

Module Interface Specification for Utrition

Team 16, Durum Wheat Semolina

Alexander Moica

Yasmine Jolly

Jeffrey Wang

Jack Theriault

Catherine Chen

Justina Srebrnjak

March 7, 2023

1 Revision History

Date	Version	Notes
January 18, 2023	1.0	Initial Document

2 Symbols, Abbreviations and Acronyms

See SRS Documentation, Semolina (2022b), at <https://github.com/jeff-rey-wang/utrition/blob/3c91ed8d891c50d14bab9dd2f7ddcd5d3d465f56/docs/SRS/SRS.pdf>

Contents

1	Revision History	i
2	Symbols, Abbreviations and Acronyms	ii
3	Introduction	1
4	Notation	1
5	Module Decomposition	1
6	Application Path Module	3
6.1	Module	3
6.2	Uses	3
6.3	Syntax	3
6.3.1	Exported Constants	3
6.3.2	Exported Access Programs	3
6.4	Semantics	3
6.4.1	State Variables	3
6.4.2	Environment Variables	3
6.4.3	Assumptions	3
6.4.4	Access Routine Semantics	3
6.4.5	Local Functions	3
7	MIS of Home Page Module	4
7.1	Module	4
7.2	Uses	4
7.3	Syntax	4
7.3.1	Exported Types	4
7.3.2	Exported Access Programs	4
7.4	Semantics	4
7.4.1	State Variables	4
7.4.2	Environment Variables	4
7.4.3	Assumptions	4
7.4.4	Access Routine Semantics	4
7.4.5	Local Functions	4
8	MIS of Profile Page Module	5
8.1	Module	5
8.2	Uses	5
8.3	Syntax	5
8.3.1	Exported Types	5
8.3.2	Exported Access Programs	5

8.4	Semantics	5
8.4.1	State Variables	5
8.4.2	Environment Variables	5
8.4.3	Assumptions	5
8.4.4	Access Routine Semantics	5
8.4.5	Local Functions	5
9	MIS of Nutrition Log Module	6
9.1	Module	6
9.2	Uses	6
9.3	Syntax	6
9.3.1	Exported Constants	6
9.3.2	Exported Access Programs	6
9.4	Semantics	6
9.4.1	State Variables	6
9.4.2	Environment Variables	6
9.4.3	Assumptions	6
9.4.4	Access Routine Semantics	6
9.4.5	Local Functions	7
10	MIS of Upload Page Module	8
10.1	Module	8
10.2	Uses	8
10.3	Syntax	8
10.3.1	Exported Types	8
10.3.2	Exported Access Programs	8
10.4	Semantics	8
10.4.1	State Variables	8
10.4.2	Environment Variables	8
10.4.3	Assumptions	8
10.4.4	Access Routine Semantics	8
10.4.5	Local Functions	9
11	MIS of Upload Container Module	10
11.1	Template Module	10
11.2	Uses	10
11.3	Syntax	10
11.3.1	Exported Types	10
11.3.2	Exported Access Programs	10
11.4	Semantics	10
11.4.1	State Variables	10
11.4.2	Environment Variables	10
11.4.3	Assumptions	10

11.4.4	Access Routine Semantics	10
11.4.5	Local Functions	10
12	Image Upload Module	11
12.1	Module	11
12.2	Uses	11
12.3	Syntax	11
12.3.1	Exported Types	11
12.3.2	Exported Access Programs	11
12.3.3	State Variables	11
12.3.4	Environment Variables	11
12.3.5	Assumptions	11
12.3.6	Access Routine Semantics	11
12.3.7	Local Functions	12
13	Text Upload Module	13
13.1	Module	13
13.2	Uses	13
13.3	Syntax	13
13.3.1	Exported Types	13
13.3.2	Exported Access Programs	13
13.4	Semantics	13
13.4.1	State Variables	13
13.4.2	Environment Variables	13
13.4.3	Assumptions	13
13.4.4	Access Routine Semantics	13
13.4.5	Local Functions	14
14	Voice Upload Module	15
14.1	Module	15
14.2	Uses	15
14.3	Syntax	15
14.3.1	Exported Types	15
14.3.2	Exported Access Programs	15
14.4	Semantics	15
14.4.1	State Variables	15
14.4.2	Environment Variables	15
14.4.3	Assumptions	15
14.4.4	Access Routine Semantics	15
14.4.5	Local Functions	16

15 Navigation Bar Module	17
15.1 Module	17
15.2 Uses	17
15.3 Syntax	17
15.3.1 Exported Types	17
15.3.2 Exported Access Programs	17
15.4 Semantics	17
15.4.1 State Variables	17
15.4.2 Environment Variables	17
15.4.3 Assumptions	17
15.4.4 Access Routine Semantics	17
15.4.5 Local Functions	18
16 MIS of Input Pre-Processing Module	19
16.1 Module	19
16.2 Uses	19
16.3 Syntax	19
16.3.1 Exported Constants	19
16.3.2 Exported Access Programs	19
16.4 Semantics	19
16.4.1 State Variables	19
16.4.2 Environment Variables	19
16.4.3 Assumptions	19
16.4.4 Access Routine Semantics	19
16.4.5 Local Functions	20
17 MIS of Training Dataset Module	21
17.1 Module	21
17.2 Uses	21
17.3 Syntax	21
17.3.1 Exported Constants	21
17.3.2 Exported Access Programs	21
17.4 Semantics	21
17.4.1 State Variables	21
17.4.2 Environment Variables	21
17.4.3 Assumptions	21
17.4.4 Access Routine Semantics	21
17.4.5 Local Functions	22
18 MIS of Image Classification Module	23
18.1 Module	23
18.2 Uses	23
18.3 Syntax	23

18.3.1	Exported Constants	23
18.3.2	Exported Access Programs	23
18.4	Semantics	23
18.4.1	State Variables	23
18.4.2	Environment Variables	23
18.4.3	Assumptions	23
18.4.4	Access Routine Semantics	23
18.4.5	Local Functions	24
19	MIS of Nutritional Data Retriever Module	25
19.1	Module	25
19.2	Uses	25
19.3	Syntax	25
19.3.1	Exported Constants	25
19.3.2	Exported Access Programs	25
19.4	Semantics	25
19.4.1	State Variables	25
19.4.2	Environment Variables	25
19.4.3	Assumptions	25
19.4.4	Access Routine Semantics	26
19.4.5	Local Functions	26
20	MIS of Profile Data Calculation Module	27
20.1	Module	27
20.2	Uses	27
20.3	Syntax	27
20.3.1	Exported Constants	27
20.3.2	Exported Access Programs	27
20.4	Semantics	27
20.4.1	State Variables	27
20.4.2	Environment Variables	27
20.4.3	Assumptions	27
20.4.4	Access Routine Semantics	28
20.4.5	Local Functions	28
21	Appendix	30

3 Introduction

The following document details the Module Interface Specifications for Utrition. Utrition is an application that will provide the nutritional facts for an inputted food item. Users can provide input through text, voice, or image. Utrition will also log past input food data for users to easily view their eating habits and nutritional intake.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at <https://github.com/jeff-rey-wang/utrition>.

4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol $:=$ is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | \dots | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by Utrition.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	\mathbb{Z}	a number without a fractional component in $(-\infty, \infty)$
natural number	\mathbb{N}	a number without a fractional component in $[1, \infty)$
real	\mathbb{R}	any number in $(-\infty, \infty)$

The specification of Utrition uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. Utrition also uses user frontend events to signal some function executions. The type JSON is heavily used to transport data to be displayed in the application interface. In addition, Utrition uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document, Semolina (2022a), for this project.

Level 1	Level 2
Hardware-Hiding Module	N/A
Behaviour-Hiding Module	Application Path Module Home Page Module Profile Page Module Nutrition Log Module Upload Page Module Upload Container Module Image Upload Module Text Upload Module Voice Upload Module Navigation Bar Module
Software Decision Module	Input Pre-Processing Module Training Dataset Module Image Classification Module Nutritional Data Retriever Module Profile Data Calculation Module

Table 1: Module Hierarchy

6 Application Path Module

6.1 Module

App

6.2 Uses

NavBar, Home, Upload, Profile

6.3 Syntax

6.3.1 Exported Constants

None

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
App	-	App	-

6.4 Semantics

6.4.1 State Variables

None

6.4.2 Environment Variables

path: String

6.4.3 Assumptions

Users will not try to purposefully edit the site path to a nonexistent page.

6.4.4 Access Routine Semantics

App():

- transition: $path := \text{"/"}$
- output: $out := self$
- exception: None

6.4.5 Local Functions

None

7 MIS of Home Page Module

7.1 Module

Home

7.2 Uses

N/A

7.3 Syntax

7.3.1 Exported Types

Home = ?

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
Home	-	Home	-

7.4 Semantics

7.4.1 State Variables

None

7.4.2 Environment Variables

None

7.4.3 Assumptions

None

7.4.4 Access Routine Semantics

Home():

- transition: Page rendered with general information about Utrition
- output: $out := self$
- exception: None

7.4.5 Local Functions

None

8 MIS of Profile Page Module

8.1 Module

Profile

8.2 Uses

NutritionLog

8.3 Syntax

8.3.1 Exported Types

Profile = ?

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
Profile	-	Profile	-

8.4 Semantics

8.4.1 State Variables

None

8.4.2 Environment Variables

None

8.4.3 Assumptions

None

8.4.4 Access Routine Semantics

Profile():

- transition: Page rendered with information from NutritionLog
- output: $out := self$
- exception: None

8.4.5 Local Functions

None

9 MIS of Nutrition Log Module

9.1 Module

NutritionLog

9.2 Uses

UserLogData

9.3 Syntax

9.3.1 Exported Constants

None

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
dispData	-	Display component	-

9.4 Semantics

9.4.1 State Variables

data: UserLogData type

9.4.2 Environment Variables

None

9.4.3 Assumptions

None

9.4.4 Access Routine Semantics

dispData():

- transition: $data := \text{getData}()$
- output: $out := (data \neq \text{NULL}) \Rightarrow \text{Display component with user data} \mid (data == \text{NULL}) \Rightarrow \text{Display component displaying String stating "No user data available"}$
- exception: None

9.4.5 Local Functions

None

10 MIS of Upload Page Module

10.1 Module

Upload

10.2 Uses

UploadContainer

10.3 Syntax

10.3.1 Exported Types

Upload = ?

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
Upload	-	Upload	-

10.4 Semantics

10.4.1 State Variables

None

10.4.2 Environment Variables

None

10.4.3 Assumptions

None

10.4.4 Access Routine Semantics

Upload():

- transition: Page rendered with components from ImageUpload, TextUpload, and VoiceUpload
- output: $out := self$
- exception: None

10.4.5 Local Functions

None

11 MIS of Upload Container Module

11.1 Template Module

UploadContainer

11.2 Uses

ImageUpload, TextUpload, VoiceUpload

11.3 Syntax

11.3.1 Exported Types

UploadContainer = ?

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
UploadContainer	\mathbb{Z}	UploadContainer	-

11.4 Semantics

11.4.1 State Variables

None

11.4.2 Environment Variables

None

11.4.3 Assumptions

None

11.4.4 Access Routine Semantics

None

11.4.5 Local Functions

None

12 Image Upload Module

12.1 Module

ImageUpload

12.2 Uses

None

12.3 Syntax

12.3.1 Exported Types

ImageUpload

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
ImageUpload	-	ImageUpload	-
handleImage	Event	-	-
getData	-	-	BadResponseError

12.3.3 State Variables

image: String

responseData: JSON

12.3.4 Environment Variables

None

12.3.5 Assumptions

The input file is of an appropriate type and not empty. The backend of Utrition will always send a response.

12.3.6 Access Routine Semantics

ImageUpload():

- transition: *image, responseData* := "", ""
- output: *out* := *self*
- exception: None

handleImage(e):

- transition: *image* := path of uploaded image via *setImage(e)*
- exception: None

getData():

- transition: send image path, then listen for a response from backend
setResponseData(response)
- exception: (*responseData* == error) \Rightarrow BadResponseError

12.3.7 Local Functions

setImage(s)

- transition: *image* := *s*
- exception: None

setResponseData(r)

- transition: *responseData* := *r*
- exception: None

13 Text Upload Module

13.1 Module

TextUpload

13.2 Uses

None

13.3 Syntax

13.3.1 Exported Types

TextUpload = ?

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
TextUpload	-	TextUpload	-
handleFoodItem	Event	-	-
getData	-	-	BadResponseError

13.4 Semantics

13.4.1 State Variables

foodDesc : (String, \mathbb{Z})

responseData: JSON

13.4.2 Environment Variables

None

13.4.3 Assumptions

None

13.4.4 Access Routine Semantics

TextUpload():

- transition: *foodDesc*, *responseData* := "", ""
- output: *out* := *self*
- exception: None

handleFoodItem(e)

- transition: *foodDesc* := the contents of the text fields via *setFoodDesc*(e)
- exception: None

getData():

- transition: send food item, then listen for a response from backend
 setResponseData(response)
 display nutritional output
- exception: (*responseData* == error) \Rightarrow BadResponseError

13.4.5 Local Functions

setFoodDesc((foodName, servings))

- transition: *foodDesc* := (foodName, servings)
- exception: None

setResponseData(r)

- transition: *responseData* := *r*
- exception: None

14 Voice Upload Module

14.1 Module

VoiceUpload

14.2 Uses

None

14.3 Syntax

14.3.1 Exported Types

VoiceUpload = ?

14.3.2 Exported Access Programs

Name	In	Out	Exceptions
VoiceUpload	-	VoiceUpload	-
handleVoiceInput	Event	-	-
getData	-	-	BadResponseError

14.4 Semantics

14.4.1 State Variables

detectSpeech: String

responseData: JSON

14.4.2 Environment Variables

None

14.4.3 Assumptions

None

14.4.4 Access Routine Semantics

VoiceUpload():

- transition: *detectSpeech*, *responseData* := "", ""
- output: *out* := *self*
- exception: None

handleVoiceInput(e)

- transition: *detectSpeech* := the detected speech input via setDetectSpeech(e)
- exception: None

getData():

- transition: send voice input, then listen for a response from backend
setResponseData(response)
display nutritional output
- exception: (*responseData* == error) \Rightarrow BadResponseError

14.4.5 Local Functions

setDetectSpeech(s)

- transition: *detectSpeech* := s
- exception: None

setResponseData(r)

- transition: *responseData* := r
- exception: None

15 Navigation Bar Module

15.1 Module

NavBar

15.2 Uses

N/A

15.3 Syntax

15.3.1 Exported Types

NavBar = ?

15.3.2 Exported Access Programs

Name	In	Out	Exceptions
NavBar	-	NavBar	-
changePage	Event		-

15.4 Semantics

15.4.1 State Variables

None

15.4.2 Environment Variables

None

15.4.3 Assumptions

Users will not try to purposefully change the paths for each button.

15.4.4 Access Routine Semantics

NavBar():

- output: $out := self$
- exception: None

changePage():

- transition: $path := "/" \vee "/profile" \vee "/upload"$
- exception: None

15.4.5 Local Functions

None

16 MIS of Input Pre-Processing Module

16.1 Module

InputPreProcess

16.2 Uses

ImageClassification

16.3 Syntax

16.3.1 Exported Constants

None

16.3.2 Exported Access Programs

Name	In	Out	Exceptions
open	String	String	-

16.4 Semantics

16.4.1 State Variables

filePath: String

foodIdentified: String

16.4.2 Environment Variables

None

16.4.3 Assumptions

It is assumed that there exists a valid image file at the provided image file path.

16.4.4 Access Routine Semantics

open(path):

- transition: $filePath := path$
- output: $out := foodIdentified$
- exception: None

16.4.5 Local Functions

None

17 MIS of Training Dataset Module

17.1 Module

TrainingDataset

17.2 Uses

N/A

17.3 Syntax

17.3.1 Exported Constants

None

17.3.2 Exported Access Programs

Name	In	Out	Exceptions
loadData	seq of (seq of \mathbb{Z}), \mathbb{Z}	Dictionary	-

17.4 Semantics

17.4.1 State Variables

imageArray: seq of (seq of \mathbb{Z})
flag: \mathbb{Z}

17.4.2 Environment Variables

None

17.4.3 Assumptions

It is assumed the file path and file type of the CIFAR-100 datasets are respectively constant and standard.

17.4.4 Access Routine Semantics

loadData(array, f):

- transition: *imageArray*, *flag* := *array*, *f*
- output: *out* := Dictionary consisting of image labels and classes used in the machine learning model
- exception: None

17.4.5 Local Functions

- `unpickle(file)`: takes in a file path and opens it into bytestream. Specific dictionary entries are retrieved and returned depending on the filepath that was passed as an argument
- `main()`: used for debugging a single file. Calls `loadData(None, None)` and prints the resulting retrieved dictionary entries.

18 MIS of Image Classification Module

18.1 Module

ImageClassification

18.2 Uses

TrainingDataset

18.3 Syntax

18.3.1 Exported Constants

None

18.3.2 Exported Access Programs

Name	In	Out	Exceptions
startModel	seq of (seq of \mathbb{Z})	String	-

18.4 Semantics

18.4.1 State Variables

weights: seq of (seq of \mathbb{Z})

imageArray : seq of (seq of \mathbb{Z})

foodItem: String

18.4.2 Environment Variables

None

18.4.3 Assumptions

It is assumed that there is a relationship between the uploaded image and the image labels that the machine learning model is aware of. It is also assumed that the food in an uploaded image has a one to one relation with a label that the machine learning model is aware of.

18.4.4 Access Routine Semantics

startModel(array):

- transition: *imageArray* := *array*
- output: *out* := *foodItem*
- exception: None

18.4.5 Local Functions

`tf.compat.v1.train.GradientDescentOptimizer(learning_rate).minimize(loss)`: Execute `GradientDescentOptimizer` and tries to minimize loss by computing the gradients of its trainable variables. Optimizes weights system variable on pass.

19 MIS of Nutritional Data Retriever Module

19.1 Module

NutritionalData

19.2 Uses

N/A

19.3 Syntax

19.3.1 Exported Constants

None

19.3.2 Exported Access Programs

Name	In	Out	Exceptions
getNutritionalData	String	tuple of (food_name: String, calories: String, total_fat: String, saturated_fat: String, cholesterol: String, sodium: String, total_carbohydrate: String, dietary_fiber: String, sugars: String, protein: String, potassium: String)	IllegalArgumentException

19.4 Semantics

19.4.1 State Variables

result: tuple of Strings

19.4.2 Environment Variables

None

19.4.3 Assumptions

None

19.4.4 Access Routine Semantics

getNutritionalData(*food_item*):

- output: *result* := tuple of (food_name: String, calories: String, total_fat: String, saturated_fat: String, cholesterol: String, sodium: String, total_carbohydrate: String, dietary_fiber: String, sugars: String, protein: String, potassium: String)
- exception: (*food_item* \Rightarrow *result* := NULL) \Rightarrow IllegalArgumentException

19.4.5 Local Functions

None

20 MIS of Profile Data Calculation Module

20.1 Module

ProfileData

20.2 Uses

os, datetime

20.3 Syntax

20.3.1 Exported Constants

None

20.3.2 Exported Access Programs

Name	In	Out	Exceptions
logData	JSON	-	-
calculateTotalNutrients	JSON	seq of String	-
readFile	-	seq of JSON	-
readFileAsJson	-	JSON	-
totalCaloriesPerDay	Date	\mathbb{R}	-
totalFoodsPerDay	Date	\mathbb{Z}	-
totalCaloriesPerDaySummaryList	-	seq of JSON	-
mostEatenFood	-	\mathbb{R}	-

20.4 Semantics

20.4.1 State Variables

None

20.4.2 Environment Variables

None

20.4.3 Assumptions

None

20.4.4 Access Routine Semantics

logData(foodData):

- transition: $\text{nutritionLog}+ = \text{csvRow} : \text{String}$
- exception: None

calculateTotalNutrients(foodData):

- output: $\text{out} := \{\text{foodName} : \text{String}, \text{calories} : \mathbb{R}, \text{totalFat} : \mathbb{R}, \text{saturatedFat} : \mathbb{R}, \text{cholesterol} : \mathbb{R}, \text{sodium} : \mathbb{R}, \text{totalCarbohydrate} : \mathbb{R}, \text{dietaryFibre} : \mathbb{R}, \text{sugars} : \mathbb{R}, \text{protein} : \mathbb{R}, \text{potassium} : \mathbb{R}\}$
- exception: None

readFile():

- output: $\text{out} := \text{seq of Row} : \text{String}$
- exception: None

readFileAsJson():

- output: $\text{out} := \{\text{timeStamp} : \text{String}, \text{foodName} : \text{String}, \text{calories} : \mathbb{R}, \text{totalFat} : \mathbb{R}, \text{saturatedFat} : \mathbb{R}, \text{cholesterol} : \mathbb{R}, \text{sodium} : \mathbb{R}, \text{totalCarbohydrate} : \mathbb{R}, \text{dietaryFibre} : \mathbb{R}, \text{sugars} : \mathbb{R}, \text{protein} : \mathbb{R}, \text{potassium} : \mathbb{R}\}$
- exception: None

totalCaloriesPerDay(day):

- output: $\text{out} := \sum_{i=0}^{|\text{foods}|-1} \text{food}_i.\text{calories}$
- exception: None

totalFoodsPerDay(day):

- output: $\text{out} := \sum_{i=1}^{|\text{foods}|} 1$
- exception: None

totalCaloriesPerDaySummaryList():

- output: $\text{out} := \text{seq of } \{\text{data}, \text{sumPerDay}, \text{foodsPerDay}\}$
- exception: None

mostEatenFood():

- output: $\text{out} := \text{mode}(\text{seq of foods})$
- exception: None

20.4.5 Local Functions

None

References

- Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.
- Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL <http://citeseer.ist.psu.edu/428727.html>.
- Durum Wheat Semolina. Module guide. <https://github.com/jeff-rey-wang/utrition/blob/39b1a9bda7ec90db43221e17b035e57b7fa29650/docs/Design/MG/MG.pdf>, 2022a.
- Durum Wheat Semolina. System requirements specification. <https://github.com/jeff-rey-wang/utrition/blob/39b1a9bda7ec90db43221e17b035e57b7fa29650/docs/SRS/SRS.pdf>, 2022b.

21 Appendix