unit Testing:
 - Success Case:
 - Display ‘Successfully Registered’
 - Redirected to homepage
 - Failure Case:
 - Display ‘Registration Failed’
 - Time Execution: 1 minute.
- Integration Testing:
 - Success Case:
 - Confirms that account is stored in database
 - Failure Case:
 - Account is not stored in database
 - Time Execution: 1 minute.
- Negative Testing:
 - Example Input: “[bobby!!gmail.com](#):hks!ksmAA”
 - Make sure you are logged out
 - Click Register
 - Enter invalid email,username, or password in respective text boxes
 - Success Case:
 - Displays ‘Registration Failed’
 - Failure Case
 - Displays ”‘Successfully Registered’
 - Time Execution: 1 minute.

10 Login:

Assumptions:

- User is on any view logged out
- User selects login button and
- User information is retrieved from the Profile.

10.1 User attempts to log into their account

User inputs valid login Email and Password:

Success Scenarios:

- User is logged in and taken to homepage

Failure Scenarios:

- User is told they entered wrong username/password
- User is not logged in but taken to homepage
- User remains in login page with no update

User inputs invalid login information:

Success Scenarios:

- User is told they entered wrong username/password
- User remains in login page
- User is told they have five opportunities before being locked out

Failure Scenarios:

- User is logged into an account
- User is sent back to homepage
- No error is displayed
- Wrong error displayed

10.1 Test Plan:

- Execution steps:
 - Create account if you do not already have one
 - User clicks Login .
 - Enters Email and Password in respective text fields
- Example Input: "bobby@gmail.com:hks!ksmAA"
- Test 1 Unit Testing:
 - Success Case:
 - Successfully Logged in
 - Failure Case:
 - Displayed 'Invalid Login'

- Time Execution: 1 minute.
- Test 2 Integration Testing:
 - Success Case:
 - Backend confirms that it is the correct information being input.
 - Failure Case:
 - 403 Error from backend
 - Time Execution: 1 minute.
- Test 3 Negative Testing:
 - Steps:
 - Create account if you do not already have one
 - User clicks Login
 - Enters incorrect Email and Password in respective text fields
 - Success Case:
 - Displays 'Invalid Login'
 - Failure Case
 - Successfully logs into account with invalid credentials
 - Time Execution: 1 minute.

10.2 User inputs five wrong username/password combinations:

Success Scenarios:

- User is temporarily locked out
- User is taken back to homepage

Failure Scenarios:

- User is logged in
- Incorrect attempt is not saved
- User is not shown error
- Wrong error shown

10.2 Test Plan:

- Example Input: "bobby@gmail.com:hks!ksmAA"
- Execution steps:
 - Create account if you do not already have one
 - User clicks Login .

- Enters an incorrect Email and Password in respective text fields and click 'Login'
- Repeat previous step 5 times
- Test 1: Unit Testing:
 - Success Case:
 - Displays 'Your account is locked! Try again later or reset password.'
 - Failure Case:
 - Does not display text saying account is locked
 - Time Execution: 2 minute.
- Test 2: Integration Testing:
 - Success Case:
 - Account is flagged as locked in the database.
 - 403 Error
 - Failure Case:
 - Database does not have the account flagged as locked.
 - Time Execution: 2 minute.

10.3 User has two step authentication

Assumptions:

- User has an account
- 2FA is enabled to you
- User has correct username and password
- User is in login page

User enters username/password information

Success Scenarios:

- Email verification is sent in a timely manner
- User is taken to passcode page

Failure Scenarios:

- Email is never sent
- Email arrives late
- User is logged in without needing to verify
- User is not taken to passcode page
- User is taken to homepage
- User is taken back to login page

User enters correct passcode:

Success Scenarios:

- User is verified and logged in

- User is taken back to homepage

Failure Scenarios:

- User is not logged in
- User is taken to homepage
- User is taken back to login page
- No error displayed
- Wrong error displayed

User enters invalid passcode:

Success Scenarios:

- User is given error
- User remains in passcode page
- User is told they have 5 tries before being locked out

Failure Scenarios:

- User is logged in
- User is sent back to homepage
- No error is displayed
- Wrong error displayed

10.3 Test Plan:

- Example Login: bobby@gmail.com:hks!ksmAA
- Auth code must be retrieved from authenticator
- Execution steps:
 - Create account if you do not already have one
 - Ensure 2FA is enabled and sign out.
 - User clicks Login .
 - Enters Email and Password in respective text fields and click 'Login'
- Test 1: Unit Testing:
 - Success Case:
 - Account is logged in, and redirected to home page.
 - Failure Case:
 - Account is not logged in. Displays Internal Error.
 - Time Execution: 2 minute.
- Test 2: Integration Testing:
 - Success Case:

- Logs in user from back end comparing generated auth code successfully.
- Failure Case:
 - Auth code is successful but Log In fails. Internal Error
- Time Execution: 2 minute.
- Test 3: Negative Test:
 - Example Login: bobby@gmail.com:hks!ksmAA
 - Auth code: "abcd"
 - Success Case:
 - Account is not logged in.
 - Failure Case:
 - Account is logged in. Authentication Error.
 - Time Execution: 2 minute.

11. Authorization/Security

This section will consist of mainly automated functions which will run when running our test cases. Authentication and security is hard to test through only navigating the website.

Requirements: Our internal testing files, Python 3.8 and above , Selenium

Execution Steps: Download testing folder. Execute main file.

Test Case	Steps to create?	Type?	Execution Time	Success	Fail
Authorized User Check	Log into a user account, and get the path/link for a Record. Sign out, and sign into another account. In address bar of browser enter the Record path.	Security Test	1 Minute	403 Error: User does not have access.	User is allowed to view a Record
XSS Testing	In user input locations such as where you enter your Email to log in, you enter a string such as <code><script>alert('XSS')</script></code> . You then do this multiple times with different common xss attacks.	Security Test	5 minutes	No XSS Vulnerabilities found.	XSS Vulnerability found.
Brute Force Testing	Get proxies from team members. Use python to create a multithreaded requests application that tries logging into one account from many proxies.	Stress Test	5 minutes	Brute force attack occurring.	Brute force attack not detected.

Cost, Effort, and Resources

Summary of Test Features and Resources

No.	Features	# of Test Cases	Time (min)	Resource
1	Medication/Health Recorder	14	20	Dante
2	Reminders	8	10	Angel
3	Health Locator	8	10	Rami
4	Medication Lookup	13	14	Emily
5	Weight Management/Calorie Counter	22	30	Sebastian
6	Track Records	5	5	Jacob
7	Diet Recommendations	5	6	Dante
8	Hot Topics	3	3	Angel
9	Registration	3	3	Rami
10	Log in	8	13	Emily
11	Registration	3	11	Jacob

Summary of Efforts and Cost

No.	Recourse Name	Effort (min)	Cost (\$30/hr) (0.5/1min)
1	Dante	26	13.5
2	Angel	13	6.5
3	Jacob	16	8
4	Sebastian	30	15
5	Emily	27	13.5
6	Rami	13	6.5
Totals		109 (1.82 hrs)	\$63

Total Estimated Time to complete the testing is approximately 2 hrs (109 minutes) with a total cost of \$63 with the assumption of 30\$/hr per resource.