

**Florida International University
School of Computing and Information Sciences**

Final Deliverable - Senior Project

Food Frequency Questionnaire 2.0

Team Members

Dariana Gonzalez

Daykel Muro

Product Owner(s): Dr. Cristina Palacios

Instructor: Masoud Sadjadi

The MIT License (MIT)

Copyright (c) 2016 Florida International University

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON INFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Abstract

This document describes the second iteration of the Food Frequency Questionnaire (FFQR), an application intended to help clinicians and pediatricians to assess nutritional requirements in infants between 0 and 24 months. This iteration of the project mainly focus on the implementation of food and nutrients recommendations based on questionnaire responses and nutrient breakdown done in previous version. This document is intended to provide a clear understanding of the planning, design, development, implementation and validation conducted during version 2.0.

The document's first section focuses on the introduction of version 2.0 and explains how it adds business value to FFQR version 1.0. The second section comprises the description of the system design, project plan and validation process. Third section shows diagrams, graphs and tables to bring a high level representation of the system's functionalities, capabilities and design.

Table of Contents

INTRODUCTION	5
Current System Overview	5
Purpose of version 2.0	5
Implemented User Stories	6
Pending User Stories	19
PROJECT PLAN	19
Hardware and Software Resources	19
SPRINTS PLAN	20
Sprint 1	20
Sprint 2	21
Sprint 3	21
Sprint 4	22
Sprint 5	22
Sprint 6	23
SYSTEM DESIGN	23
Architectural Patterns	23
Design Patterns	24
System and Subsystem Decomposition	24
SYSTEM VALIDATION	24
Regression Testing	24
Testing of Current Version	26
GLOSSARY	30
APPENDIX	31
Appendix A UML Diagrams	31
Appendix B User Interface Design	34
Appendix C Sprint Meeting Reports	41
Appendix D Documentation	46
REFERENCES	50

INTRODUCTION

Infants require a special nutritional care to guarantee their progress toward wellness and healthy growth. Several diseases can be avoided by implementing an appropriate diet that follow the proper recommendations. The first two years of life are critical in the infant life to develop healthy dietary patterns[1]. A healthy diet in the early time of an infant's life can help to reduce the risk of getting later in adulthood chances of different chronic diseases, including obesity.[2]

Nowadays, there are some tools available for the calculation of infant nutrition but most of them are outdated and not validated. The few tools that are validated are very limited and they do not include the many new infant foods introduced in the market in recent years. The assessment of infant diet by clinicians and pediatricians is still cumbersome.

At FIU Department of Dietetics and Nutrition, Dr. Cristina Palacios and her team created and validated an Infant Food Frequency Questionnaire in 2015 that gives the necessary tools to guarantee an appropriate assessment of infants' nutrition. This tool has called the attention of many colleagues, researchers and parents.

Current System Overview

FFQR version 1.0 was implemented to provide a software solution to gather answers from parents through an online version of the questionnaire that was previously being filled out as a Word Document. To achieve this purpose the online questionnaire is accessible to a *“Participant”* through a URL (<http://localhost:4200>). Once in the landing page, *Participants* must enter a *“valid”* questionnaire ID that must be provided by the *“Administrator”* in order to access the questionnaire page. The *“Participant”* fills the questionnaire and submits it. Once the questionnaire is submitted the system will calculate the nutrients breakdown for each food item completed by the *“Participant”*. Current system displays the nutrients breakdown in a modal window and once that window is closed all the calculations are lost. Questionnaire results are not persistent in v1.0. They are not being saved on the database and also the calculations are not supposed to be viewed by the *“Participant”*. This was a palliative solution implemented by the previous team to show the nutrient breakdown calculations to the product owner during version 1.0 demo.

The current system does not provide any administrative functionality to modify, create and delete the food items and nutrients that are included in the questionnaires. Also, it does not provide access to responses of questionnaires previously submitted.

Purpose of version 2.0

The purpose of version 2.0 is to implement the Admin Portal of the Food Frequency Questionnaire web application. The Admin Portal comprises three main features (Food Items, Questionnaire Results and Recommendations) all of them intended to be used only by the *“Administrator”*. Through the Food Items menu in the Admin Portal, the *“Administrator”* will be able to add, modify and delete the food items that are included in the questionnaires and also the nutrient values for each one of them. The Admin Portal will allow the *“Administrator”* to view the responses from submitted questionnaires by accessing the Questionnaire Results menu. By navigating to the *“Recommendations”* menu, the *“Administrator”* will see the recommendations for food and nutrients generated by the systems based on the predetermined recommended values provided by Dr. Palacios and the responses the *“Participant”* submitted.

Version 2.0 will also address some functionalities that were left out of version 1.0 scope and are critical for the development of the current version such as saving the questionnaires results in the database, adding the infant age in the questionnaires and grouping food items by food category.

USER STORIES

The following section describes the breakdown of the work done during the second iteration of the Food Frequency Questionnaire web application. We divided the work for these iteration into three epics: (1) Research and Setup, (2) Nutrient Breakdown Results and (3) FFQR Admin Page. The first two epics were very brief and comprised only one spring. They were mainly focused on getting familiar with the implementation done on the previous iteration, evaluating the actual scope of version 1.0 and identify blockers that were fixed as part of the second epic. Epic 1 consisted of tasks such as “Setup and run v1.0 in local environments” and “Explore actual scope of v1.0”. For example, as a result of the exploration conducted during Epic 1, it was found that the questionnaire results were not being saved into the database. Instead, they were just being displayed in a modal window on the user side. This and other issues were solved during Epic 2 in order to get everything set up for the main epic: (3) FFQR Admin Page.

Epic 3 has the user stories associated with the implementation of the Admin Portal, a component of the Food Frequency Questionnaire web application intended for the Administrator to manage the food and nutrients as well as viewing the questionnaires' results and food/nutrients recommendations. All stories are described in this section.

Implemented User Stories

Epic 2 - Nutrient Breakdown Results

- **FFQR-2-01 Result Class Diagram**

As a developer, I want to represent in a class diagram every aspect of a questionnaire result so every team member can have a better understanding of its structure.

- *Acceptance Criteria:*

1. Every aspect of the questionnaire result is represented in the class diagram.

- **FFQR-2-02 Save questionnaire results**

As a developer, I want the results of the questionnaires to be stored in the database after the user fill and submit them through the Questionnaire page. The results are required to be shown later in the Admin Page for the Administrator to view them and also needed to calculate food and nutrients recommendations.

- *Acceptance Criteria:*

1. After the user submits a questionnaire, the results corresponding to that questionnaire are saved in the Results collection in the MongoDB.
2. The results document corresponding to the questionnaire submitted contains all fields described in the Result class diagram.

- **FFQR-2-03 Retrieve all questionnaire results**

As a developer, I want an endpoint in the ffq-food-item-service that serve requests to retrieve all results in the database. This endpoint will serve the front end application.

- *Acceptance Criteria:*

1. After at least one questionnaire has been submitted and the fff-food-item-service is running on localhost:9090, when I do a GET request to endpoint <http://localhost:9090/ffq/results/all> using Postman, I get a Json payload with a list of results.

- **FFQR-2-04 View "Success" message on questionnaire submission**

As a user, I want to be able to view a "Success" message on the Questionnaire Page when I submit the responses and they are successfully saved.

- *Acceptance Criteria:*

1. After the responses to the questionnaire are submitted the user does not see the results on the page.
2. After the responses to the questionnaire are submitted the user sees a Success message indicating the responses were successfully submitted and saved.

- **FFQR-2-05 Include questionnaire ID on answers submission payload**

As a developer, I want to include the questionnaire ID in the JSON payload that is sent to the backend when users submit questionnaires, in order for the API to save it into the database as part of the result object.

- *Acceptance Criteria:*

1. When submit questionnaire responses the questionnaire ID is sent through the POST request to the back-end.

Epic 3 - FFQR Admin Page

- **FFQR-2-06 Admin Page UI Mockup**

As a developer, I want to have a visual representation of the Admin Page UI flow should look like.

- *Acceptance Criteria:*

1. Admin UI mockup is created and approved by PO in order to start its implementation.

- **FFQR-2-07 Create Admin Page / Admin Page Routing**

As a user, I want to be able to access the Admin Portal by entering in my browser the same URL used for the questionnaire page followed by the word "admin".

- *Acceptance Criteria:*

1. When I enter <http://localhost:4200/admin> in the browser I can see a temporary blank landing page.
2. When I enter <http://localhost:4200> in the browser I can see the Questionnaire page.

- **FFQR-2-08 View list of food items (Back-end)**

As a developer, I want an endpoint in the ffq-food-item-service that serve requests to retrieve all the food items currently existing in the database. This endpoint is to be consumed by the front end application to get the list of food items and show it in the "Food Items" page.

- *Acceptance Criteria:*

1. When I do a GET request to endpoint <http://localhost:9090/ffq/allfoodsnutrients> through Postman, the response obtained contains a JSON payload with a list of the 54 existing food items.

- **FFQR-2-09 View list of all foods items (Front-end)**

As a user, I want to be able to view the list of all foods/nutrients in the food item page of the Admin portal.

- *Acceptance Criteria:*

2. When I click to the Food Item page inside the Admin Portal, I can see the list of all 54 existing food items.
3. All food items should have include the hyperlink to move into the Update Modal component.

- **FFQR-2-10 Add food item/nutrients (Front-end)**

As an administrator, I want to be able to add new food items and nutrients to the database through the Admin Page.

- *Acceptance Criteria:*

1. When I navigate to the Admin Page (<http://localhost:4200/admin>) and click on the “New Food Item” button, the application redirects me to the New Food Item view (<http://localhost:4200/admin/fooditem/new>).
2. Once in the New Food Item view, the application allows me to fill out all fields corresponding to the food item and corresponding nutrients.
3. When I click the Add Food Item button, I see a pop up message notifying “*Food item added successfully!*”.
4. When I click on the “*Ok*” button in the pop up message, the application redirects me to the Food Items view where I can see the new food item I just added listed at the end of the list.

- **FFQR-2-11 Update food item/nutrients (Front-end)**

As an administrator, I want to be able to update food items and nutrients to the database through the Admin Page.

- *Acceptance Criteria:*

1. When I navigate to the Food Items tab in the Admin Page (<http://localhost:4200/admin>) and click on food item I want to update , the application redirects me to the Update Food Item view (<http://localhost:4200/admin/fooditem/{fooditemid}>).
2. Once in the Update Food Item view, the application allows me to modify all fields corresponding to the food item and corresponding nutrients.
3. When I click the Update Food Item button, I see a pop up message notifying “*Food item updated successfully!*”.
4. When I click on the “*Ok*” button in the pop up message, the application redirects me to the Food Items view where I can see the food item I just updated listed.
5. When I click on the same food item, the application redirects me to the Update Food Item view where the updates I just did should show up.

- **FFQR-2-12 Delete food items/nutrients (Front-end)**

As a user, I want to be able to delete food items/nutrients inside the Food Item page of the Admin portal.

- *Acceptance Criteria:*

1. When select desired food item and click on the trash can icon, a popup windows should be displayed asking the user for confirmation. That confirmation will have a Cancel option just if the user decide not to delete the item.
2. After User confirmation of deletion, the food item and all its nutrients related should be deleted from the database.
3. Refresh automatically the admin page after deletion to show the food item deletion on the UI.

- **FFQR-2-13 Add food items/nutrients (Back-end)**

As a developer, I want to have an endpoint in the food item microservice that allows me to add new food items and nutrients through a POST http request.

- *Acceptance Criteria:*

1. When I do a POST request to the endpoint <http://localhost:4200/createfoodnutrients> passing the proper JSON payload in the request body with the data corresponding to a new food

item/nutrients, the back-end microservice should save the new food item/nutrients in the database and return a 200 message.

- **FFQR-2-14 Update food items/nutrients (Back-end)**

As a developer, I want to have an endpoint in the food item microservice that allows me to update a food item and its nutrients through a PUT http request.

- *Acceptance Criteria:*

1. When I do a PUT request to the endpoint `http://localhost:4200/updatefoodnutrients` passing the proper JSON payload in the request body with the data corresponding to the food item and corresponding nutrients to be updated, the back-end microservice should update the existing food item and its nutrients in the database and return a 200 message.

- **FFQR-2-15 Delete food item/nutrients (Back-end)**

As a developer, I want to have an endpoint in the food item microservice that allows me to delete food items and its nutrients through a DELETE http request.

- *Acceptance Criteria:*

1. When I do a DELETE request to the endpoint `http://localhost:4200/delete?id={objectId}` passing as query parameter the object id of the food item to be deleted, the back-end microservice should delete the corresponding food item and its nutrients from the database and return a 200 response.

- **FFQR-2-16 View all questionnaire results (Front-end)**

As a user, I want to see on the Admin portal's results page the list of questionnaire calculations stored in the database.

- *Acceptance Criteria:*

1. In the admin page a list of all questionnaire results stored into the database should be displayed. Results must include:
 - *Questionnaire ID*
 - *Patient name*
 - *Infant age*
 - *User choices that includes: name of the food item, frequency, frequency type, servings and sugar added.*
 - *Breakdown calculations for nutrients .*
2. If no questionnaire results are available , a message of "No questionnaire have been submitted yet! " should be displayed.

- **FFQR-2-17 Implement in the add/update views the handling of "servings" (Front-end)**

As a developer, I want the application to be able to properly handle the entering of multiple servings in Add Food Item or Update Food Items views.

- *Acceptance Criteria:*

1. When the user enters multiple servings for a food item in the Add Food Item or Update Food Items views, separated by commas in the "Servings" field, the application should parse the servings string and create a Serving object for each serving entered that is then added to a 'servings' array that is included in the JSON payloads for the POST and UPDATE requests to add and update food items.

- **FFQR-2-18 Implement in add / update view the handling of "multiples food types" (Front-end)**

As an administrator, I want to be able to add multiple food types for food items.

- *Acceptance Criteria:*

1. When I have a food item that has multiple food types (i.e Rice and Pasta has two food types: Whole Grain and Regular) where each food type has an individual nutrients list, I want to be able to manage the multiple food types through the Add Food Item or Update Food Items views.
- **FFQR-2-19 REFACTOR: Add "infant age" field in questionnaire form (Front-end)**

As a user, I want to be able to enter the infant age in the questionnaire page and I want that age to be saved in the database along with my questionnaire results.

 - *Acceptance Criteria:*
 1. The user is able to enter the infant age in months in the questionnaire form.
 2. The angular service retrieves the age entered from the input field and send it in the JSON payload to the corresponding endpoint.
 - **FFQR-2-20 Create results model and service in web app (Front-end)**

As a developer, I want to have, in the food item API, a model that maps the attributes of a Results as well as a service that handle the database operations through the MongoRepository class.

 - *Acceptance Criteria:*
 1. A Result model is added to the food item API that maps the following fields for Results objects:
 - *ObjectId id*
 - *patientName*
 - *ageInMonths*
 - *questionnaireId*
 - *userChoices*
 - *weeklyTotals*
 - *dailyAverages*
 2. A Results service is added to the food item API, that serve the following database operations:
 - *findAll()*
 - *save(results)*
 - *getResultByQuestionnaireID(questionnaireID)*
 - **FFQR-2-21 View all questionnaire results (Back-end)**

As a developer, I want to have an endpoint in the food item microservice that allows me to obtain the list of all questionnaire results through a GET HTTP request.

 - *Acceptance Criteria:*
 1. When I do a GET request to the endpoint `http://localhost:4200/results/all`, the back-end microservice should return a JSON payload with the list of all questionnaire results retrieved from the database in a 200 response.
 - **FFQR-2-22 REFACTOR: API to handle new field added to DB "PortionSize"**

As an administrator, I want to include the portion size for food items which is the size of food that contains the nutrients amounts to be entered.

 - *Acceptance Criteria:*
 1. When I navigate to the Add Food Item or Update Food Items views, I'm able to add/update the portion size for food items.
 2. The portion size for food items is saved as a field for food items documents in MongoDB.
 - **FFQR-2-23 REFACTOR: Add "infant age" field in questionnaire and food item services (Back-end)**

As a developer, I want the existing questionnaire controller and questionnaire model to handle the handling of the infant age in the ffq-questionnaire microservice.

- **Acceptance Criteria:**
 1. The infant age is included as an attribute in the FFQuestionnaire model and constructors are refactored to properly handle it:
@JsonProperty("patientAge")
private String patientAge;
 2. The QuestionnaireController is refactored to properly handle the new questionnaire field.

- **FFQR-2-24 Validate result view when there are no questionnaire submissions to display**

As a user, I want to see a validation on admin page when no questionnaires have yet been submitted.

- **Acceptance Criteria:**
 1. If no questionnaires are submitted at the time of admin being logged . A “*No questionnaires have been submitted yet!* ” validation message should be displayed.

- **FFQR-2-25 Display success message upon successful submission/update of food item**

As a user, I want to validate that a food item was added/ updated successfully on submit.

- **Acceptance Criteria:**
 1. A message “*Food item added/updated successfully!*” upon submission of food item added/updated action.

- **FFQR-2-26 Display total calories in questionnaire results page**

As a user, I want to be able to see the calculation for Total Calories after questionnaire submission.

- **Acceptance Criteria:**
 1. On questionnaire page must appear the total calories calculations next to the total % calories for fat, proteins and carbs.
 2. This calculations should be included into the table with all nutrients breakdown.

- **FFQR-2-27 Redesign database schema to include nutrients and food recommendations tables**

As a developer, I want to update the database schema in order to include the new following tables:

sys_nutrients_recommendations and *sys_food_recommendations*

- **Acceptance Criteria:**
 1. An updated Database Schema diagram is obtained that represent the design of the database including the new tables added.

- **FFQR-2-28 Implement Java API models for nutrients recommendations (Back-End)**

As a developer, I want to add the proper models in the food item microservice (Java API), to handle and map nutrient recommendation objects.

- **Acceptance Criteria:**
 1. A *Recommendation* model is added to the food item microservice and includes the following fields:
 - *nutrientName*
 - *calculatedAmount*
 - *recommendedAmount*
 - *status*
 2. A *NutrientRecommendation* model is added to the food item microservice and includes the following fields:
 - *questionnaireId*
 - *patientName*

- *patientAgeInMonths*
- *recommendationsList* (a list of *Recommendation* objects)

- **FFQR-2-29 Implement Java API controller to handle nutrients recommendations (Back-End)**

As a developer, I want to implement a *NutrientRecommendation* controller (endpoint) in the food item API, to handle HTTP requests related to the nutrients recommendations.

- *Acceptance Criteria:*

1. A *NutrientRecommendationController* is implemented to serve HTTP requests on *http://localhost:9090/nutrientsrecommendations* endpoint.
2. When I do a GET request to endpoint *http://localhost:9090/nutrientsrecommendations/all*, a JSON payload with the list of all system nutrient recommendations is returned in the body of the response.
3. When I do a GET request to endpoint *http://localhost:9090/nutrientsrecommendations/calculate/{questionnaireID}*, the controller calculates nutrient recommendations based on the questionnaire responses of the questionnaire passed and the system nutrient recommendations baseline.
4. When I do a POST request to endpoint *http://localhost:9090/nutrientsrecommendations/create* with a JSON payload in the request body that contains a new system nutrient recommendation object, the controller successfully performs the actions to save new system nutrient recommendation in the database.

- **FFQR-2-30 Implement Java API service and repository to calculate nutrients recommendations (Back-End)**

As a developer, I want to implement a *SysNutrientRecommendation* service and a *SysNutRecommendation* repository in the food item API, to handle the database interactions related to the nutrient recommendations.

- *Acceptance Criteria:*

1. A *SysNutrientRecommendation* service and a *SysNutRecommendation* repository are implemented and properly handle the database interactions related to nutrients recommendations.
2. The *SysNutrientRecommendation* service includes the following database actions:
 - *findAll()*
 - *findBynutrientName(sysNutrientName)*
 - *save(sysNutrientRecommendation)*

- **FFQR-2-31 Implement Angular web service to handle nutrients recommendations requests (Front-End)**

As a developer, I want to implement a service in the Angular web application that handle the HTTP requests to the Java API endpoint and map the results to the proper *FFQNutrientsRecommendations* model.

- *Acceptance Criteria:*

1. A *NutrientsRecommendationsService* is implemented in the Angular web application that handle the GET requests to the endpoint *http://localhost:9090/ffq/nutrientsrecommendations/calculate/{questionnaireID}*
2. The *NutrientsRecommendationsService* successfully map the JSON object that is returned in the HTTP request body to a *FFQNutrientsRecommendations* object.

- **FFQR-2-32 Create mockup for nutrients recommendations (Design)**

As a developer, I want to have a visual representation of the "Nutrients Recommendations" page UI flow should look like.

- *Acceptance Criteria:*
 1. A "Nutrients Recommendations" page UI mockup is created and approved by PO in order to start its implementation.
- **FFQR-2-33 Wire the nutrients recommendations component with the nutrients recommendations service (Front-End)**

As a user, I want to be able to select each recommendation for each questionnaire submitted.

 - *Acceptance Criteria:*
 1. A "Nutrients Recommendations" page is implemented and accessible from the Admin Portal where the user can see the recommendations for nutrients intake.
 2. The "Nutrients Recommendations" page shows tags for each patient that clearly indicate either the nutrients intake is above or below the recommended values.
- **FFQR-2-34 Implement nutrients recommendations Angular component (Front-End)**

As a developer, I want to implement an Angular component to hold all actions and attributes corresponding to the nutrient recommendations.

 - *Acceptance Criteria:*
 1. A model template and component are implemented, and it has all the methods required by the nutrients recommendations HTML template to work properly.
- **FFQR-2-35 Implement nutrients recommendations Angular HTML template (Front-End)**

As a user, I want to be able to view the recommendations per questionnaire for nutrients.

 - *Acceptance Criteria:*
 1. The "Nutrients Recommendations" page includes the amounts being taken by the patient, the amounts recommended for the patient and the tag indicating *Normal*, *Below* or *Above*.
 2. “*Normal*” label should be colored green, and “*Above*” and “*Below*” labels are red and brown respectively.
- **FFQR-2-36 Add DB sys_nutrients_recommendations collection population on food_item_service start**

As a developer, I want the sys_nutrients_recommendations Mongo collection to be populated from a JSON file that contains recommended nutrients amounts based on infant age and it was built after the excel spreadsheet provided by PO.

 - *Acceptance Criteria:*
 1. When the food item service is started, the application loads the system nutrient recommendations from the JSON file (SysNutrientsRecommendationsPayload.json) and creates/populates the sys_nutrients_recommendations Mongo collection.
- **FFQR-2-37 Implement Java API FoodRecommendation controller (Back-end)**

As a developer, I want to implement a FoodRecommendation controller (endpoint) in the food item API, to handle HTTP requests related to the food recommendations.

 - *Acceptance Criteria:*
 1. A *FoodRecommendationController* is implemented to serve HTTP requests on *http://localhost:9090/foodrecommendations* endpoint.
 2. When I do a GET request to endpoint *http://localhost:9090/foodrecommendations/calculate/{questionnaireID}*, the controller calculates food recommendations based on the questionnaire responses of the questionnaire passed and the system food recommendations baseline.

3. When I do a POST request to endpoint *http://localhost:9090/foodrecommendations/create* with a JSON payload in the request body that contains a new system food recommendation object, the controller successfully performs the actions to save new system food recommendation in the database.
- **FFQR-2-38 Add DB sys_food_recommendation collection population on food_item_service start**
As a developer, I want the sys_food_recommendations Mongo collection to be populated from a JSON file that contains recommended food amounts based on infant age and it was built after the excel spreadsheet provided by PO.
 - *Acceptance Criteria:*
 1. When the food item service is started, the application loads the system food recommendations from the JSON file (SysFoodRecommendationsPayload.json) and creates/populates the sys_food_recommendations Mongo collection.
 - **FFQR-2-39 Implement JAVA API models for food recommendations - (Back-End)**
As a developer, I want to add the proper models in the food item microservice (Java API), to handle and map food recommendation objects.
 - *Acceptance Criteria:*
 1. A *FoodCategoryRecommendation* model is added to the food item microservice and includes the following fields:
 - *categoryName*
 - *rangeFrom*
 - *rangeTo*
 - *label*
 - *calculatedAmount*
 3. A *FoodRecommendationRange* model is added to the food item microservice and includes the following fields:
 - *from*
 - *to*
 - *label*
 4. A *SysFoodRecommendation* model is added to the food item microservice and includes the following fields:
 - *categoryName*
 - *unit*
 - *recommendationsByAge* (a map of age range to *FoodRecommendationRange* objects)
 5. A *FoodRecommendation* model is added to the food item microservice and includes the following fields:
 - *questionnaireId*
 - *patientName*
 - *patientAgeInMonths*
 - *foodCategoryRecList* (a list of *FoodCategoryRecommendation* objects)
 - **FFQR-2-40 Implement JAVA API service for food recommendations calculations (Back-End)**
As a developer, I want to implement a FoodRecommendation service in the food item API, to handle the database interactions related to the food recommendations calculations.
 - *Acceptance Criteria:*
 1. A *FoodRecommendation* service is implemented and properly handle the database interactions related to food recommendations.
 2. The *FoodRecommendation* service includes the following database actions:

- *findAll()*
- *findByFoodName(FoodName)*
- *save(FoodRecommendation)*

- **FFQR-2-41 Implement Angular web service to handle foods recommendations requests (Front-End)**

As a developer, I need a service in the Angular web application that handle the HTTP requests related to the food recommendations.

- *Acceptance Criteria:*

1. All JSON objects that are sent from the back-end are properly mapped inside the angular component.
2. All POST requests have an error catch properly implemented (error passed and error handled in angular components).

- **FFQR-2-42 Implement modal component to view foods recommendations in Angular application (Front-End)**

As a user, I want to be able to see the food recommendations based on the questionnaires calculations.

- *Acceptance Criteria:*

1. All subcategories of food items should be displayed in a table view.
2. All categories must the five types of labels: below, adequate, little below , little above and above.
3. Ranges from and range to must be displayed in conjunction with the calculated amount.

- **FFQR-2-43 Add "category" field to FoodType model and FoodItem payload (Back-end)**

As a user, I want the food items to be grouped by food category in a way that the food recommendations can be generated based on the food categories.

- *Acceptance Criteria:*

1. A "category" field is added to the FoodItem payload and each food item is assigned a corresponding food category according to excel spreadsheet provided by PO.
2. A "category" field is added to the FoodType model:

```
@JsonProperty("category")
private String category;
```

- **FFQR-2-44 Add validation on add/update food items to avoid duplicated nutrient IDs (Back-end)**

As a user, I want to get a proper message if I try to add a new food item with a nutrient ID that already exist instead of error.

- *Acceptance Criteria:*

1. When I try to enter a new food item or update an existing food item with a nutrient ID that already exists in the database, the application should properly handle it by notifying on the issue instead of throwing an error.

- **FFQR-2-45 Implement food recommendations model in Angular application (Front-End)**

As a developer, I want to be able to map all JSON that comes from the end points (back-end) into the angular components involved.

- *Acceptance Criteria:*

1. Objects implemented in the front-end must be handled in the same as the response incoming.
2. Must include/hold: Questionnaire ID, patient name, infant age, user choices and food items recommendation lists.
3. Inside the food item recommendation should be mapped the following: food item name, range from and range to, label and user calculations.

- **FFQR-2-46 Implement validation in add/update food items forms (Front-End)**

As a user, I want to be able to input the food item information without errors and without leaving any required field empty.

- *Acceptance Criteria:*

1. When update food items: name, servings, nutrients, and food item id must not be empty.
2. When adding food items: name, servings, nutrients, and food item id must not be empty.
3. If when adding food item, the same item is added by mistake, the web application should not let me add the repeated food item by stopping user action and displaying an error message.

- **FFQR-2-47 Fix CSS in questionnaire page to improve appearance**

As a user, I want the boxes that hold food items in the questionnaire page to look uniform and have the same size.

- *Acceptance Criteria:*

1. When I navigate to the Questionnaire page and access to a questionnaire, I see that the boxes that hold the food items look uniform (they all have the same size).

- **FFQR-2-48 Refactor nutrients recommendations calculations to calculate/include percentage of recommended value (Back-end)**

As a user, I want to see in the nutrient's recommendations, what percentage is the calculated value of the recommended value instead of just the Above, Normal, Below labels.

- *Acceptance Criteria:*

1. When I navigate to the "Recommendations" page in the Admin Portal and click in the flask icon to view the nutrients recommendations for a particular questionnaire, I should be able to see, in the Indicator column the percentage that the calculated amounts represent based on the recommended value besides the Below, Normal and Above labels.

- **FFQR-2-49 Refactor nutrients recommendations view to show colors based on new indicators that include percentages**

As a user, I want to see in the nutrient's recommendations, what percentage is the calculated value of the recommended value.

- *Acceptance Criteria:*

1. Nutrients consumption labeled with *normal* should have text in green.
2. Nutrients consumption labeled with *below* should have text in red.
3. Nutrients consumption labeled with *above* should have text in brown.

- **FFQR-2-50 Containerize Services, Mongo, and Web Application**

As a product owner, I want a simple way to launch both services and web application without the need to install various tools and or command line knowledge.

- *Acceptance Criteria:*

1. The admin portal must be deployed using dockers containers in the PO's computer.
2. Only must need docker-compose build, to build the image and docker-compose up to spin up the whole environment. (web application, java services and database).
3. PO must access it through <http://localhost:4200/admin> in a web browser.

- **FFQR-2-51 Regression Testing of v1.0 features**

As a QA Engineer, I want to verify that all features implemented in version 1.0 works the same way as when the product was initially given to our team.

- *Acceptance Criteria:*

1. The Questionnaire landing page can be accessed through the URL <http://localhost:4200>.
2. When a valid questionnaire id is entered, the application redirects to the view where the questionnaire form is.
3. The Questionnaire form allows to enter the user choices and boxes corresponding to food items that the user does not want to fill out can be hidden by clicking on the close icon on the right corner of the box.
4. When clicking on the Submit button, the questionnaire responses are submitted and saved in the database.
5. When the database is verified, all user choices and nutrient breakdown corresponding to the submitted questionnaire are present in the database.

- **FFQR-2-T01 Validate ffq/allfoodsnutrients endpoint**

As a QA Engineer, I want to verify that the ffq/allfoodsnutrients endpoint successfully handle the retrieval of all food items and their corresponding list of nutrients.

- *Acceptance Criteria:*

1. When I do a GET HTTP request through Postman to the <http://localhost:9090/ffq/allfoodsnutrients>, a JSON payload is returned in the HTTP response body that contains a list of all food items and corresponding nutrients existing in the database.

- **FFQR-2-T02 Validate ffq/createfoodnutrients endpoint**

As a QA Engineer, I want to verify that the ffq/createfoodnutrients endpoint successfully handle addition of new food items and their corresponding lists of nutrients.

- *Acceptance Criteria:*

1. When I do a POST HTTP request through Postman to the following URL:
<http://localhost:9090/ffq/allfoodsnutrients> that contains JSON payload with a new food item object in HTTP request body, the new food item object is properly saved into the database.

- **FFQR-2-T03 Validate ffq/updatefoodnutrients endpoint**

As a QA Engineer, I want to verify that the ffq/createfoodnutrients endpoint successfully handle the updating of existing food items and their corresponding lists of nutrients.

- *Acceptance Criteria:*

1. When I do a PUT HTTP request through Postman to the following URL:
<http://localhost:9090/ffq/updatefoodnutrients> that contains JSON payload with an existing food item object in HTTP request body, the existing food item object is properly updated into the database with the new data.

- **FFQR-2-T04 Validate ffq/results/all endpoint**

As a QA Engineer, I want to verify that the ffq/results/all endpoint successfully handle the retrieval of all questionnaire results.

- *Acceptance Criteria:*

1. When I do a GET HTTP request through Postman to the <http://localhost:9090/ffq/results/all>, a JSON payload is returned in the HTTP response body that contains a list of all questionnaire results existing in the database.

- **FFQR-2-T05 Validate ffq/foodrecommendations/calculate/{questionnaireID} endpoint**

As a QA Engineer, I want to verify that the ffq/foodrecommendations/calculate/{questionnaireID} endpoint successfully perform the calculations of food recommendations.

- *Acceptance Criteria:*

1. When I do a GET HTTP request through Postman to the following URL:
`http://localhost:9090/ffq/foodrecommendations/calculate/{questionnaireID}`, a JSON payload is returned in the HTTP response body that contains the calculations of food recommendations in a FoodRecommendation object.

- **FFQR-2-T06 Verification of food intake recommendations**

As a QA Engineer, I want to be able to observe the food recommendations intake and have accuracy for its calculations and parameters.

- *Acceptance Criteria:*

1. Food intake recommendations will be displayed in its own modal.
2. Modal displays calculated amounts , recommended amounts and visual aid for the user, little above, little below, normal, below and above.
3. Different test cases are selected:
 - a. Two general test cases for manual testing
 - b. Five group test cases for manual testing with the following food groups: **Breast milk/Formula/Cow's Milk/Other milks, Whole Grains, Proteins, 100% Juices and Sugary Beverages.**

All manual cases have exactly the same values as calculated by the web application.

- **FFQR-2-T07 Food item delete option functionality testing**

As a QA Engineer, I want to be able to delete a food item from the food item page in the admin portal.

- *Acceptance Criteria:*

1. Admin portal and food item service is running.
2. When click on the trash icon on the food item portal a verification modal should be displayed asking for user confirmation to delete the item.
3. On confirmation modal user can cancel the delete confirmation or proceed with the deletion of the food item.
4. Food item page is reloaded and updated the food item list.

- **FFQR-2-T08 Questionnaire results page shows results per questionnaires submitted**

As QA Engineer, I want to see the questionnaires results when click on the results button.

- *Acceptance Criteria:*

1. Admin portal, food item service and questionnaire service is running.
2. When click on Questionnaire results page , a list of questionnaires submitted must be showed. In case of no questionnaire submissions a "No questionnaire have been submitted yet!" message is displayed.
3. When click on results button on each questionnaire submitted, a list of user choices, the questionnaire ID, Patient name and, nutrients calculations must be displayed.

- **FFQR-2-T09 Add/Update food item form validations**

As a QA Engineer, I want to be able to add/update food items and its nutrients with all required values and no empty fields in order to avoid errors.

- *Acceptance Criteria:*

1. Admin portal and food item service is running.
2. Food name and/or serving and/or food category and/or ID fields are left empty by the user.
3. User is not able to click add new food item or update food item if any of the fields mentioned before are empty.
4. When all required fields are filled out the add food item/ update food item is highlighted and user can submit changes.

5. New food item is displayed in the food item page and updates are reflected for the food item that was updated.
- **FFQR-2-T10 Recommendation page shows nutrients and food intake recommendations**
As a QA Engineer, I want to be able to see the recommendations for food intake and nutrient consumption.
 - *Acceptance Criteria:*
 1. Admin portal and food item service and questionnaire service is running
 2. When click on Recommendation page a list of Questionnaire submitted must be displayed in table, each row must have associated a fork and knife icon to display the food intake recommendations and an flask icon to display the nutrients recommendation.
 3. Nutrients recommendation modal must show nutrients name, recommended values, calculated values and the labels *above*, *below* or *normal*. Label *normal* must be in green , *below* in red and *above* in brown.
 4. Food intake recommendation modal must show food item category names, recommended range for food intake, label aid for the user and calculated values.
 5. Labels are: *above*, *below*, *normal* , *little above* and *little below*.

Pending User Stories

- **Include food item “category” in add/update operations**
Description/Explanation: The “category” field for FoodType was added later in the semester and for time constraints the team was not able to include this new field in the Add Food Item and Update Food Item functionalities. When the administrator navigate to the Add Food Item / Update Food Item pages, it should be able to see a field corresponding to the food category as well as able to add a category to a new food item being added or update the category of an existing food item.
- **Automate testing for nutrient and food recommendations**
Description/Explanation: Automated test cases should be implemented to validate the amount generated for the food and nutrients recommendations.

PROJECT PLAN

The following section describes the planning conducted, following an agile development approach, for the execution of this project. The details of each planned sprint are detailed and the stories comprised on each one of them. Also, the hardware and software resources used for this project are specified.

Hardware and Software Resources

The following hardware and software resources were used for the execution of this project.

Hardware

Laptop 1

Operating System: Windows 10 Enterprise 64-bit (10.0, Build 15063) (15063.rs2_release.170317-1834)
 Processor: Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz (8 CPUs), ~2.9GHz
 Memory: 16384MB RAM
 Hard drive: Intel SATA 512 GB

Laptop 2

Operating System: Windows 10 Home 64-bit (10.0, Build 17763)

Processor: Intel(R) Core(TM) i7-8565HQ CPU @ 1.9GHz (8 CPUs), ~2.0GHz

Memory: 16384MB RAM

Hard drive: Intel SATA 512 GB

Software

Apache Tomcat 8.0.27

Spring-Boot 2.0

Java (build 1.8.0_201-b09)

Git Version 2.23.0 windows1

Balsamiq MockUps3

Visual Studio Code 1.39.2

MongoDB 4.2.0 2008R2Plus SSL

MongoDB Compass Community

Node JS 10.16.3

Postman 7.10.0

Docker Desktop version 19.03.1

Docker ToolBox

SPRINTS PLAN

This chapter describes the main goals of each sprint and list the tasks/stories that were completed during each sprint. The description and acceptance criteria for stories was previously described in Chapter 1.

Sprint 1

Time Frame: 09/09/2019 - 09/23/2019

During this first sprint the main goal was to explore the version 1.0 of the product to define its actual scope, identify possible blockers for the implementation of version 2.0, get familiar with the frameworks and technologies used in the previous version, set up the development local environments and run the version 1.0 locally.

Tasks completed in Sprint 1

TASK-01 Install and set up required frameworks and tools

TASK-02 Watch Angular, MongoDB and Spring Boot tutorials

TASK-03 Set up and run v1.0 in local environment

TASK-04 Explore actual scope for v1.0

Stories completed in Sprint 1

FFQR-2-01 Result Class Diagram

FFQR-2-02 Save Questionnaire Results

FFQR-2-03 Retrieve all questionnaire results

FFQR-2-04 View "Success" message on questionnaire submission

FFQR-2-05 Include questionnaire ID when submit answer

Sprint 2

Time Frame: 09/23/2019 - 10/07/2019

During this second sprint we focused on implementing the functionalities that would allow the system administrator to view all existing food item/nutrients and perform CRUD (create, update, delete) operations on them. After version 1.0, the only way to manipulate/access this data was by accessing the MongoDB documents (database) and risking that any changes in the documents (MongoDB tables) structure would break the entire application. This approach was not suitable for users since require a higher level of expertise than the one a regular user would have.

Stories completed in Sprint 2

FFQR-2-06 Create Admin Page Mockup (Design)

FFQR-2-07 Add Admin Page routing (Front-End / UI)

FFQR-2-08 Implement end-point to return all food items and nutrients (Back-End)

FFQR-2-09 Food Items View (Front-End)

FFQR-2-10 Add foods/nutrients (Front-End / UI)

FFQR-2-11 Update foods/nutrients (Front-End / UI)

FFQR-2-12 Delete foods/nutrients (Front-End / UI)

FFQR-2-13 Add foods/nutrients (Back-End / API)

FFQR-2-14 Update foods/nutrients (Back-End / API)

FFQR-2-15 Delete foods/nutrients (Back-End / API)

Sprint 3

Time Frame: 10/07/2019 - 10/21/2019

During this sprint we addressed an issue that arise as a result of the research of v1.0's scope conducted during Spring 1. We refactored the existing questionnaire-service code to include the patient age in the questionnaires since this information is required for the nutrients/foods intake recommendations that would be implemented during Spring 4 and 5. We also implemented the "Results" view and corresponding API endpoints to allow the system administrator to access the responses and results for the submitted questionnaires. We spent another portion of the sprint fixing some defects encountered during Spring 2: the manipulation and display of multiple servings for some of the food items and the handling of food items that have more than one food type and nutrients list.

Stories completed in Sprint 3

FFQR-2-16 View all questionnaire results (Front-end).

FFQR-2-19 REFACTOR: Add "infant age" field in questionnaire form (Front-end).

FFQR-2-20 Create results model and service in web app (Front-end).

FFQR-2-21 View all questionnaire results (Back-end).

FFQR-2-22 REFACTOR: API to handle new field added to DB "PortionSize".

FFQR-2-23 REFACTOR: Add "infant age" field in questionnaire and food item services. (Back-end)

FFQR-2-24 Validate result view when there are no questionnaire submissions to display.(Front-end)

FFQR-2-17 Implement in the add / update views the handling of "servings" (Front-end).

FFQR-2-18 Implement in add / update view the handling of "multiples food types" (Front-end).

FFQR-2-25 Display success message upon successful submission/update of new food item. (Front-end)

FFQR-2-26 Display total calories in questionnaire results page.(Front-end)

Sprint 4

Time Frame: 10/21/2019 - 11/04/2019

During the sprint 4, the main goal was to implement the calculations of nutrient recommendations. Such implementation relies on the nutrient breakdown calculation done on version 1.0 (prior semester) and a spreadsheet provided by the product owner where a recommended amount of nutrients is given based on the age of the infant. The results were displayed in section of the Admin Portal in a table that indicates if the nutrients values calculated from the questionnaire responses are either “Above”, “Normal” or “Below” the recommended values.

Stories completed in Sprint 4

FFQR-2-27 Redesign database schema to include nutrient recommendations table

FFQR-2-28 Implement JAVA API models for nutrients recommendations (Back-End)

FFQR-2-29 Implement JAVA API controller to handle nutrients recommendations - (Back-End)

FFQR-2-30 Implement JAVA API service and repository to calculate nutrients recommendations - Back-End

FFQR-2-31 Implement Angular web service to handle nutrients recommendations requests (Front-End)

FFQR-2-32 Create mockup for nutrients recommendations (Design)

FFQR-2-33 Wire the nutrients recommendations component with the nutrients recommendations service (Front-End)

FFQR-2-34 Implement nutrients recommendations Angular component (Front-End)

FFQR-2-35 Implement nutrients recommendations Angular HTML template (Front-End)

FFQR-2-36 Add DB sys_nutrients_recommendations table population on food_item_service start

FFQR-2-37 Implement FoodRecommendation Controller (Back-end)

Sprint 5

Time Frame: 11/04/2019 - 11/18/2019

Sprint 5 focused on the implementation of the food recommendations. For food recommendations the food items were grouped into 10 categories as specified in spreadsheet provided by Product Owner. Once grouped, the servings amount user submit in the questionnaire for all foods in each category were added to calculate a recommendation based on the infant age.

Stories completed in Sprint 5

FFQR-2-37 Implement FoodRecommendation Controller (Back-end)

FFQR-2-38 Add DB sys_food_recommendation table population on food_item_service start

FFQR-2-39 Implement JAVA API models for food recommendations - (Back-End)

FFQR-2-40 Implement JAVA service for food recommendations - (Back-End)

FFQR-2-41 Implement Angular web service to handle foods recommendations requests (Front-End)

FFQR-2-42 Implement modal component to view foods recommendations in Angular application (Front-End)

FFQR-2-43 Add "category" field to FoodType model and FoodItem payload (Back-end)

FFQR-2-44 Add validation on add/update food items to avoid duplicated nutrient IDs (Back-end)

FFQR-2-45 Implement food recommendations model in Angular application (Front-End)

FFQR-2-46 Implement validation in add/update food items forms (Front-End)

FFQR-2-47 Fix CSS in questionnaire page to improve appearance

Sprint 6

Time Frame: 11/18/2019 - 12/02/2019

Sprint 6 mainly focused on wrapping up the project by implementing a few last minute product owner requests, testing the application and working on the deployment of the application through Docker containerization.

Stories completed in Sprint 6

FFQR-2-48 Refactor nutrients recommendations calculations to calculate/include percentage of recommended value (Back-end)

FFQR-2-49 Refactor nutrients recommendations view to show colors based on new indicators that include percentages

FFQR-2-50 Containerize Services, Mongo, and Web Application

FFQR-2-51 Regression Testing of v1.0 features

FFQR-2-T01 Validate ffq/allfoodsnutrients endpoint

FFQR-2-T02 Validate ffq/createfoodnutrients endpoint

FFQR-2-T03 Validate ffq/updatefoodnutrients endpoint

FFQR-2-T04 Validate ffq/results/all endpoint

FFQR-2-T05 Validate ffq/foodrecommendations/calculate/{questionnaireID} endpoint

FFQR-2-T06 Verification of food intake recommendations

FFQR-2-T07 Food item delete option functionality testing

FFQR-2-T08 Questionnaire results page shows results per questionnaires submitted

FFQR-2-T09 Add/Update food item form validations

FFQR-2-T10 Recommendation page shows nutrients and food intake recommendations

SYSTEM DESIGN

This section comprises the information related to the design of version 2.0 of the Food Frequency Questionnaire. The architecture patterns are outlined and explained. The entire system is shown in a package diagram and the subsystems are explained. Finally, the design patterns used in the project are discussed.

Architectural Patterns

Since this is a second iteration of an existing product the same architectural patterns used in version 1.0 were used for the current version. This project respects the *Microservice Architecture* and the traditional *Client-Server* model designed and implemented in version 1.0. Two micro-services implemented in Java using the Spring Boot framework were inherited from previous version and an Angular web application. The Java microservices handle all the business logic on the back-end while the Angular application provides the user interface and is responsible for all the user interactions. For this project the major development was done on the food item microservice and the Angular web application. Only small changes were done on the questionnaire microservice mainly to add pieces of functionality that were left out in the previous version and were critical for the implementation of the current one, such as the inclusion of the infant age in the questionnaire.

Design Patterns

Singleton Pattern : Utilized for the Angular service components which make the API calls to the backend microservices

Transfer Object Pattern: Used to transfer data from backend services to questionnaire client. Aside from model object, such as FFQItem we included FFQItemResponse which contains all the attributes being sent from server which we can then manipulate to get the desired model object. Inversely we'd use FFQItemRequest as the DTO to send data back to the server where it can manipulate it as necessary to fit its core domain object. In this case Response and Request served as our DTOs.

System and Subsystem Decomposition

The diagram below was updated from the diagram delivered in version 1.0 to include what was implemented in this project: the Admin Web Portal. No major changes were done on the overall architecture of the product.

FFQR System Architecture

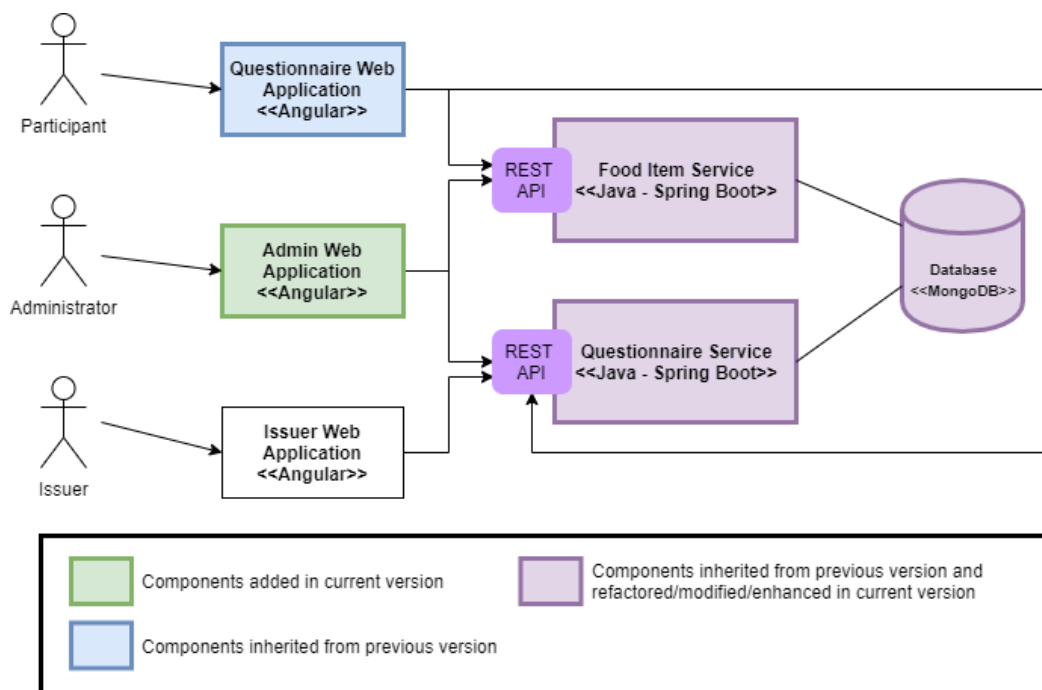


Figure 1. System Architecture

SYSTEM VALIDATION

Regression Testing

Regression testing was conducted to ensure what was implemented in version 1.0 was not broken and all functionalities were working as before we start development of version 2.0. A small set of test cases from v1.0 documentation were executed and are described next.

Test case ID: FFQR-2-RT01	
Description	Verify the questionnaire page can be viewed through a browser by navigating to URL http://localhost:4200 . Pre-condition: Angular web application should be running on localhost, port 4200
Expected Results	- When following link (http://localhost:4200), questionnaire web page should be viewed in the browser with a text field and a “Begin Questionnaire” button.
Actual Results	The landing page of the Questionnaire Page shows up in the browser.
Status	Pass

Test case ID: FFQR-2-RT02	
Description	Verify Questionnaire Form page can be accessed by entering a valid questionnaire id in the Questionnaire main page. Preconditions: Valid-1 is a valid questionnaire id existing in the database that has not been completed yet.
Expected Results	- When valid-1 is entered in the questionnaire id field and the “Begin Questionnaire” button is clicked, the application should redirect to the Questionnaire Form page.
Actual Results	The Questionnaire Form is shown.
Status	Pass

Test case ID: FFQR-2-RT03	
Description	Verify the Questionnaire Form page allows the user to enter the user choices and boxes corresponding to food items that the user does not want to fill out can be hidden by clicking on the close icon on the right corner of the box.
Expected Results	- When the user arrives to the Questionnaire Form page is able to select it choices and able to hide the boxes corresponding to food item that does not want to complete.
Actual Results	All functionalities behave as expected
Status	Pass

Test case ID: FFQR-2-RT04	
Description	Verify that the Submit functionality behaves as expected in the Questionnaire Form page.
Expected Results	- When the user finishes completing the questionnaire and click the Submit button, the application successfully saves the responses and nutrient breakdown in the database.
Actual Results	The user responses and nutrient breakdown are saved in the database
Status	Pass

Test case ID: FFQR-2-RT05	
Description	Verify that the nutrient breakdown calculations are correct.
Expected Results	- After the submission of a questionnaire, the application generates a nutrient breakdown that should match the manual calculations
Actual Results	The nutrient breakdown matches the manual calculations
Status	Pass

Test case ID: FFQR-2-RT006	
Description	Verify “Skip Additional Foods” button works as expected
Expected Results	- When the user is completing a questionnaire, it should have the option to click on the “Skip Additional Foods” button and hide all the additional foods.
Actual Results	The additional foods hide when the “Skip Additional Foods” button is clicked.
Status	Pass

Testing of Current Version

The following test cases were implemented with the purpose of validate the functionalities added as part of the implementation of the current project (version 2.0).

Test case ID: FFQR-2-T01	
Description	Verify the ffq/allfoodsnutrients endpoint in the food_item API
Expected Results	- When I do a GET HTTP request through Postman to the http://localhost:9090/ffq/allfoodsnutrients, a JSON payload is returned in the HTTP

	response body that contains a list of all food items and corresponding nutrients existing in the database
Actual Results	The correct JSON payload is received
Status	Pass

Test case ID: FFQR-2-T02	
Description	Verify the ffq/createfoodnutrients endpoint in the food_item API
Expected Results	- When I do a POST HTTP request through Postman to the following URL: http://localhost:9090/ffq/allfoodsnutrients that contains JSON payload with a new food item object in HTTP request body, the new food item object is properly saved into the database
Actual Results	The new food item is saved in the database
Status	Pass

Test case ID: FFQR-2-T03	
Description	Verify ffq/updatefoodnutrients endpoint in the food_item API
Expected Results	- When I do a PUT HTTP request through Postman to the following URL: http://localhost:9090/ffq/updatefoodnutrients that contains JSON payload with an existing food item object in HTTP request body, the existing food item object is properly updated into the database with the new data.
Actual Results	The food item is updated in the database
Status	Pass

Test case ID: FFQR-2-T04	
Description	Verify ffq/results/all endpoint in the food_item API
Expected Results	- When I do a GET HTTP request through Postman to the http://localhost:9090/ffq/results/all, a JSON payload is returned in the HTTP response body that contains a list of all questionnaire results existing in the database.
Actual Results	The correct JSON payload is received

Status	Pass
--------	------

Test case ID: FFQR-2-T05	
Description	Verify ffq/foodrecommendations/calculate/{questionnaireID} endpoint in the food_item API
Expected Results	- When I do a GET HTTP request through Postman to the following URL: http://localhost:9090/ffq/foodrecommendations/calculate/{questionnaireID}, a JSON payload is returned in the HTTP response body that contains the calculations of food recommendations in a FoodRecommendation object.
Actual Results	The correct JSON payload is received
Status	Pass

Test case ID: FFQR-2-T06	
Description	Manual testing of food item add and update component.
Expected Results	Food intake recommendations will be displayed in its own modal. Modal displays calculated amounts , recommended amounts and visual aid for the user, little above, little below, normal, below and above. All manual cases have exactly the same values as calculated by the web application.
Actual Results	All recommendations were displayed properly. All calculations were accurate.
Status	Pass

Test case ID: FFQR-2-T07	
Description	Manual testing of delete functionality.
Expected Results	When clicking on the trash icon on the food item portal a verification modal should be displayed asking for user confirmation to delete the item. On confirmation modal user can cancel the delete confirmation or proceed with the deletion of the food item. Food item page is reloaded and updated the food item list.
Actual Results	Clicked on trash can icon, delete warning showed. When press delete option the food item is deleted and page refreshed.
Status	Pass

Test case ID: FFQR-2-T08	
Description	Manual testing of Questionnaire results page functionality
Expected Results	When click on Questionnaire results page , a list of questionnaires submitted must be showed. In case of no questionnaire submissions a <i>"No questionnaire have been submitted yet!"</i> message is displayed. When click on results button on each questionnaire submitted, a list of user choices, the questionnaire ID, Patient name and, nutrients calculations must be displayed.
Actual Results	Questionnaires results were displayed, all user choices were shown and required information is displayed properly. When no questionnaire submitted, proper message was displayed.
Status	Pass

Test case ID: FFQR-2-T09	
Description	Manual Testing for add/update food item functionality
Expected Results	User is not able to click add new food item or update food item if any of the required fields. When all required fields are filled out the add food item/ update food item is highlighted and user can submit changes. New food item is displayed in the food item page and updates are reflected for the food item that was updated.
Actual Results	Add/Update Food Item button is not functional until all required fields are filled out. Food item was added successfully to the food item table.
Status	Pass

Test case ID: FFQR-2-T010	
Description	Manual testing of the Questionnaire page functionality
Expected Results	When click on Recommendation page a list of Questionnaire submitted must be displayed in table, each row must have associated a fork and knife icon to display the food intake recommendations and an fask icon to display the nutrients recommendation.
Actual Results	A list of questionnaire submitted were displayed properly with all related information as expected . All icons were displayed properly and all icons were working when clicked.

Status	Pass
--------	------

Test case ID: FFQR-2-T011	
Description	Manual testing of the nutrients recommendations modal
Expected Results	Nutrients recommendation modal must show nutrients name, recommended values, calculated values and the labels <i>above</i> , <i>below</i> or <i>normal</i> . Label <i>normal</i> must be in green , <i>below</i> in red and <i>above</i> in brown.
Actual Results	All values were displayed properly. All labels have the associated color and range as expected.
Status	Pass

Test case ID: FFQR-2-T013	
Description	Manual testing of the food item recommendations modal
Expected Results	Food intake recommendation modal must show food item category names, recommended range for food intake, label aid for the user and calculated values.
Actual Results	All values were displayed properly. All labels have the associated range as expected.
Status	Pass

GLOSSARY

Food item : An item in the questionnaire and all its options (servings, food types, sugar, etc.)

Questionnaire ID : Code provided by an Administrator to an Issuer and then given to a Participant to access the questionnaire. A valid questionnaire ID is one that exists in the questionnaire database and was not submitted previously.

Participant : A parent or guardian taking the questionnaire for an infant.

Issuer : A pediatrician or researcher.

Administrator : Dr Cristina Palacios, will be able to modify food items and its nutrients by using the food item page in the Admin portal.

Nutrients and Food item recommendations: A metric to assess the diet quality of questionnaire results, based on an algorithm provided by Dr. Palacios.

Primary food items: Water, breast milk, formula, and cereal added to milk. Many infants will only consume these items.

Secondary food items: All food items that are not primary. They can be collapsed in bulk and brought back if necessary.

APPENDIX

Appendix A UML Diagrams

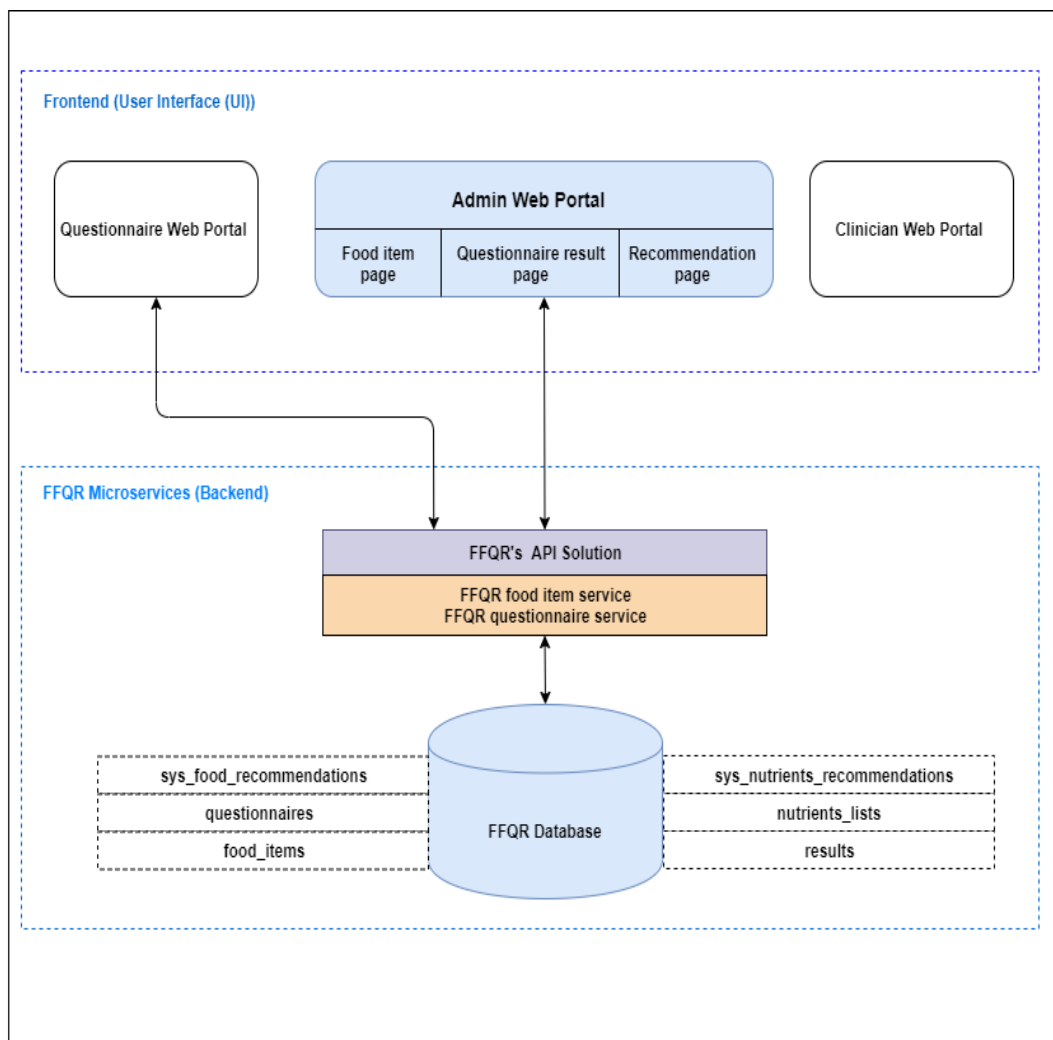


Figure 2. UML System Design Diagram

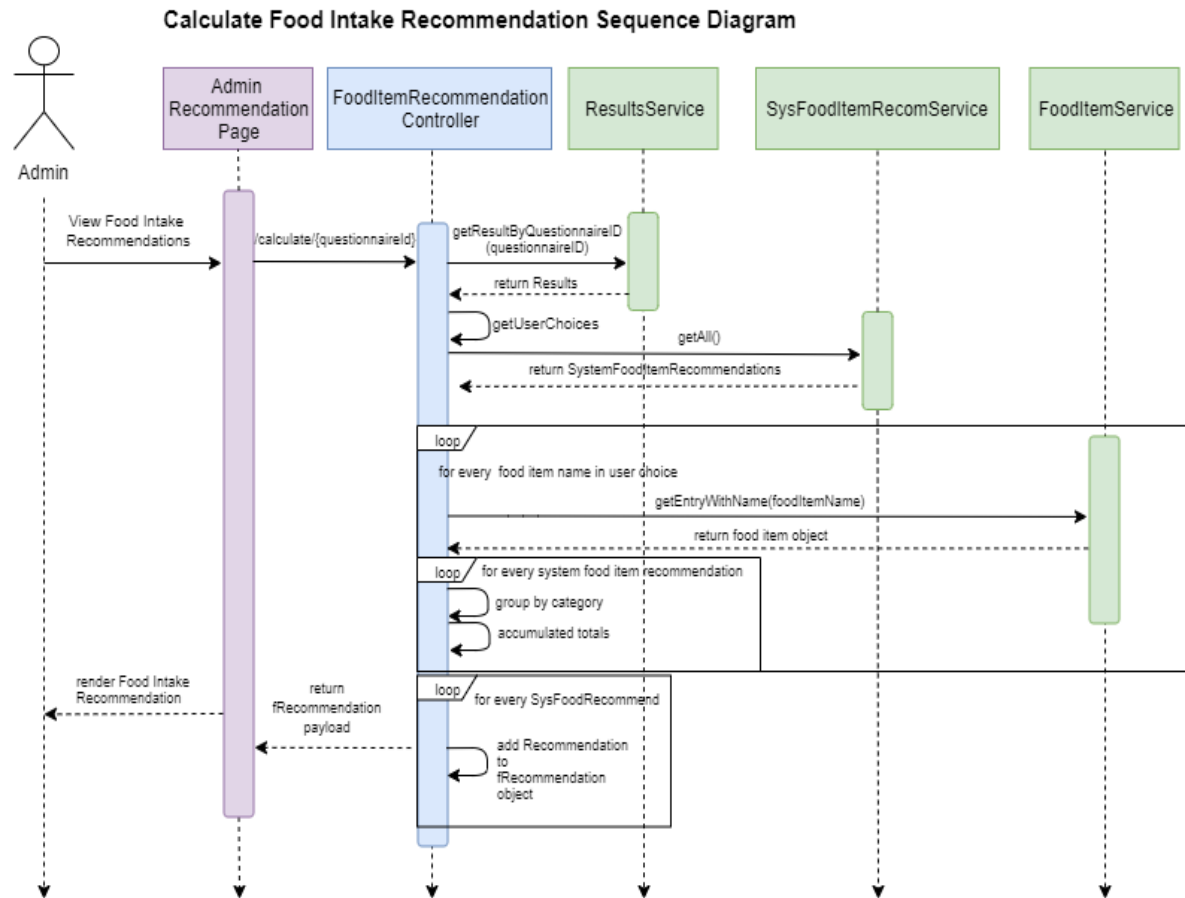


Figure 3. UML Sequence Diagram for Calculate Food Intake Recommendations

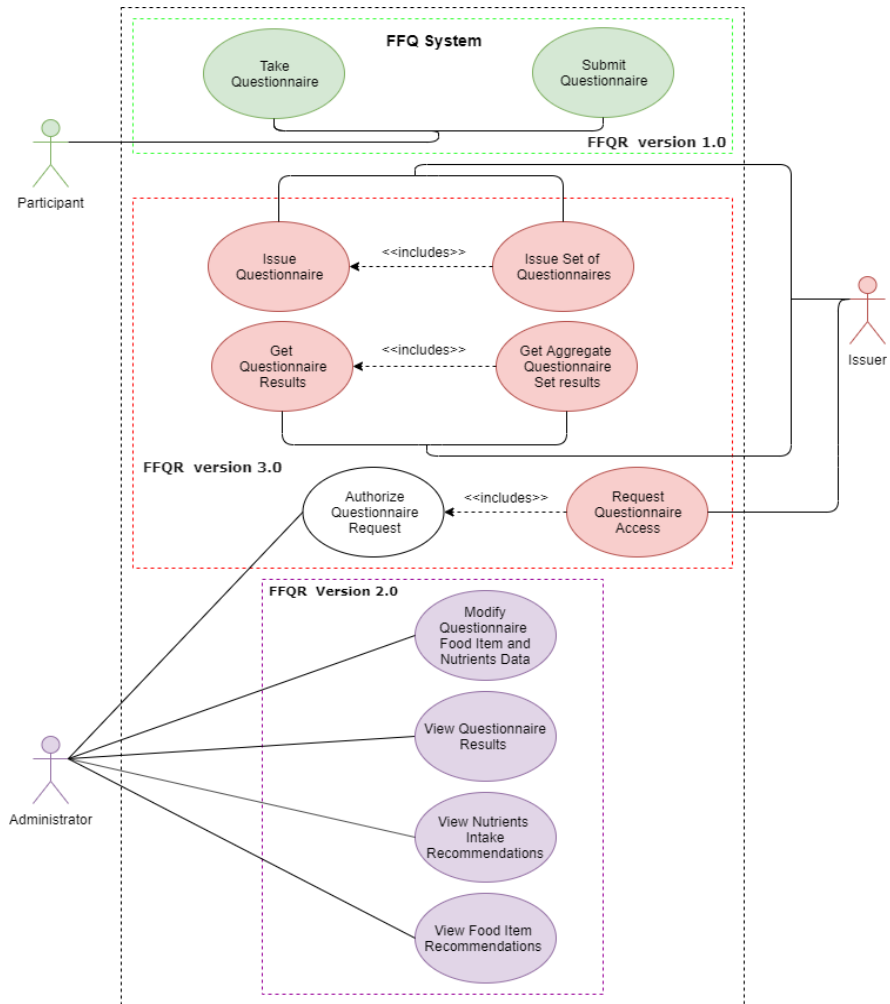


Figure 4. UML Diagram for FFQR System

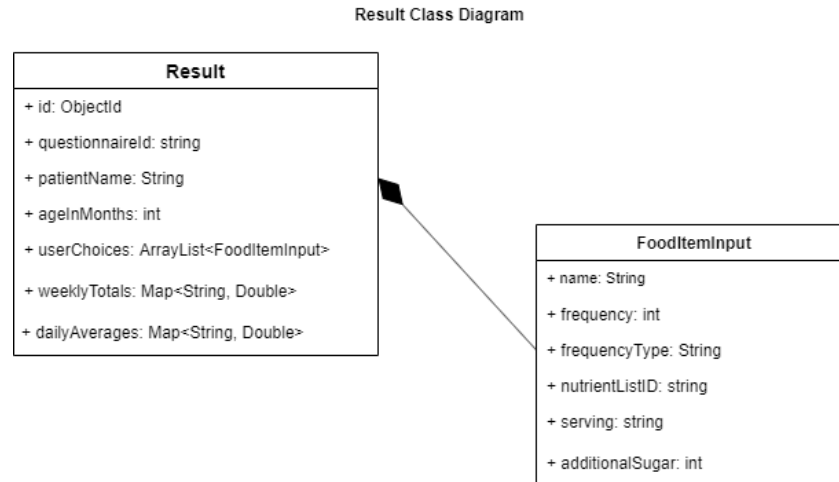


Figure 5. UML Result Class diagram

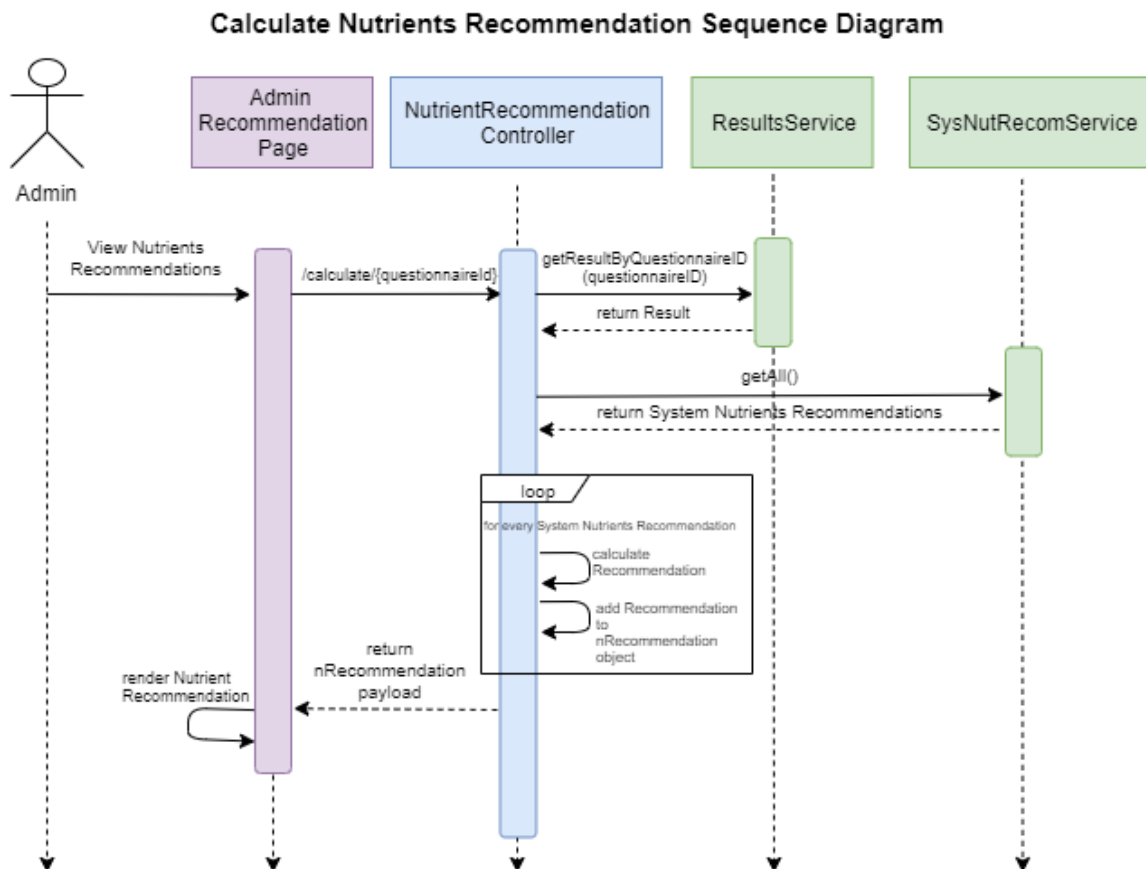


Figure 6. Calculate Nutrients Recommendation Sequence Diagram



Milk with chocolate/strawberry or vanilla (include frozen beverages, shakes, hot chocolate, etc.) Frequency * times per Frequency type *

Servings *

Sugar added: Extra Sugar tsp(s)



Other milks (example: soy, rice, almond, goat, etc.) Frequency * times per Frequency type *

Servings *



Cheese Frequency * times per Frequency type *



Ice Cream Frequency * times per Frequency type *

Figure 7. Questionnaire view “before” fixing CSS

Milk with chocolate/strawberry or vanilla (include frozen beverages, shakes, hot chocolate, etc.) 

Frequency * time(s) per Frequency type *

Servings *

Sugar added: Extra Sugar tsp(s)

Other milks (example: soy, rice, almond, goat, etc.) 

Frequency * time(s) per Frequency type *

Servings *

Cheese 

Frequency * time(s) per Frequency type *

Ice Cream 

Frequency * time(s) per Frequency type *

Figure 8. Questionnaire view “after” fixing CSS

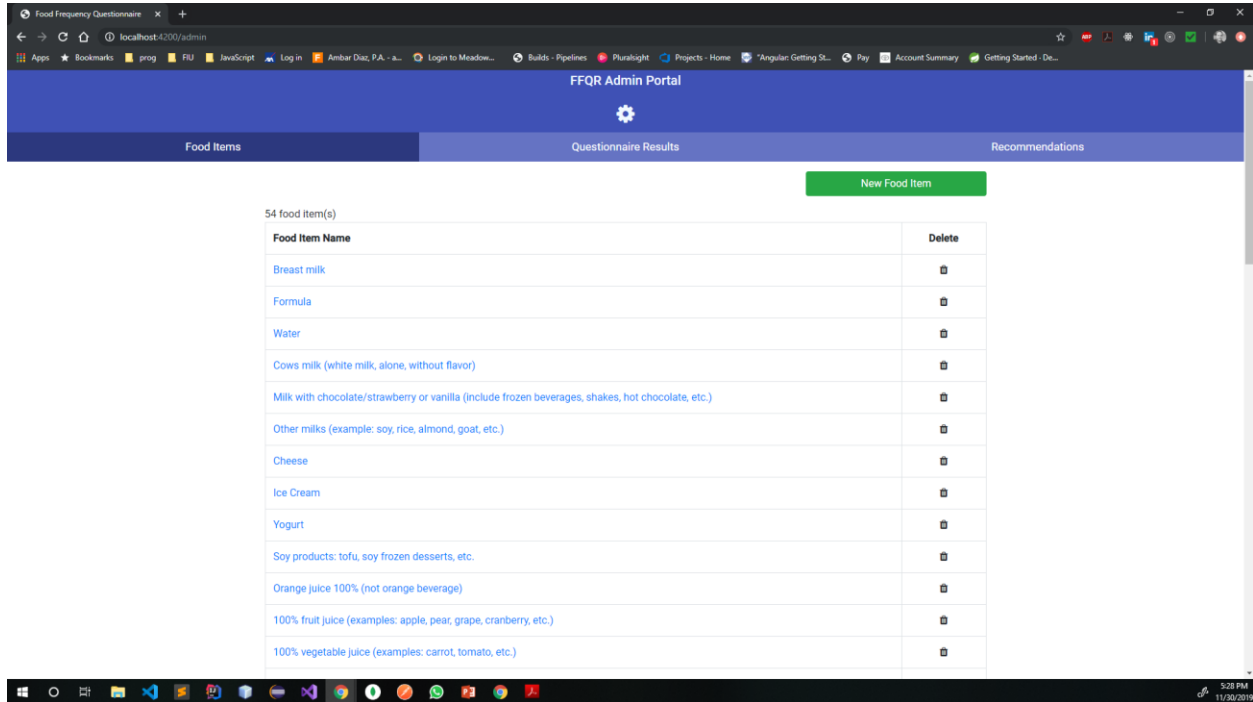


Figure 9. Admin Portal landing page

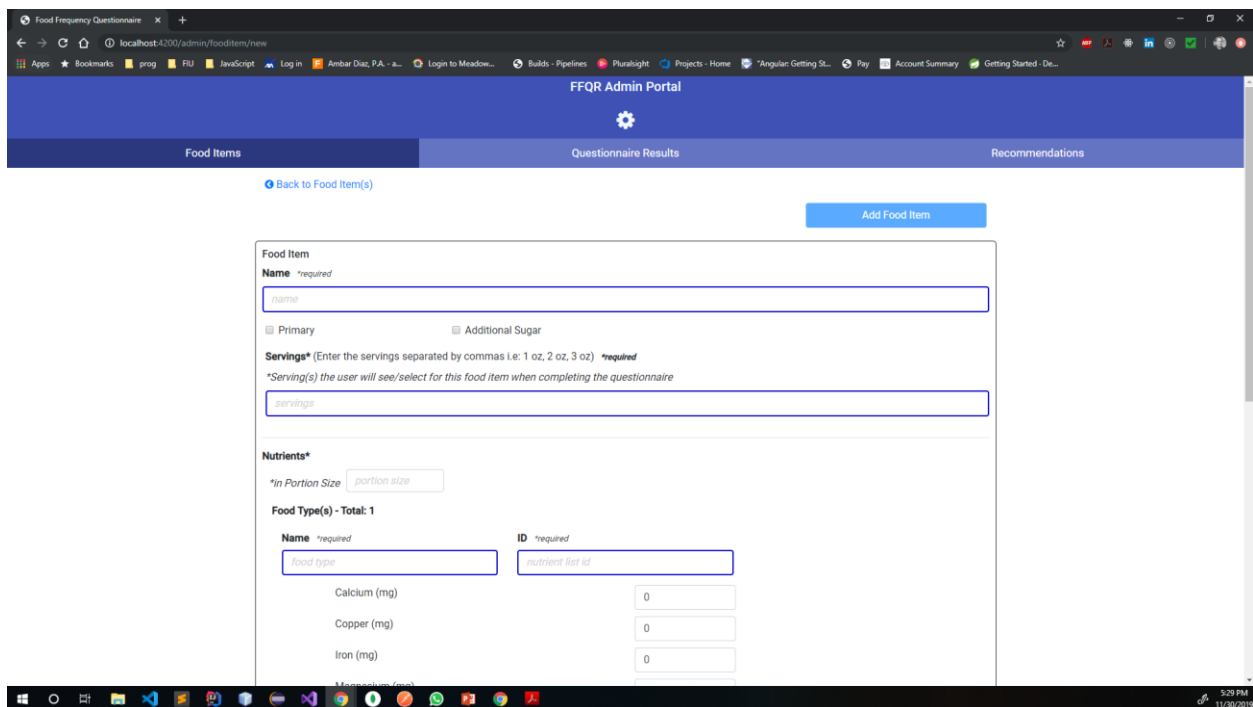
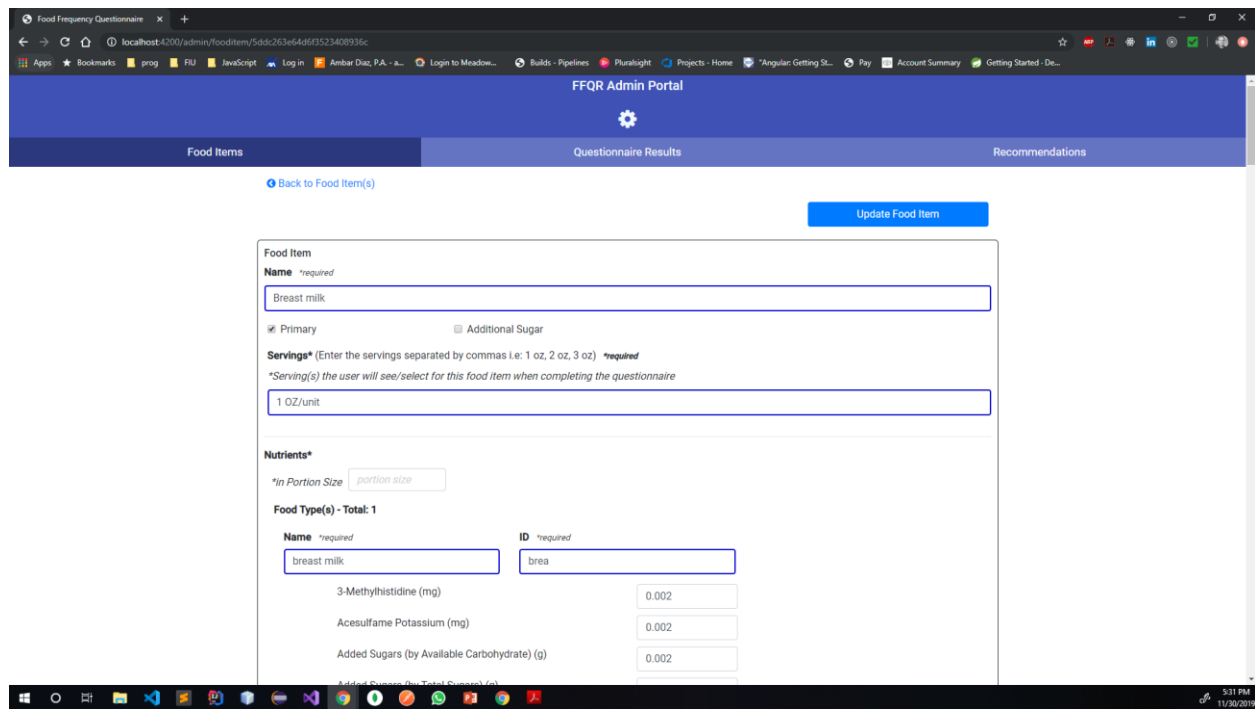


Figure 10. Add food Item page



Food Frequency Questionnaire

FFQR Admin Portal

Food Items | Questionnaire Results | Recommendations

[Back to Food Item\(s\)](#) [Update Food Item](#)

Food Item

Name *required
Breast milk

☒ Primary ☐ Additional Sugar

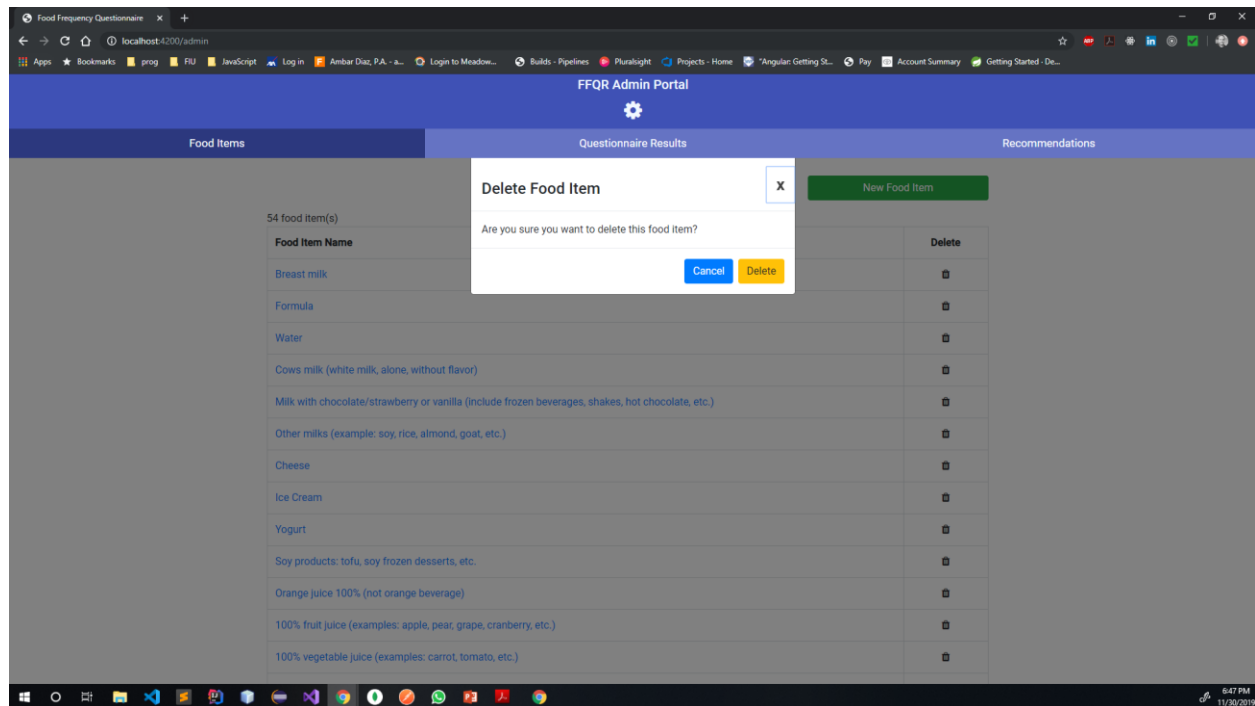
Servings* (Enter the servings separated by commas i.e: 1 oz, 2 oz, 3 oz) *required
*Serving(s) the user will see/select for this food item when completing the questionnaire
1 OZ/unit

Nutrients*
*in Portion Size

Food Type(s) - Total: 1

Name	ID
breast milk	brea
3-Methylhistidine (mg)	0.002
Acesulfame Potassium (mg)	0.002
Added Sugars (by Available Carbohydrate) (g)	0.002

Figure 11. Update Food Item page



Food Frequency Questionnaire

FFQR Admin Portal

Food Items | Questionnaire Results | Recommendations

[New Food Item](#)

Delete Food Item

Are you sure you want to delete this food item?

[Cancel](#) [Delete](#)

54 food item(s)

Food Item Name	Delete
Breast milk	Delete
Formula	Delete
Water	Delete
Cows milk (white milk, alone, without flavor)	Delete
Milk with chocolate/strawberry or vanilla (include frozen beverages, shakes, hot chocolate, etc.)	Delete
Other milks (example: soy, rice, almond, goat, etc.)	Delete
Cheese	Delete
Ice Cream	Delete
Yogurt	Delete
Soy products: tofu, soy frozen desserts, etc.	Delete
Orange juice 100% (not orange beverage)	Delete
100% fruit juice (examples: apple, pear, grape, cranberry, etc.)	Delete
100% vegetable juice (examples: carrot, tomato, etc.)	Delete

Figure 12: Delete Food Item validation

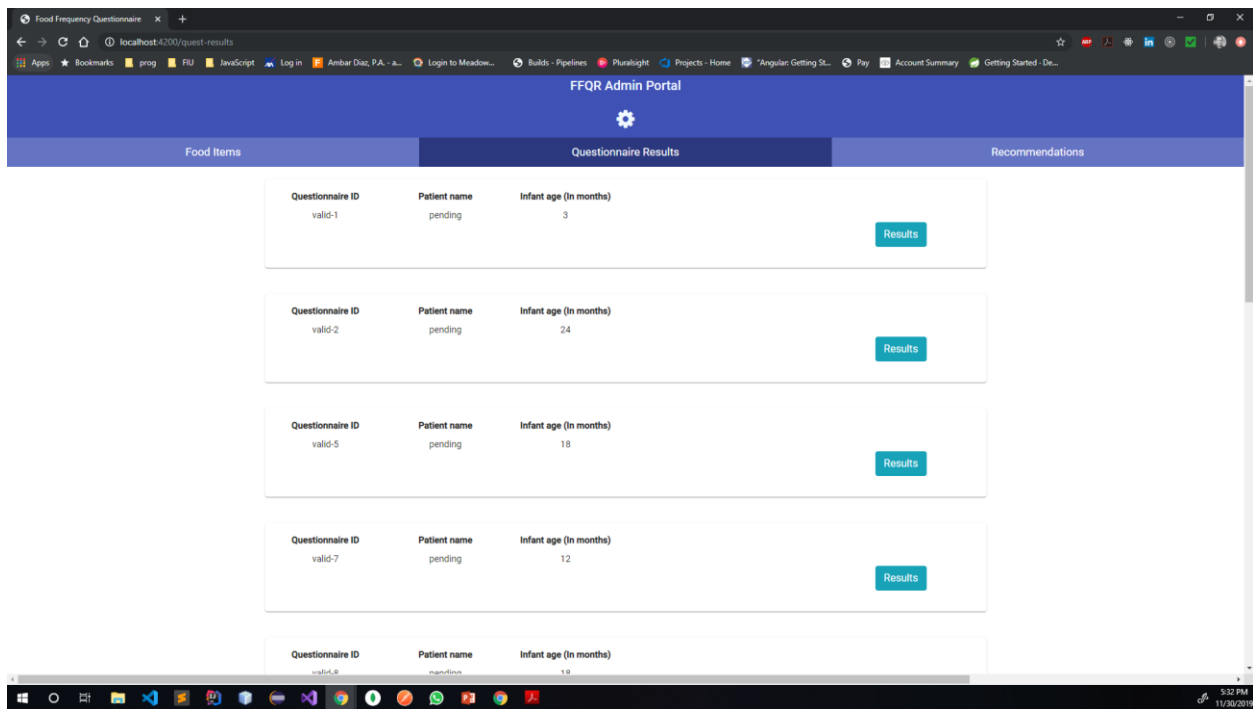


Figure 13. Questionnaire Results page

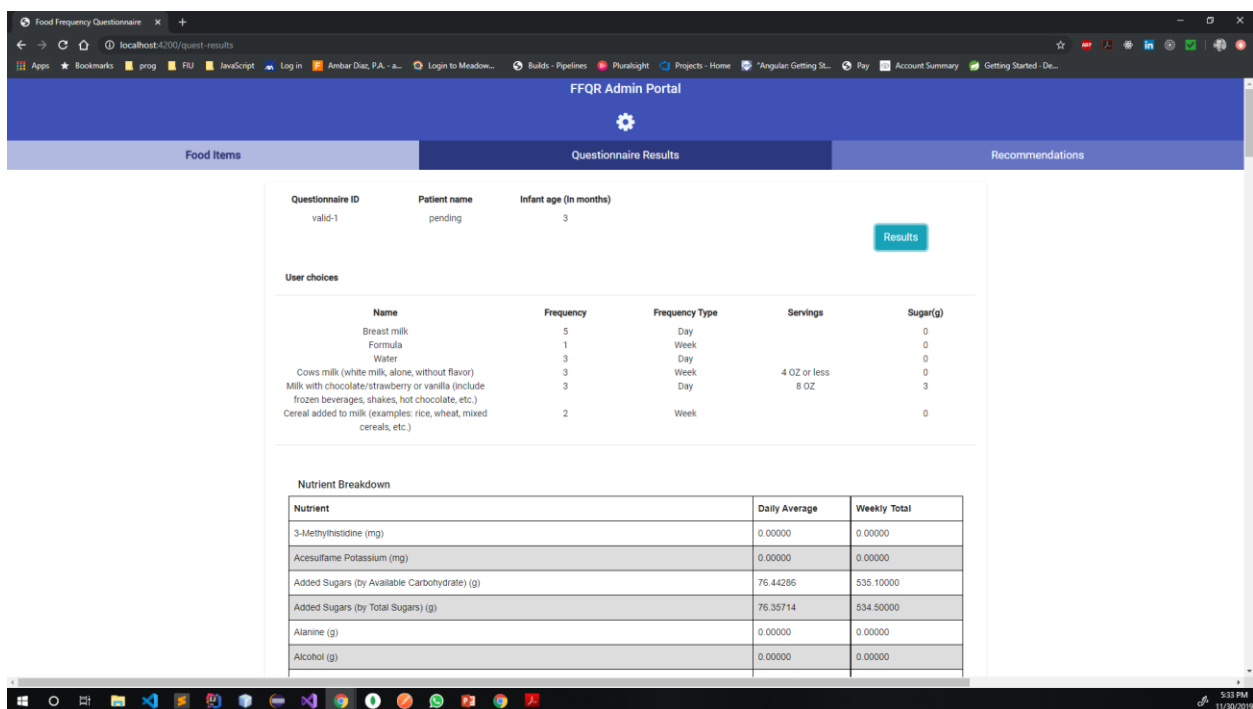


Figure 14. Questionnaire Results page (Results expanded)

The screenshot shows the FFQR Admin Portal interface. The top navigation bar includes links for Apps, Bookmarks, prog, FIU, JavaScript, Log in, Amber Diaz, P.A. - a..., Login to Meadow..., Builds - Pipelines, Pluralight, Projects - Home, Angular Getting St..., Pay, Account Summary, and Getting Started - De... The main header is 'FFQR Admin Portal' with a gear icon. Below the header, there are three tabs: 'Food Items', 'Questionnaire Results' (selected), and 'Recommendations'. The 'Questionnaire Results' tab displays a table with the following data:

Nutrient	Value 1	Value 2
Alpha-Carotene (provitamin A carotenoid) (mcg)	0.00000	0.00000
Animal Protein (g)	26.06071	182.42500
Arginine (g)	0.00000	0.00000
Ash (g)	7.60714	53.25000
Aspartame (mg)	0.00000	0.00000
Aspartic Acid (g)	2.61071	18.27500
Available Carbohydrate (g)	124.42857	871.00000
Beta-Carotene (provitamin A carotenoid) (mcg)	55.13929	385.97500
Beta-Carotene Equivalents (derived from provitamin A carotenoids) (mcg)	55.13929	385.97500
Beta-Cryptoxanthin (provitamin A carotenoid) (mcg)	0.00000	0.00000
Beta-Tocopherol (mg)	0.00000	0.00000
Betaine (mg)	5.38571	37.70000
Biochanin A (mg)	0.00000	0.00000
Caffeine (mg)	12.00000	84.00000
Calcium (mg)	922.82500	6,459.77500
Cholesterol (mg)	65.90714	461.35000
Choline (mg)	140.39286	982.75000
Copper (mg)	0.02500	0.17500
Coumestrol (mg)	0.00000	0.00000
Cystine (g)	0.00000	0.00000

Figure 15. Questionnaire Results page (Nutrient Breakdown list)

The screenshot shows the FFQR Admin Portal interface. The top navigation bar is the same as in Figure 15. The main header is 'FFQR Admin Portal' with a gear icon. Below the header, there are three tabs: 'Food Items', 'Questionnaire Results', and 'Recommendations' (selected). The 'Recommendations' tab displays a table with the following data:

Questionnaire ID	Patient name	Infant age (in months)	Nutrients	Food Items
valid-1	pending	3	⚠️	🍽️
valid-2	pending	24	⚠️	🍽️
valid-5	pending	18	⚠️	🍽️
valid-7	pending	12	⚠️	🍽️
valid-8	pending	18	⚠️	🍽️
valid-9	pending	18	⚠️	🍽️
valid-10	pending	18	⚠️	🍽️
valid-11	pending	12	⚠️	🍽️
valid-12	pending	18	⚠️	🍽️
valid-13	pending	18	⚠️	🍽️
valid-14	pending	24	⚠️	🍽️
valid-15	pending	18	⚠️	🍽️
valid-24	pending	3	⚠️	🍽️

Figure 16. Recommendations page

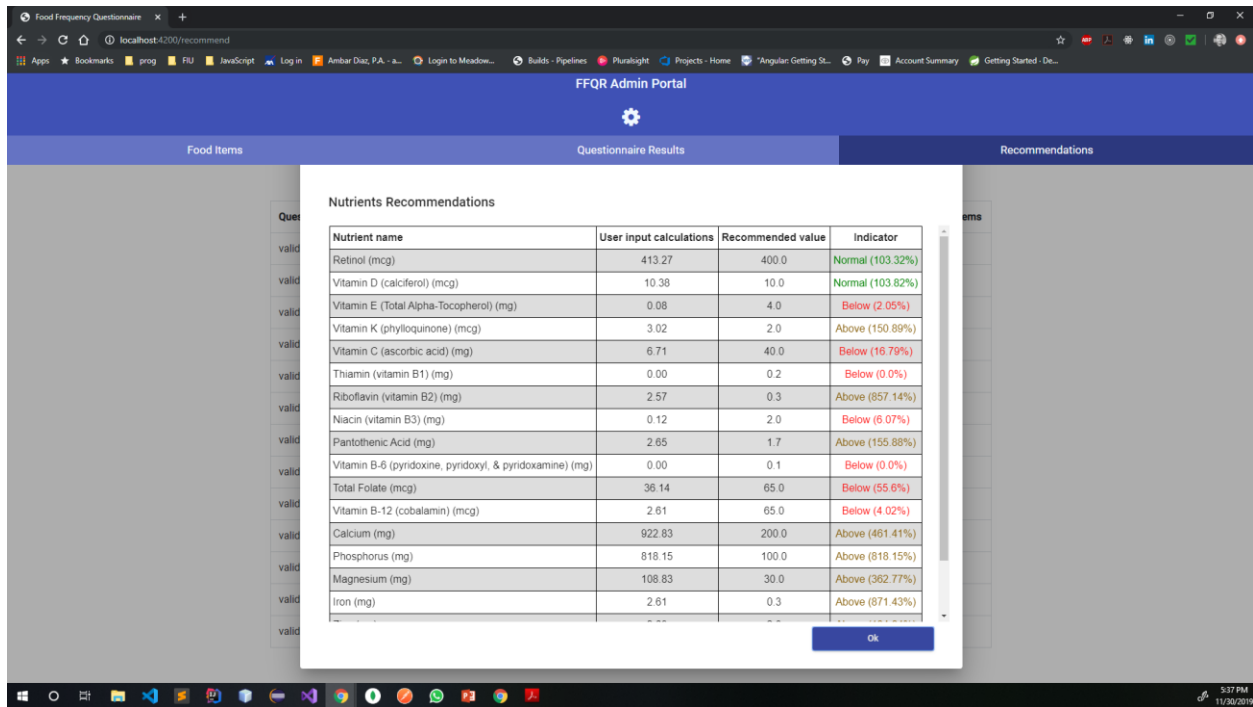


Figure 17. Nutrients recommendations modal with recommended and calculated nutrients

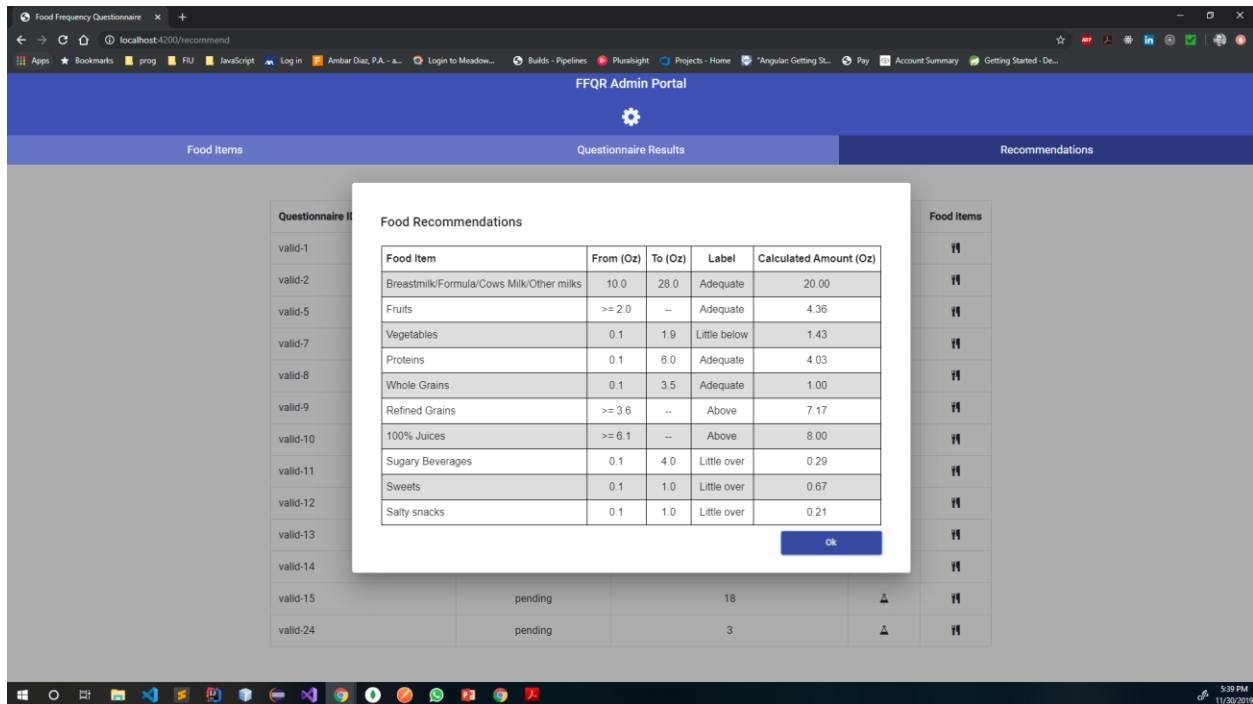


Figure 18. Food intake recommendations modal with recommended and calculated food intake

Appendix C Sprint Meeting Reports

Sprint 1

Sprint 1 Review

Date: 09/10/2019

Start time: 11:30 am

End time: 12:15 pm

Attendees: Dariana Gonzalez, Daykel Muro, Cristina Palacios

After meeting and discussion with product owner on 09/10/2019 the team agreed on the following tasks and stories for this sprint.

Sprint 1 - Week 1

TASK-01 Install and set up required frameworks and tools

TASK-02 Watch Angular, MongoDB and Sprint Boot tutorials

TASK-03 Set up and run v1.0 in local environment

TASK-04 Explore actual scope for v1.0

Sprint 1 - Week 2

FFQR-2-01 Result Class Diagram

FFQR-2-02 Save Questionnaire Results

FFQR-2-03 Retrieve all questionnaire results

FFQR-2-04 View "Success" message on questionnaire submission

FFQR-2-05 Include questionnaire ID when submit answer.

Sprint 1 Retrospective

Date: 09/23/2019

Attendees: Dariana Gonzalez - Daykel Muro

Start time: 7:15 am

End time: 7:25 am

What went right?

- All stories and tasks were completed as planned.
- There was good communication among team members throughout the sprint.

What went wrong?

- Spotty communication between product owner and team members.

Process adjustments for next Sprint?

All communication with the product owner will be done by email since she stated accessing Google Drive documents is an inconvenient.

Sprint 2 Review

Date: 10/07/2019

Attendees: Dariana Gonzalez - Daykel Muro - Cristina Palacios - Jennifer Bolton

Start time: 8:30 am

End time: 9:05 am

After a show and tell presentation, the following stories were approved by Product Owner:

Sprint 2 - Week 1

FFQR-2-06 Create Admin Page Mockup (Design)

FFQR-2-07 Add Admin Page routing (Front-End / UI)

FFQR-2-08 Implement end-point to return all food items and nutrients (Back-End)

FFQR-2-09 Food Items View (Front-End)

Sprint 2 - Week 2

FFQR-2-10 Add foods/nutrients (Front-End / UI)

FFQR-2-11 Update foods/nutrients (Front-End / UI)

FFQR-2-12 Delete foods/nutrients (Front-End / UI)

FFQR-2-13 Add foods/nutrients (Back-End / API)

FFQR-2-14 Update foods/nutrients (Back-End / API)

FFQR-2-15 Delete foods/nutrients (Back-End / API)

Sprint 2 Retrospective

Date: 10/07/2019

Attendees: Dariana Gonzalez - Daykel Muro

Start time: 7:15 am

End time: 7:25 am

What went right?

- All stories and tasks were completed.
- There was good communication among team members throughout the sprint.
- We added value to the product and team members got more familiar with the product and technology used.

What went wrong?

- Database design and implementation in version 1.0 wasn't intended to be used for this administrator portal for the food item CRUD operations. Lots of data manipulation and redesigning required in order to access and display the information properly.
- Still pending properly display of serving sizes for the food items.

Process adjustments for next Sprint?

Trying to keep up the pace in order to use extra time to get rid off any technical debt and fix/remove any bug that can be detected.

Sprint 3 Review

Date: 10/21/2019

Attendees: Dariana Gonzalez - Daykel Muro

Start time: 6:00 pm

End time: 6: 30 pm

After a show and tell presentation, the following stories were approved by Product Owner:

Sprint 3 - Week 1

FFQR-2-16 View all questionnaire results (Front-end).

FFQR-2-19 REFACTOR: Add "infant age" field in questionnaire form (Front-end).

FFQR-2-20 Create results model and service in web app (Front-end).

FFQR-2-21 View all questionnaire results (Back-end).

FFQR-2-22 REFACTOR: API to handle new field added to DB "PortionSize".

FFQR-2-23 REFACTOR: Add "infant age" field in questionnaire and food item services. (Back-end)

FFQR-2-24 Validate result view when there are no questionnaire submissions to display.(Front-end)

Sprint 3 - Week 2

FFQR-2-17 Implement in the add / update views the handling of "servings" (Front-end).

FFQR-2-18 Implement in add / update view the handling of "multiples food types" (Front-end).

FFQR-2-25 Display success message upon successful submission/update of new food item. (Front-end)

FFQR-2-26 Display total calories in questionnaire results page.(Front-end)

Sprint 3 Retrospective

Date: 10/21/2019

Attendees: Dariana Gonzalez - Daykel Muro

Start time: 5:45 pm

End time: 6:00 pm

What went right?

- Team managed to deliver all stories adding more value to the final product.
- All information is displayed properly in a fashionable way. Web-page is working for different sizes of devices (ipad, ipad pro and tablets)
- Communication between team members was exceptional. New ideas were added regarding the UI implementation and those ideas were applied to the app.
- From the architectural point of view the solutions were implemented also thinking in future improvements and add-ons to the web application.

What went wrong?

- Unused libraries that were installed and later no needed, causes conflicts in one of the team members environments, but thanks to good communication the problem was solved quickly .

Sprint 4 Review

Date: 11/05/2019

Attendees: Dariana Gonzalez - Daykel Muro - Cristina Palacios - Jennifer Bolton

Start time: 1:00 pm

End time: 1:30 pm

After a show and tell presentation, the following stories were approved by Product Owner:

Sprint 4 - Week 1

FFQR-2-27 Redesign database schema to include nutrient recommendations table - Back-End
FFQR-2-28 Implement JAVA API model for nutrients recommendations - Back-End
FFQR-2-32 Create mockup for nutrients recommendations - Design
FFQR-2-34 Implement nutrients recommendations Angular component (Front-End)
FFQR-2-35 Implement nutrients recommendations Angular HTML template (Front-End)
FFQR-2-36 Implement DB system nutrients recommendations table population on food_item_service start

Sprint 4 - Week 2

FFQR-2-29 Implement JAVA API controller to handle nutrients recommendations - (Back-End)
FFQR-2-30 Implement API service to calculate nutrients recommendations - Back-End
FFQR-2-31 Implement Angular web service to handle nutrients recommendations requests (Front-End)
FFQR-2-33 Wire the nutrients recommendations component with the nutrient recommendation service - Front-End
FFQR-2-34 Implement nutrients recommendations Angular component (Front-End)

Sprint 4 Retrospective

Date: 11/04/2019

Attendees: Dariana Gonzalez - Daykel Muro

Start time: 6:00 pm

End time: 6:20 pm

What went right?

- Team managed to deliver all stories adding more value to the final product.
- All information is displayed properly in a fashionable way. Web-page is working for different sizes of devices (ipad, ipad pro and tablets)
- Communication between team members was exceptional.
- Nutrients recommendations are being displayed according to PO info provided.

What went wrong?

- At the time of this review, PO stated that the table provided with Nutrients Recommendations was out-dated and unrevised.
- Small changes (recommended and calculation values round-up) requested after the sprint ends.
- Recommended normal value was not provided as a range of values, but as a fixed value instead.

Sprint 5 Review

Date: 11/18/2019

Attendees: Dariana Gonzalez - Daykel Muro - Cristina Palacios - Jennifer Bolton

Start time: 8:30 am

End time: 9:00 am

After a show and tell presentation, the following stories were approved by Product Owner:

Sprint 5 - Week 1

FFQR-2-38 Add DB sys_food_recommendation table population on food_item_service start
FFQR-2-39 Implement JAVA models for food recommendations - (Back-End)
FFQR-2-40 Implement JAVA service for food recommendations - (Back-End)
FFQR-2-44 Add validation on add/update food items to avoid duplicated nutrient IDs (Back-end)

FFQR-2-45 Implement ffqfood-recommendation model in Angular application (Front-End)

FFQR-2-37 Implement Food Recommendation Controller (Back-end)

Sprint 5 - Week 2

FFQR-2-41 Implement Angular web service to handle foods recommendations requests (Front-End)

FFQR-2-42 Implement modal component to view foods recommendations in Angular application (Front-End)

FFQR-2-43 Add "category" field to FoodType model and FoodItem payload (Back-end)

FFQR-2-46 Implement validation in add/update food items forms (Front-End)

FFQR-2-47 Fix CSS in questionnaire page to improve appearance (Front-end)

Sprint 5 Retrospective

Date: 11/18/2019

Attendees: Dariana Gonzalez - Daykel Muro

Start time: 8:30am

End time: 9:00am

What went right?

- Team communication was outstanding. Because of that, the team was able to deliver at the end of the sprint.
- All stories were completed. No technical debt accrued for the project.

What went wrong?

- Improper handling of food items in version 1.0 made impossible to implement properly the food recommendation for *Breastmilk/Formula/Cow's Milk/Other milks* (no servings were implemented for breast milk and formula)
- Sugar was not handled as an independent food item in the version 1.0. It was not possible to add it to the recommendations.

Sprint 6 Review

Date: 12/2/2019

Attendees: Dariana Gonzalez - Daykel Muro - Cristina Palacios - Jennifer Bolton

Start time: 8:30 am

End time: 9:00 am

After a show and tell presentation, the following stories were approved by Product Owner:

Sprint 6 - Week 1

FFQR-2-48 Refactor nutrients recommendations calculations to calculate/include percentage of recommended value (Back-end)

FFQR-2-49 Refactor nutrients recommendations view to show colors based on new indicators that include percentages

FFQR-2-50 Containerize Services, Mongo, and Web Application

FFQR-2-51 Regression Testing of v1.0 features

Sprint 6 - Week 2

FFQR-2-T01 Validate ffq/allfoodsnutrients endpoint

FFQR-2-T02 Validate ffq/createfoodnutrients endpoint

FFQR-2-T03 Validate ffq/updatefoodnutrients endpoint

FFQR-2-T04 Validate ffq/results/all endpoint

FFQR-2-T05 Validate ffq/foodrecommendations/calculate/{questionnaireID} endpoint
 FFQR-2-T06 Verification of food intake recommendations
 FFQR-2-T07 Food item delete option functionality testing
 FFQR-2-T08 Questionnaire results page shows results per questionnaires submitted
 FFQR-2-T09 Add/Update food item form validations
 FFQR-2-T10 Recommendation page shows nutrients and food intake recommendations

Note: Stories labeled FFQR-2-TXX are testing stories, T for testing and XX for consecutive number

Sprint 6 Retrospective

Date: 12/2/2019

Attendees: Dariana Gonzalez - Daykel Muro

Start time: 5:30 pm

End time: 6:00 pm

What went right?

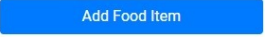
- Deployment of FFQR system in product owner local was done successfully.
- The percentages in the nutrients recommendations were added.
- All stories were completed. No technical debt accrued for the project.

What went wrong?

- Due to time constraints, the inclusion of the category field in the Add Food Item and Update Food Item could not implemented

Appendix D Documentation

Admin Web Portal User Manual

1. The Landing/Homepage is the Admin portal with three basic components inside. **Food Item** page, **Questionnaire Results** page and **Recommendations Page**.
2. When run locally, this would look like **localhost:4200/admin**
3. **Food Item** page should display a list of 54 food items and inside those food items should be a total of 127 nutrients for that food item. Inside the **Food Item** page *Admin* is able to Update The Food item , Delete (using the trash can icon in the right column ) and **Add New Food Items** .
4. **Questionnaire Results** page should have a list of questionnaires submitted, when click on **Results**  button. That questionnaire information is displayed. Admin must be able to check the calculations results, user choices and Questionnaire ID.
5. **Recommendations** page should have a list of questionnaire submitted with name, Questionnaire ID and two icons, one icon for food intake recommendations  and the other icon for nutrients recommendations . (See **Appendix B**)

Questionnaire Service Installation/Maintenance Document

- **Running the Questionnaire service as a standalone**

Your machine needs to meet the following requirements:

1. Java SDK version 1.8 or above.
2. Maven needs to be installed. Check with command: `mvn -v` Before executing the service, make sure:
 - An instance of MongoDB is running on host localhost and port 27017 on which exists a database called `ffq_database`.
 - There are no processes running on port 9080. You can do so by executing the following command.
`lsof -i:9080`
Running the service:
 1. Go to `ffquestionnaireservice` repo directory from the command line.
 2. To build and start the application, run the following command: `mvn spring-boot:run`

Program needs to be imported as an existing Maven project from IDE for continuing development.

- **Loading questionnaire database on startup (optional)**

To load sample questionnaires on startup into mongo, run the following command instead:
`mvn spring-boot:run Dmongo.questionnaires.load="true"`

Questionnaire API

The base endpoint of the API is: **localhost:9080/ffq**

Questionnaires collection name: `questionnaires`

GET calls

- Get all questionnaires: `/questionnaires`
- Get individual questionnaire by questionnaireID: `/questionnaireID/{questionnaireID}` (Currently case sensitive)

Validate questionnaireID: `/validate/{questionnaireID}`

If the exists and submitted fields are false, user can take questionnaire.

If the exists field is true, and submitted is false, user already took questionnaire.

POST calls

- Create a single questionnaire: `/create`
In the request body, pass one `FFQuestionnaire` object in a JSON payload.
Example:
`{"questionnaireID":"valid-25", "issuerID":"Daykel Muro"}`
- Create multiple questionnaires: `/createMany`

In the request body, pass an arraylist of `FFQuestionnaire` objects in a JSON payload.

Example:

```
[{"questionnaireID":"palacios33", "issuerID":"Cristina Palacios"}, {"questionnaireID":"leia003", "issuerID":"Leia Diaz"}]
```

PUT calls

- To mark a questionnaire as submitted: `/update`

In the request body, pass a `FFQuestionnaire` object with the updated info. The `questionnaireID` in this object must be valid.

Example:

```
{"questionnaireID":"palacios33", "submitted":true}
```

DELETE calls

- Delete questionnaire by questionnaireID: /delete/{questionnaireID} Pass the questionnaireID in a request parameter. Make sure the questionnaireID exists.

Food Item Service Installation/Maintenance Document

- Running the Food Item service as a standalone

Your machine needs to meet the following requirements:

1. Java SDK version 1.8 or above.
2. Maven needs to be installed. Check with command: mvn -v

Before executing the service, make sure:

1. An instance of MongoDB is running on host localhost and port 27017 on which exists a database called ffq_database.
2. There are no processes running on port 9090. You can do so by executing the following command.
lsof -i:9090

- Running the service:

1. Go to ffqfooditemservice repo directory from the command line.
 2. To build and start the application, run the following command: mvn spring-boot:run
- Program needs to be imported as an existing Maven project from IDE for continuing development.

- Loading food item database on startup (optional)

On the resources folder, there is a FoodItemPayload JSON file containing all the info for the food items in the questionnaire and a FFQRDatabase excel file with the nutrient information for each food item.

To load these food items and nutrient lists on startup into mongo collections, run the following command instead:
mvn spring-boot:run Dmongo.fooditems.load="true"

Web App Installation/Maintenance Document

- Running the Questionnaire Web App as a standalone

The following needs to be installed on your machine:

Angular CLI: 7.3.0

Node: 11.1.0

Angular:

Package	Version
@angulardevkit/architect	0.13.0
@angulardevkit/core	7.3.0
@angulardevkit/schematics	7.3.0
@schematics/angular	7.3.0
@schematics/update 0.13.0rxjs	6.3.3
typescript	3.2.2

Most if not all of these can be installed via npm .

1. Clone the ffquestionnaireweb repo
2. From the root directory, run:
 - a. ng serve

3. Application is accessible via browser at:
- localhost:4200/admin

Docker Installation/Maintenance Document

Docker Requirements

- Docker version 18.09.2
- Install Docker Desktop: <https://docs.docker.com/dockerformac/install/>

Spin Up Environment

1. Make sure Docker is running.
2. Clone the **Repo_Docker_1** repository in Azure DevOps.
3. Make sure you are logged in to the project's DockerHub account. ** user: daykelms password: Cuba12345 repository: daykelms/ffq-questionnaire **
4. Run **docker-compose build** to build the container images. This only needs to be done once as the images will be saved on your local machine.
5. Run **docker-compose up**
6. Wait for all containers to start which is signalled by:
 - o *"questionnaire-web-app | wdm: Compiled successfully."* note: **If you are using Desktop Tool Box last info you will see in the CLI is the ng build of angular**
7. Environment is up and running and can be accessible by sending Postman requests to either of the REST endpoints or going to the questionnaire web application via localhost:4200 or 192.168.99.100:4200 if using Desktop Tool Box
8. Shut down containers with **docker-compose down**

Update The Container Images Web Application Image

DockerHub allows 1 free repo. We used the web application for this one. Each repository has its own Dockerfile in the root directory.

Build the latest version of the application via ng build Then build the image via docker build -t fiuffqr/questionnaire-web-app:TAG where you substitute TAG with a tag of choice for this image. Then upload the image via docker push daykelms/questionnaire-web-app:TAG

Service Images

The ultimate goal would be to all containers in DockerHub which would require a monthly subscription of \$7. Currently, the service images are built and launched via .jar files which is why we created a repo to store them. Compile the latest version of each service of each jar and substitute it in its appropriate directory Until images are all in DockerHub, these changes should be checked in via source control The Dockerfile in the repo will build these containers from their .jars when docker-compose build is run

Once the service containers are uploaded to DockerHub, this repo is no longer necessary. The docker-compose file can be updated to pull the images from DockerHub each time. This would also allow a separate docker-compose file to be written for just the services which can be included in the web-app repository.

Questionnaire Service Wishlist

- Be able to delete a bulk of questionnaires for an issuer in one API call Food Service Wishlist.
- Refactor the serving name in the ServingOptions class to make sure that each serving name consists of a number followed by space and then by the unit. The refactoring would also make sure all the units in the serving name are valid.
- Implement a create method that takes a NutrientList object for each new Food Item. This would be used when the nutritional information for an item is not in the nutrient list collection yet.
- Change the creation of food items and nutrient lists such that they are created in the same method, so that if a nutrient list ID is removed, the payload does not have to be changed to remove the food item that was related to that nutrient list ID.

Web App Wishlist

- Add frequency count limits depending on Frequency Type and warn the user if these are exceeded, allowing them to continue if they confirm. Product Owner recommended max per day is 20 and per week is 100.
- Learning curve for Angular and CSS took a lot of time to implement features and it was an ultimate goal to complete the Admin portal and be able to display results so Product Owner can have a working prototype.
- Improve form validations for the food items, so far is only restricted to 4 required fields. No string or Integer validation is implemented.
- Recommendations should be improved to display the information in a more intuitive way
- Still missing the Profile management of the Admin portal in which admin has to login to access the questionnaire results and also is able to provide all necessary validation info to the Issuer.

REFERENCES

1. Infant and Toddler Nutrition | CDC. (n.d.). Retrieved November 13, 2019, from <https://www.cdc.gov/nutrition/infantandtoddlernutrition/index.html>
2. Poor Nutrition | CDC. (n.d.). Retrieved November 13, 2019, from <https://www.cdc.gov/chronicdisease/resources/publications/factsheets/nutrition.htm>
3. Websystiqueadmin. (2017, September 30). Spring Boot Rest API Example. Retrieved October, 25, 2019, from <http://websystique.com/springboot/springbootrestapiexample/>
4. Busy Developers' Guide to HSSF and XSSF Features. (n.d.). Retrieved September 27, 2019, from <https://poi.apache.org/components/spreadsheet/quickguide.html>
5. Docker Documentation. (2019). Retrieved from <https://docs.docker.com/>
6. Angular Docs. (2019). Retrieved from <https://angular.io/docs>