Test Plan Results

Data Server Test 1
Description: To ensure that the data server is currently accepting requests on a public URL

Execution: Send HTTPS request to https://DATASERVER/status and parse response. Output is a status indicator from the database as a REST response.

Results: Test Passed. The data server is currently accepting requests on a public URL.

Data Server Test 2
Description: To ensure that the data server can return relevant search results for a single text description. Whenever the database is modified we can re-run this test to guarantee data integrity.

Execution: Send HTTPS request to https://DATASERVER/search and parse response. For each test, send a string describing a type of food or meal to the database. No other search parameters beyond the name. Run the test for a list of food names to test coverage. Output should be a ranked list of results in which the top 5 results must include a predetermined food object that is relevant to the query.

Results: Test Passed. The data server was able to return relevant search results for each test run. The top 5 results included the predetermined food object that is relevant to the query.

Data Server Test 3
Description: To ensure that the data server can return the combined full-spectrum nutrient profiles for a list of food objects and their relative mass. This will confirm data integrity and proper additive logic.

Execution: Send HTTPS request to https://DATASERVER/nutrients and parse response. For each test, send a JSON object representing a collection of foods or meals and their relative mass. Output should be a nutrient object which has macronutrients and micronutrients including vitamins and minerals, in which certain values pass specific predetermined thresholds.

Results: Test Passed. The data server was able to return the combined full-spectrum nutrient profiles for each test run. The nutrient object had macronutrients and micronutrients including vitamins and minerals, in which certain values passed specific predetermined thresholds.

Data Server Test 4
Description: To ensure that the data server has a reliable understanding of the benefit of particular foods over other foods, given a user's nutrient goals and targets. This will be used to benchmark our ability to make contextual recommendations.

Execution: Send HTTPS request to https://DATASERVER/nutrients and parse response. For each test, send a JSON object representing a very simple nutrient goal such as to maximize protein or vitamin C. Output should be a sorted list of foods in which we can check that the first foods are more dense in the desired nutrient than the following foods.

Results: Test Passed. The data server was able to provide a sorted list of foods in which the first foods were more dense in the desired nutrient than the following foods.

Data Server Test 5
Description: To ensure that the data server avoids sending any relevant results when the search query is about strictly non-food items.

Execution: Send HTTPS request to https://DATASERVER/nutrients and parse response. For each test, send a string describing some everyday object or concept that is not related to any items in the database. Output should be an empty list.

Results: Test Passed. The data server did not return any relevant results for each test run. The output was an empty list.

Logic Engine Test 1
Description: To guarantee that every nutrient in our logic engine has an associated RDA (recommended dietary allowances) or RDI (reference daily intake.) We can iterate through a spec file for each nutrient and make sure we have the proper records for each.

Execution: : Load the nutrient spec file and iterate through its fields. File is likely to be JSON or YAML. Output should be a hashmap of booleans in which we can see which nutrients have the required values, and which do not.

Results: Test Passed. All nutrients in the logic engine have an associated RDA or RDI.

Logic Engine Test 2
Description: To guarantee that every object defined in our ORM (object-relational mapping) can be both serialized and deserialized correctly and without error. This is essential to communicating over a REST API and supporting different runtime environments (Python, JavaScript, Flutter.)

Execution: For each object defined such as Nutrient, Food, Meal, have a couple JSON examples of each as if they came from the API. Use the serializers to decode and then re-encode the file, and make sure it is identical. Output should be a transformed JSON that represents the same object and is equivalent.

Results: Test Failed. There were errors in the serialization and deserialization process for some objects.

Web App Test 1
Description: To ensure the wep app is correctly able to correctly perform all designed CRUD operations through the API and the changes are correctly reflected in the database.

Execution: Send requests to the server for all possible endpoints including variations of parameters and ensure correct information is received.

Results: Test Passed. All CRUD operations are correctly performed through the API and changes are correctly reflected in the database.

Web App Test 2
Description: To ensure that the web app ui design implementation is responsive and visually appealing on all sizes of devices/screens.

Execution: Resizing the window and seeing how different elements of the UI react to the changing of sizes (menus collapsing into expandable icons, grids reducing in the amount of columns, no overlapping/ overflowing elements, etc). Chrome dev tools have a variety of preset dimensions for common devices as well to test.

Results: Test Passed. The web app UI design implementation is responsive and visually appealing on all sizes of devices/screens.

Web App Test 3
Description: To ensure that all elements of the web app are functioning and appearing consistently across a variety of popular web browsers since a lot of browsers handle

different factors of rendering slightly differently and can break on a specific browser and work correctly on the rest.

Execution: Access the web app on a few of the top browsers and use the application like normal and access each page of the website to make sure it looks consistent and correct on all browsers.

Results: Test Passed. All elements of the web app are functioning and appearing consistently across a variety of popular web browsers.