Software Requirements Specification for Utrition

Team 16, Durum Wheat Semolina
Catherine Chen, Yasmine Jolly,
Jeffrey Wang, Jack Theriault,
Alex Moica, Justina Srebrnjak

November 1, 2022

Contents

Revision History

Date	Version	Notes
November 1, 2022	1.0	Initial Version

1 Introduction

1.1 Purpose of Document

This document describes the requirements for 'Utrition'. It will provide a detailed overview of the application, and a comprehensive description of the software's functional and non-functional requirements, as well as other details relevant to the product. The template for this document is a modified version of the Volere template (Robertson and Robertson, 2012).

1.2 Scope of Requirements

When considering the files the user uploads to the system, the requirements of this document are constrained to consider only a subset of all possible file types. Furthermore, requirements are created under the impression that files are of a manageable size, and are of food objects identifiable by a machine learning model. Lastly, we assume the user possesses sufficient knowledge to access the system, and does so with internet through a machine capable of running most modern applications. That is to say, the user possesses sufficient hardware capabilities to access all capabilities of the system.

1.3 Characteristics of Intended Reader

This document is intended for the developers and testers working on this system, so that the requirements of the system can be communicated throughout the development process. Readers should have an understanding of the software design life cycle as well as the ability to read discrete math at a university-level in order to fully understand the contents and equations used in this document.

1.4 Organization of Document

The rest of the SRS will discuss the purpose and context of the system, followed by the functional requirements it must satisfy. Then the document will mention its non-functional requirements, providing justifications for each. This is followed by a discussion of potential issues that may arise during the project's development. Lastly, the document lists likely and unlikely changes, and includes a traceability matrix for the system.

2 Project Drivers

2.1 The Purpose of the Project

The purpose of the Utrition application is to help individuals learn more about maintaining a healthy lifestyle and assisting them in making wise decisions. This application is built for the users to learn more about what their meals consist of by breaking down the calories and nutrition facts of the food they intend on eating or purchasing.

2.2 The Stakeholders

2.2.1 The Client

The client requesting this application is the instructor of the Software Engineering 4G06, Dr. Smith and the teaching assistants. This application is expected to be unique as compared to products in the market that attempt to tackle a similar problem as per the client's request.

2.2.2 The Customers

The customers of Utrition are assumed to be health-conscious individuals that are attempting to improve their daily eating habits by tracking the nutrients and calories found in their previous meals.

2.2.3 Other Stakeholders

Other stakeholders of the Utrition application consist of grocery stores, restaurants, and gyms. These stakeholders are affected as the users of this program may change their previous dynamic with these stakeholders due to their increase in knowledge of the nutrients they are consuming. In terms of a grocery store, their stock may be affected due to an increase in individuals buying more health-conscious foods. For restaurants, they might experience a decrease in customers if users develop a preference to cook from home in order to eat healthier. Lastly, gyms might see an increase in attendees if Utrition users wish to burn off extra calories they discover they have consumed.

2.3 Mandated Constraints

As we wish to make Utrition as accurate as possible when identifying food, the application will only be trained for individual food items or ingredients rather than whole meals. For example, the user is expected to provide a picture of lettuce rather than an entire sandwich if they wish to track that lettuce appeared in their sandwich. This constraint is to ensure that the user gets an accurate estimate of exactly the nutritional value of their meal since the application may not detect all separate food items in one picture.

2.4 Naming Conventions and Terminology

symbol	description
A	Assumption
API	Application programming interface
App	Application
CP	Cultural and Political Requirement
FR	Functional Requirement
LC	Likely Change
LF	Look and Feel Requirement
LR	Legal Requirement
MS	Maintainability and Support Requirement
OE	Operational and Environmental Requirement
PR	Performance Requirement
SR	Security Requirement
SRS	Software Requirements Specification
UH	Usability and Humanity Requirement
UI	User Interface
ULC	Unlikely Change

2.5 Relevant Facts and Assumptions

A1: The user has an understanding of how to effectively utilize web based applications.

A2: The user is able to upload a food image to Utrition. This can be a saved image or a new image taken using a device.

A3: There is an assumption that the client will only take a picture of individual ingredients.

A4: The user is expected to have knowledge of the metric system as Utrition will be outputting nutritional value data in grams and calories.

A5. The user is expected to have internet access while using Utrition.

3 Functional Requirements

3.1 The Scope of the Work and the Product

The scope of the project is to develop a nutrition tracking application to identify food items based on an image and derive nutritional data from the classified food. This application can

be installed on the user's system and will have comprehensive software documentation.

3.1.1 The Context of the Work

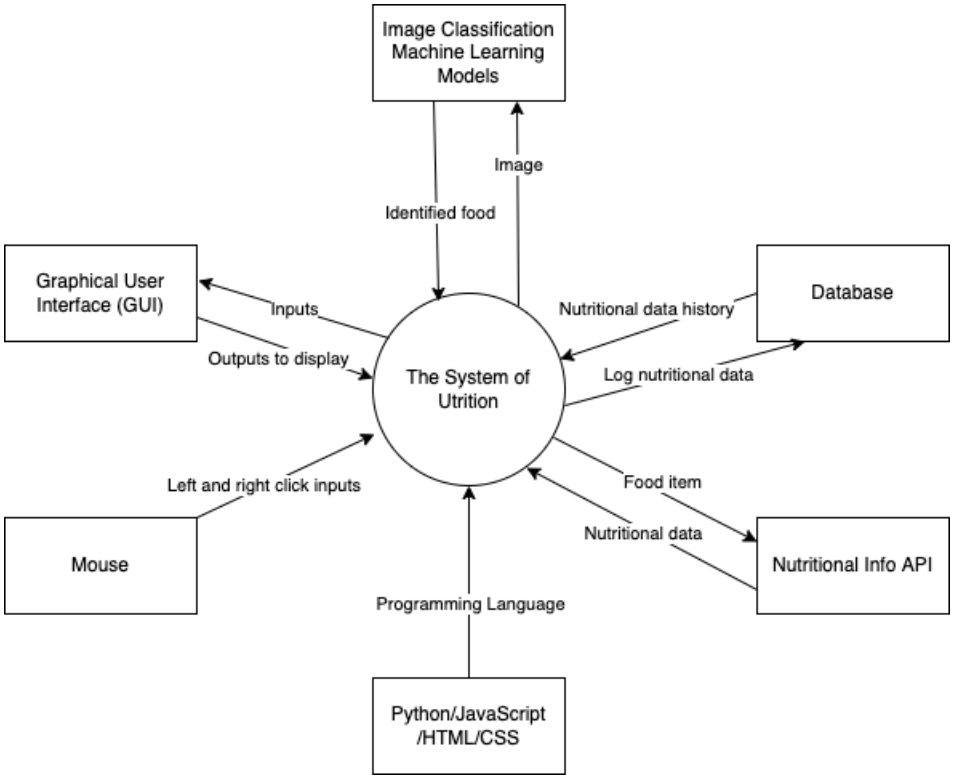

Figure 1: Work Context Diagram

3.1.2 Work Partitioning

Event Name	**Input/Output**	**Summary**
System receives image	User uploaded image (in) / Pre-processed image (out)	User uploads an image and it is pre-processed into a format that can be sent to the computer vision algorithm.
Food identified in image	Pre-processed image (in) / Identified food (out)	The pre-processed image is processed to identify the food it represents through the use of our machine learning computer vision algorithm.
Nutritional data retrieved	Identified food(in) / Nutritional data (out)	The identified food is passed into a matching algorithm to retrieve the associated nutritional data from all stored nutritional data.
Nutritional data displayed	Nutritional data (in) / Visual display of nutritional data (out)	The raw nutritional data is transformed into a human readable format and displayed to the user.
Nutritional data logged	Nutritional data (in) / Logging of data (out)	The nutritional data and search information of a user's search is internally logged and persists after the program has exited.
Past nutritional data retrieved	User mouse input (in) / Nutritional data (out)	The user is able to retrieve the nutritional data and search information of their past image uploads.
Past nutritional data chart displayed	Nutritional data (in) / Chart (out)	Past nutritional data is displayed in a graph, showing past searches and nutritional data through various periods of time.

3.1.3 Individual Product Use Cases

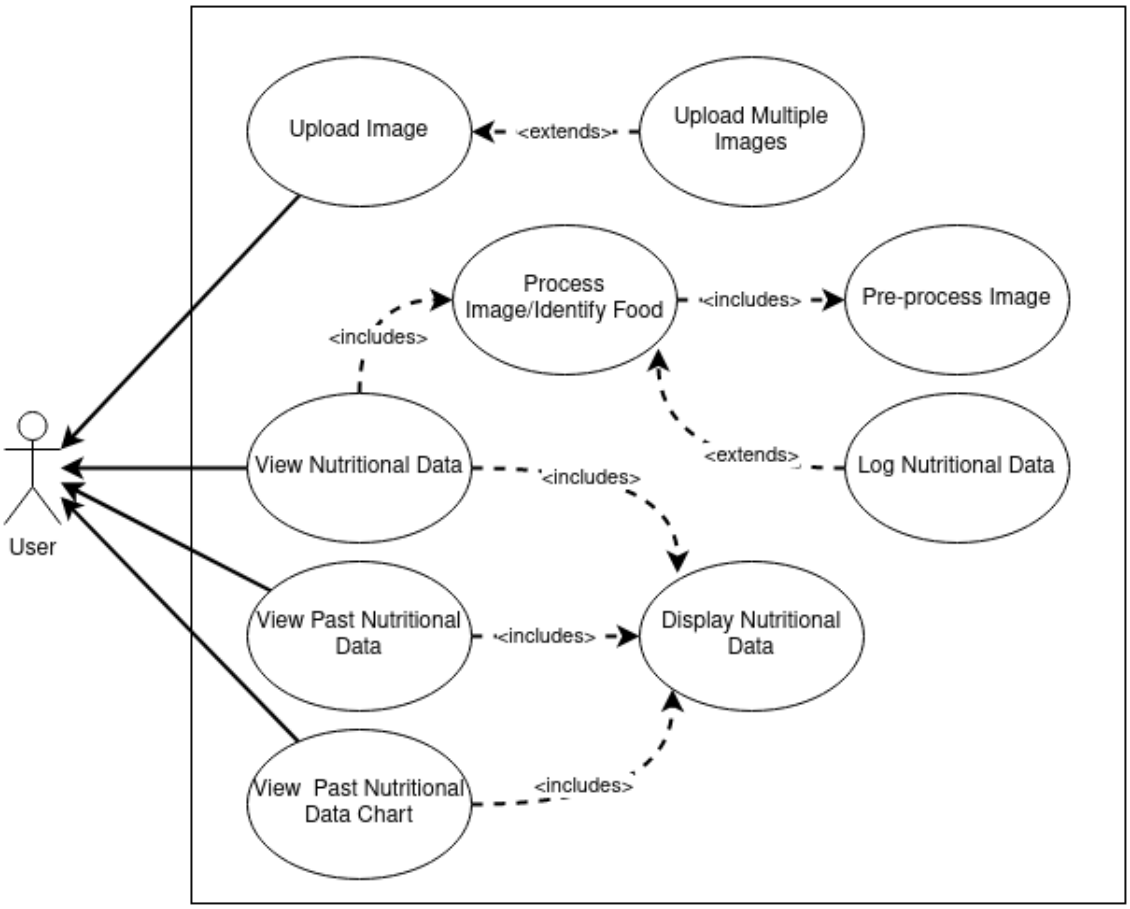

Figure 2: Use Case Diagram

3.2 Functional Requirements

FR1. The user must have the ability to upload a digital image of a **standard image type** to the system.
Formalization: $x \in A \mid x = \text{Images}, A = \{\textbf{standard image types}\}$
Rationale: In order for the system to identify the food a user is eating, the user would have to upload an image of their food to the system. The image shall be saved on the user's device, which can then be selected.
Priority: High

FR2. The user shall have the ability to upload multiple digital images of **standard image types** to the system.
Formalization: $\forall x \cdot x \in A \mid x = \text{Images}, A = \{\textbf{standard image types}\}$
Rationale: In order for the system to return the total nutritional content of a users

meal, the user would have to upload multiple images of the individual components of their meal for the system to identify. The images shall be saved on the user's device, which can then be selected.
Priority: Medium

FR3. The system will be able to identify the type of food that is captured in an image.
Formalization: $\forall x \cdot \exists y \cdot y \in B \mid x = \text{Images}, B = \{\text{food label type}\}$
Rationale: In order for the system to return the nutritional content of a food, the system must first identify the food present in an image.
Priority: High

FR4. The system will be able to make an API call to an external database of nutrition facts for a variety of foods, including their macro-nutrients, micro-nutrients, and caloric details.
Rationale: In order to provide a user with the nutritional regarding their meal, the system must first be able to fetch the nutritional data for a large variety of foods for later use.
Priority: High

FR5. The system will be able to retrieve the nutrition facts for a specific food.
Formalization: $\forall y \cdot \exists z \cdot z \in C \mid y = \text{food label}, C = \{\text{nutritional data}\}$
Rationale: In order to provide the user with the nutritional data regarding their meal, the system would have to be able to retrieve the nutritional data for a given food.
Priority: High

FR6. The system will log the nutritional data of a food.
Formalization: $\forall v \cdot \exists z \cdot z \in C \mid v = \text{logged food label}, C = \{\text{nutritional data}\}$
Rationale: Nutritional data of a food will be saved for future reference. This system shall be able to fetch previously recorded data for other processes, such as displaying visuals and calculating nutritional summaries.
Priority: Medium

FR7. The user will be able to access the nutritional data of previously logged foods.
Rationale: Nutritional data found in the database shall be available to be referenced and displayed to the user. This will allow the user to view nutritional details of the food that was previously tracked within the application.
Priority: Medium

FR8. The system will display the nutritional information of a food to the user.
Rationale: Nutritional data of a particular food item shall be displayed on the user interface, so the user can view the nutritional breakdown of the food.
Priority: High

FR9. The system will display the history of logged nutritional data in a graph.
Rationale: Previously tracked nutritional statistics will be rendered in a graph as a function over time. This display will be a visual indicator of trends and overall progress.
Priority: Medium

4 Non-Functional Requirements

4.1 Look and Feel Requirements

4.1.1 Appearance Requirements

LF1. Utrition's user interface will be clutter-free.
Rationale: Users must be able to access and read all parts of the user interface.
Fit Criteria: The distance between user interface components must exceed 20 pixels.

4.1.2 Style Requirements

N/A

4.2 Usability and Humanity Requirements

4.2.1 Ease of Use Requirements

UH1. The user interface will be easy to navigate.
Rationale: This allows the user to efficiently learn and use Utrition.
Fit Criteria: The user will be able to navigate to the main menu from any page in 2 clicks.

4.2.2 Personalization and Internationalization Requirements

UH2. Utrition will retain the user's diet history.
Rationale: Users will be able to view their dietary habits and can make nutritional adjustments based on their observations.
Fit Criteria: The user must be able to view every past inputted food item.

4.2.3 Learning Requirements

UH3. Utrition will be able to be used by people aged 14 and higher who have received no training on the product.
Rationale: Utrition should be able to be used by all individuals who are concerned with their dietary habits.
Fit Criteria: 90% of a test panel of the target demographic should be able to upload their food and retrieve the corresponding nutritional facts in under 2 minutes.

4.2.4 Understandability and Politeness Requirements

UH4. Key information will be easy to understand on Utrition.
Rationale: Users will be more likely to use the web application if they are able to understand it efficiently.
Fit Criteria: Every clickable object will have a symbol associated with it. Calories, fat, sodium, carbohydrates, sugar, and protein will have a symbol associated next to it.

UH5. Utrition will not overwhelm the user with information.
Rationale: The user is only concerned with the nutritional information and not the calculations leading up to it.
Fit Criteria: All backend calculations, such as identifying the uploaded food and retrieving the nutritional information, will not be displayed to the user.

4.2.5 Accessibility Requirements

UH6. Utrition will be able to be used by deaf people.
Rationale: Durum Wheat Semolina aims to allow as many individuals as possible to use Utrition.
Fit Criteria: There will be no sound used by the Utrition web application.

4.3 Performance Requirements

4.3.1 Speed and Latency Requirements

PR1. Switching between different user interface screens will be quick.
Rationale: The user should not have to wait to access different parts of the Utrition web application.
Fit Criteria: The time between the user's click and displaying the new UI will not exceed 2 seconds.

PR2. Identifying and displaying each food item will be done quickly.
Rationale: No process should force users to wait extended periods of time.
Fit Criteria: After uploading a picture, Utrition will always display the identified food item within 10 seconds in a test with multiple images.

PR3. Displaying a food's nutritional value will be done quickly.
Rationale: No process should force users to wait extended periods of time.
Fit Criteria: After identifying the correct food item, Utrition will always display the nutritional facts within 5 seconds in a test with multiple images.

4.3.2 Safety-Critical Requirements

PR4. Utrition will return an error message when the user uploads an abnormal image format that is not .png, .jpg, or .jpeg.

Rationale: Utrition should not crash by improper user input. Users should have an opportunity to upload a new file of an appropriate format.
Fit Criteria: Error message should prompt user upon upload of an abnormal image.

PR5. Utrition will return an error message when the user uploads an image file that exceeds the maximum size.
Rationale: Utrition should not crash by improper user input. Users should have an opportunity to upload a new file of an appropriate size.
Fit Criteria: Error message should prompt user upon upload of an image later than 30MB.

PR6. Utrition will return an error message when the user uploads more than three images at once.
Rationale: Utrition should not crash by improper user input. Users should have an opportunity to upload three or fewer images to the system.
Fit Criteria: Error message should prompt user upon upload of more than 3 images.

PR7. Utrition will prompt the user if food identification cannot be completed successfully. The user will be notified on the type of error that occurs.
Rationale: Food identification may fail due to a variety of reasons, and the user should be notified so they may attempt to find a workaround for the issue.
Fit Criteria: Error message should prompt user upon failure to identify any food items in the uploaded image

PR8. Utrition will return an error message if the request to retrieve nutritional information cannot be completed successfully.
Rationale: Information retrieval requests may fail due to a variety of reasons, and the user should be notified of the reason why the service could not be completed as expected.
Fit Criteria: Error message should prompt user upon failure to request nutritional info regarding food items in the uploaded image.

PR9. Utrition will return an error message if the nutritional information of a specific item cannot be found.
Rationale: The user should be notified if the nutritional data of their food item cannot be fetched.
Fit Criteria: Error message should prompt user upon failure to find nutritional info regarding food items in the uploaded image.

PR10. Utrition will return an error message if the user's part nutritional logs cannot be found.
Rationale: The user should be notified if their nutritional data of past meals cannot be found.
Fit Criteria: Error message should prompt user upon failure to access user's past nutritional info.

PR11. Utrition will return an error message if the user's part nutritional logs have been deleted.
Rationale: The user should be notified if their nutritional data of past meals are no longer saved in the system.
Fit Criteria: Error message should prompt user if user's past nutritional info is absent.

PR12. Utrition will prompt the user if past nutritional trends cannot be displayed successfully. The user will be notified on the type of error that occurs.
Rationale: Failure to display past nutritional trends may fail due to a variety of reasons. The user should be made aware of the issue, and the underlying cause behind it.
Fit Criteria: Error message should prompt user if user's past nutritional info cannot be depicted on a chart.

PR13. Utrition will prompt the user if their device is not connected to the internet when attempting to access the system.
Rationale: The user should be notified if they are unable to connect to the system so they may apply a fix to the issue.
Fit Criteria: Request will time out after 20 seconds, and user will be prompted that their device is not connected to the internet.

4.3.3 Precision or Accuracy Requirements

PR14. Identifying and displaying each food item will be done correctly.
Rationale: The web app should provide the user with accurate information.
Fit Criteria: In a test with multiple images, Utrition will identify and display the correct food with an average accuracy of 70%.

PR15. The displayed nutritional facts will be correct.
Rationale: The nutritional information given to the user should be accurate.
Fit Criteria: In a test with multiple images, Utrition will display the correct nutritional information for an identified food 90% of the time.

4.3.4 Reliability and Availibility Requirements

PR16. Utrition will be available to users at all times.
Rationale: The user should be able to track their diet whenever they decide to eat.
Fit Criteria: Since Utrition is a local web app, the user will always be allowed to use Utrition unless their device or internet connection is down.

4.3.5 Robustness or Fault-Tolerance Requirements

N/A

4.3.6 Capacity Requirements

PR17. Utrition will allow 3 images to be uploaded as a group.
Rationale: The user will have to wait too long for the calculated results if more than 3 images are uploaded at once.
Fit Criteria: The system will not allow more than 3 photos to be uploaded at once.

4.3.7 Scalability or Extensibility Requirements

PR18. Utrition will have no limit to the size of the user base for its lifespan.
Rationale: Utrition should be available to all individuals who are health conscious.
Fit Criteria: Utrition is a local web app and does not depend on a server.

4.3.8 Longevity Requirements

PR19. Utrition should operate indefinitely once fully developed.
Rationale: Nutritional food values will never change and people will always want ways to regulate their diet.
Fit Criteria: Utrition will be available to use on GitHub for an indefinite amount of time.

4.4 Operational and Environmental Requirements

4.4.1 Expected Physical Environment

N/A

4.4.2 Requirements for Interfacing with Adjacent systems

N/A

4.4.3 Productization Requirements

OE1. Utrition will be available on GitHub as a public repository.
Rationale: Utrition is free to be used by anyone.
Fit Criteria: Users will be able to find and pull the repository on to their personal device.

OE2. Utrition will be able to be installed by any individual aged 14 or older after reading the installation guide on the Utrition GitHub page.
Rationale: Utrition should have a simple installation process.
Fit Criteria: 95% of a test panel with the intended demographic will be able to install Utrition after reading the installation guide.

4.4.4 Release Requirements

OE3. Utrition will not have any updates after final implementation is complete.
Rationale: Food components and their nutritional value do not change over time which means no updates are needed to keep the system functioning and relevant.
Fit Criteria: Utrition's GitHub repository will receive no updates after final implementation.

4.5 Maintainability and Support Requirements

4.5.1 Maintenance Requirements

MS1. During development, new data can be used to improve Utrition's machine learning model.
Rationale: Utrition's machine learning model can receive new data to improve itself and detect food at 70% accuracy.
Fit Criteria: Any backend developer is able to successfully upload new data to Utrition's machine learning model at any time during development.

4.5.2 Supportability Requirements

MS2. Technical support related to installation and the use of Utrition will be found in Utrition's GitHub repository.
Rationale: Users who are not technologically advanced should be able to use Utrition given an installation guide and user manual.
Fit Criteria: All users will be able to view the support manuals on how to install and use Utrition.

4.5.3 Adaptability Requirements

MS3. Utrition will be used on any computer with access to ".png", ".jpg", or ".jpeg" photo format.
Rationale: The user must be able to upload image files to Utrition in one of these formats.
Fit Criteria: 100% of the tested Windows, macOS, and Linux devices should be able to use all functionality provided for Utrition.

4.6 Security Requirements

4.6.1 Access Requirements

SR1. The entirety of Utrition can be accessed by all individuals.
Rationale: The user has the flexibility to improve Utrition and the right to know what they're downloading onto their computer.
Fit Criteria: Anyone will be able to pull the open-source project from GitHub

4.6.2 Integrity Requirements

SR2. Utrition will periodically save user's data during use.
Rationale: In the event of unexpected shutdown, the user should not lose all information from the last session. Periodically saving user information will allow users to continue from their last step in the event of an unexpected shutdown.
Fit Criteria: Utrition will save user data ever 3 minutes throughout the application runtime.

4.6.3 Privacy Requirements

SR3. Utrition will store the user's data locally.
Rationale: Only the user should have to view their stored dietary habits.
Fit Criteria: Utrition will not contain personalized data from one user on another user's system.

4.6.4 Audit Requirements

N/A

4.6.5 Immunity Requirements

N/A

4.7 Cultural or Political Requirements

4.7.1 Cultural Requirements

CP1. Utrition will not display any culturally insensitive content.
Rationale: Utrition should not show culturally insensitive content since the application is intended for individuals of any race or religion.
Fit Criteria: 95% of the test panel agrees that Utrition does not provide any culturally insensitive content to the user.

4.7.2 Political Requirements

N/A

4.8 Legal Requirements

4.8.1 Compliance Requirements

LR1. Utrition follows Canada's Anti-Spam Legislation Requirements for Installing Computer Programs (CRTC, 2020).
Rationale: Durum Wheat Semolina complies with the law to provide an application

void of spam and computer viruses.
Fit Criteria: No malicious software will be present in Utrition's GitHub repository.

4.8.2 Standards Requirements

LR2. Durum Wheat Semolina follows Canada's "Guide for Publishing Open Source Code" (Government of Canada, 2020).
Rationale: This guide will assist Durum Wheat Semolina in setting up the development environment for Utrition.
Fit Criteria: Durum Wheat Semolina will not commit infractions when following Canada's guide.

LR3. Durum Wheat Semolina will adhere to the PEP8 Python coding conventions.
Rationale: This guide ensures code is consistent which allows the development team to work more efficiently.
Fit Criteria: Pylint will be used throughout development to ensure every pull request contains properly formatted code.

LR4. Utrition is created following the standards set by the Google HTML/CSS Style Guide (Google, a) and Google Java Style Guide (Google, b).
Rationale: This guide ensures code is consistent which allows the development team to work more efficiently.
Fit Criteria: Before every pull request, code will be reviewed to ensure no violations of the standards occur.

LR5. Utrition's front end will follow Google's guide for material design (Google, c)
Rationale: The appearance of Utrition will be professional and consistent when following this guide.
Fit Criteria: Utrition's user interface will be reviewed to ensure all fonts and colours adhere to the style guide.

5 Project Issues

5.1 Open Issues

Below are open issues concerning the development of Utrition:

- Depending on the amount of identified food image data our team can obtain, the accuracy of our food detection machine learning model may be low
- Currently, Utrition is planned to saved past user input locally. It is unknown how this will be done. One method such as employing an account and password system for each user is added functionality that may be difficult to implement

- One functional requirement states that past nutritional values of food will be stored for easy access when the system needs it again. When saving large amounts of nutritional data, processing could become slow for the entire system.

5.2 Off-the-Shelf Solutions

5.2.1 Ready-Made Products

After investigating the market, there are two ready-made solutions that provide similar functionality to Utrition. Firstly, the application Calorie Mama is able to identify calories when a picture is uploaded of the user's meal. Although this implements some of Utrition's functionality, it fails to identify nutritional information or save any past calorie calculations.

Another product the team found to be similar to Utrition is Bitesnap. This application will be able to identify food items, provide their corresponding nutritional information and calories, and store this information to provide eating insights for the user. Frequent reviews state that the system is not able to identify many foods provided in an image by the user. This means that the application's food database is small. Utrition will focus on creating a machine learning model that is able to identify a vast number of food items by supplying a large dataset of labelled food images.

5.2.2 Reusable Components

Through research into libraries and APIs, the team has identified two systems that can be used for development of Utrition. Firstly, our team plans to create a machine learning model to identify different food items from an uploaded image. To do this, the TensorFlow library can be used. This free and open-source library will be used to create and employ our machine learning model.

Secondly, Utrition must be able to return nutritional information about a food when an image of that food is uploaded. In order to retrieve this nutritional information, calls to Nutritionix API can be made. Nutritionix contains data on a vast range of foods and their nutritional value. Using the free version, our team can make calls to this API to retrieve needed nutritional information.

5.2.3 Products That Can Be Copied

The team will be using datasets of labelled food images in order to train our machine learning model. These datasets can be freely obtained online and used in our development.

5.3 New Problems

N/A

5.4 Tasks

5.4.1 Project Planning

Team members will follow a strict schedule to ensure the completion of Utrition. The team will work in 1 week Sprint cycles where tasks are distributed during the first meeting of the week (Monday at 6:30pm) and all tasks must be finished before that meeting the week after. There is also a scheduled progress meeting half-way through the Sprint (Thursday at 1:30pm) to give an update to the team on each person's progress and discuss any blocking issues. The team will create tasks each week based on the due dates provided in SFWRENG 4G06. This ensures that all deadlines are met.

5.5 Migration to the New Product

N/A

5.6 Risks

5.6.1 Low Productivity

Working on a team with multiple developers creates the possibility that some people will not be productive and complete their assigned tasks. Missed work will cause a higher risk for an incomplete or inadequate final implementation. This risk can be identified when a developer is frequently missing deadlines and delivering poorly done work. To address this risk, the offending team member will be addressed directly by the group and asked about their opinion on possible solutions. This can include working on different areas of the project, asking for help when needed, etc.
Probability: %15

5.6.2 Inadequate Implementation

Due to the volume of requirements, the difficulty of the implementation, and the timeline of the project, there is a risk that the development team will not be able to create a good quality product. This includes partial implementation of features or hardcoded workarounds. To identify this risk before it becomes an issue, the development team hosts weekly progress meetings to ensure all members are proceeding smoothly. In the event that this risk is unavoidable, the team will alter the final goals of Utrition to remove unnecessary functionality.
Probability: %10

5.6.3 Limited Time Frame

Since Utrition's final implementation and documentation are required in less than 7 months, there is a risk that the project does not get completed. Every team member has additional work responsibilities outside of this project which could affect the time they have to work on Utrition. One sign that this risk poses a threat to the project is missed deadlines. This sign

can be identified easily during the team's weekly meetings. If this risk becomes a problem, the team will apply an all-hands-on-deck approach where all unfinished work will be split up equally.
Probability: %8

5.7 Costs

Utrition will be development using free programming technologies and open-source libraries. This sets the current budget of the project to $0. The plan for Utrition's availability is for users to download all code files from GitHub in order to run the system. In future development, hosting Utrition as a website will cost between the range of $3.95 - $16.95 per month.

The time cost for the project is calculated based on the number of developers and the duration of the project. Utrition will be development by 6 individuals who all can provide 10 hours every week. The duration of the project is set to 8 months. Using these values, the total amount of time needed to complete the project is 1920 hours and 320 hours per developer.

5.8 User Documentation and Training

5.8.1 User Documentation Requirements

Utrition will be available to users from GitHub. Inside the repository, the README file will contain instructions for the download and setup of Utrition. Any required software needed by the user to run Utrition will be specified in this file along with solutions to common installation errors. This file will also discuss all functionality of the system and provide instructions for the user to use Utrition. This section will act as the installation guide and user manual.

5.8.2 Training Requirements

There is no training needed for users to use Utrition. Any questions regarding usage of the system will be available in the user manual.

5.9 Waiting Room

In this section, requirements for future releases of Utrition are stated from highest to lowest priority. For a more in-depth explanation of each requirement, please refer to section 3 (Stretch Goals) inside the Problem Statement and Goals document.

FR10. The system must be accessible online as a website.
Rationale: Hosting Utrition online will make it easier for users to access the system. They can search for Utrition in an online search instead of downloading the project from GitHub.

FR11. The system must be able to identify the individual foods inside a picture of a meal.
Rationale: If the system can identify visually apparent ingredients inside of a meal, the user will not need to upload pictures of each individual food item which will increase efficiency.

FR12. Users must be able to save preset meal options consisting of one or many foods.
Rationale: Users can save selected foods together as meals so they can easily log this meal as eaten without needing to manually upload images of individual ingredients.

FR13. Utrition will provide insights into the eating habits of the user such as what vitamins they should try to consume more of based on their previous inputs.
Rationale: By providing health insights, the system will become more useful to users by helping them improve their eating habits rather than solely tracking them.

PR10. Once hosted online, Utrition must compute output with similar performance to the downloaded system within a 25% margin of error.
Rationale: Hosting Utrition online should provide similar performance speeds compared to the local version so users experience the same enjoyment levels when using the system.

5.10 Ideas for Solutions

Below are some ideas for implementation of Utrition:

- Create frontend display using HTML/CSS and JavaScript to connect to backend
- Backend can be implemented with Python
- Use TensorFlow library to create a machine learning model and train it with datasets of labelled food
- Use Nutritionix API calls to retrieve nutrition facts of food items

6 Likely Changes

This section has been added to the Volere template and details the likely changes to the system that can be made later on in development.

LC1: With respect to A1, the system currently assumes the uploaded pictures will show individual ingredients. In the future, a more sophisticated computer vision model can be added to differentiate individual ingredients from a meal.

LC2: The system currently supports a model where the user uploads images of food to be evaluated. In the future, the system may allow the user to record their meals in real time and identify nutritional facts instantaneously.

LC3: The system is currently designed to run locally on the user's device. In the future, the system can be hosted online, and allow users to request nutritional data from system servers.

7 Unlikely Changes

This section has been added to the Volere template and details the unlikely changes to the system during development.

ULC1: The user's past nutritional history will always be stored locally on the user's device rather than on an online server. This will limit the number of requests in the event the system is hosted online. Furthermore, this allows the user to access their nutritional history offline.

8 Traceability Matrix

This section has been added to the Volere template. The traceability matrix below illustrates the relation between functional and non-functional requirements.

	FR1	FR2	FR3	FR4	FR5	FR6	FR7	FR8	FR9
LF1		x					x	x	x
UH1		x						x	x
UH2			x				x		x
UH3	x	x					x		
UH4	x	x					x	x	x
UH5	x	x					x	x	x
UH6	x	x						x	x
PR1							x	x	x
PR2							x	x	x
PR3					x			x	
PR4			x	x	x		x	x	
PR5									
PR6	x	x	x	x	x		x	x	x
PR7		x							
PR8									
PR9									
OE1									
OE2									
OE3									
MS1			x		x			x	
MS2									
MS3									
SR1									
SR2	x	x							
SR3						x			
CP1									
LR1									
LR2									
LR3									
LR4									
LR5		x					x	x	x

Table 1: Traceability Matrix Showing the Connections Between Functional and Non-functional

9 Appendix

This section has been added to the Volere template.

- **Standard image types**: These are image files of extensions which include .png, .jpg, .jpeg.

9.1 Constants

N/A

10 Reflection Appendix

10.1 Skills

10.1.1 Competence in specialized engineering knowledge

This skill describes one's knowledge on specialized engineering subjects in order to contribute to implementation, documentation, and the final presentation. Some examples of this knowledge include understanding each documents' purposes, the different software architectures to be used in implementation, and proper version control practices. This is highly important for the success of Utrition since all aspects of the project are technically advanced and could not be accomplished by individuals uneducated in software engineering. Ways for people to improve this skill include consulting notes from past courses taken, reviewing previous projects, and researching needed unknown areas of expertise.

10.1.2 Successfully uses engineering tools

In order to create a successful project, it is necessary to be able to successfully utilize engineering tools. There are various types of engineering tools that all accomplish different tasks. For example, GitHub ensures that a large group of developers are all able to work on one project at the same time which promotes efficiency. Another example is LaTeXwhich formats every document professionally. Kanban boards ensure that everything gets done by the deadlines and lastly, coding linters keep project code formatted consistently. Approaches to learning these skills include reading documentation, asking more knowledgeable individuals for advice, and reading online discussion boards that answer questions on how to utilize the tool.

10.1.3 Effective project management

Effective Project Management is a skill for planning and effectively managing a project's time, resources, and scope. Also, to master the skill, one must be able to follow business practices when appropriate. In relation to Utrition, these business practices are using git branches and pull requests, showing up to the biweekly meetings, working together on Microsoft Teams, and using git issues to assign tasks. Durum Wheat Semolina decided on these business practices, as they are needed to be able to allocate resources in an effective manner to ensure that the project tasks are efficiently completed. To exercise this skill, one can use GitHub to distribute tasks and assign individuals to them, use the project Kanban board frequently, and participate in all group meetings.

10.1.4 Develops models/prototypes; tests, evaluates, & iterates as appropriate

This skill describes the ability to not only make changes to a system through successive iterations, but also completing analysis and evaluations of the system at each iteration. These practices help detect faults during the development process, before they can manifest as bugs, and helps ensure changes to the system adhere to the requirements. For Utrition, this is significant as it will be built out iteratively. Repeated, small changes will be made and include a set of tests to ensure that our project is meeting requirements throughout the development process. This skill can be exercised by creating a process for automated unit testing, as well as encouraging all team members to thoroughly test new code changes. Furthermore, the team can complete root cause analysis when fixing bugs in order to better track potentially error-prone code.

10.1.5 Evaluates engineering tools, identifies their limitations, and selects, adapts, or extends them appropriately

This skill describes the ability to choose the most effective software tool for an application from a variety of options based off of intended use, tool limitations, software requirements, ease of use, and a variety of other factors. Choosing the most relevant tools for our project is the best way to ensure that our project is not inherently limited by its framework and that it is ultimately successful. Knowing tool limitations will also allow us to adapt our product to either work around them or guide our development to incorporate them in our design process. The skill can be improved through reading tool documentation, researching tool reviews, comparisons, and benchmarks, and performing our own tests and sample code executions to evaluate their performance.

10.1.6 Active contribution to the planning and execution of a team project

This skill describes the ability to contribute to the team project as an active member of the group. In order for all teammates to be more aligned with project goals and a standard of high-quality work, the group as a whole should be involved in the planning and execution of the project and be aligned on common goals. This requires the members to be invested in

the planning and execution of the various components of the project. Having team members actively contribute to the project will foster a supportive and productive team dynamic. This skill can be improved through engaging in team meetings, sharing opinions or suggestions with the team, and helping with GitHub project board organization.

10.2 Teammate Specific Learning Approaches

10.2.1 Alexander Moica

Alexander has had repeated previous experience working with a specific set of software tools but has rarely branched out to investigate and evaluate potential alternatives. In order to develop the skills necessary for the efficient evaluation of competing software tools (Section 10.1.5), Alexander will search for tool reviews, comparisons, and benchmarks. This method was chosen since this content is available in a variety of easily digestible formats - including visual representations, video reviews, and interactive comparison tables.

10.2.2 Yasmine Jolly

Yasmine recognizes that she often forgets to work iteratively and test/evaluate her work as she progresses (Section 10.1.4). She often saves this step to the end of her work and spends large amounts of time searching for mistakes. During this project, she will work to improve this skill by frequently running tests on her code to make sure there are no new regressions introduced. This method was chosen because it allows Yasmine to frequently think about edge cases in her code and create tests that will find such exceptions.

10.2.3 Jeffrey Wang

Jeffrey is familiar with the tools and programming languages frequently used in the academic setting, but needs to improve his self-directed learning ability. During the development of Utrition, there will be opportunities to handle new technology outside the scope of an academic setting that will require self-learning to understand (Section 10.1.1). Jeffrey will research unknown areas of expertise to quickly develop an understanding of the unknown area to a level adequate for this project. Furthermore, he will read the documentation of new technologies to better understand its capabilities. These methods were chosen as they best tackle learning about subjects outside the scope of an academic setting.

10.2.4 Jack Theriault

Jack has acknowledged that in the past, he has found it difficult to plan a project, taking into consideration its time, resources, and scope (Section 10.1.3). Often times, time is overestimated, resulting in some elements of the project remaining unfinished. In order for him to improve his project management skills, Jack will check the project Kanban board regularly so he knows what tasks need to be done and by what date.

10.2.5 Catherine Chen

Catherine expressed that she wanted to be more proactive in the planning and execution of the project (Section 10.1.6). She has previously struggled with being aware of what other members are working on. In order to develop her initiative and leadership skills, Catherine will lead project planning meetings. This includes spearheading team meetings for planning project deadlines, being aware of the allocation of tasks within the group, actively using GitHub issues, and project board organization. This method was chosen as it allows her to take more of a leadership role within the team where she can plan the allocation of tasks to team members, and she will gain a more holistic view on the project.

10.2.6 Justina Srebrnjak

Justina identifies that she has difficulty with understanding certain engineering tools such as LaTeX, make, and certain programming languages such as JavaScript (Section 10.1.2). In order to better understand these tools and use them successfully, Justina will read any documentation available. She has chosen this method since she possesses a visual learning style and documentation is easily accessible online.

References

CRTC. Canada's anti-spam legislation requirements for installing computer programs, 2020. URL https://crtc.gc.ca/eng/internet/install.htm.

Google. Google html/css style guide, a. URL https://google.github.io/styleguide/htmlcssguide.html.

Google. Google javascript style guide, b. URL https://google.github.io/styleguide/jsguide.html.

Google. Material design guide, c. URL https://material.io/design.

Government of Canada. Guide for publishing open source code, 2020. URL https://www.canada.ca/en/government/system/digital-government/digital-government-innovations/open-source-software/guide-for-publishing-open-source-code.html.

James Robertson and Suzanne Robertson. *Volere Requirements Specification Template*. Atlantic Systems Guild Limited, 16 edition, 2012.