Module Interface Specification for Utrition

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

January 18, 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

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 $(c_1 \Rightarrow r_1|c_2 \Rightarrow r_2|...|c_n \Rightarrow r_n)$.

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 (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)

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 Food Entry Module Upload Page Module Image Upload Module Manual Upload Module Voice Upload Module Navigation Bar Module Backend Communication Module
Software Decision Module	Input Pre-Processing Module Training Dataset Module Image Classification Module Nutritional Data Retriever Module User Log Data Structure 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: $path := "/"$
- 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* := getData()
- output: *out* := (*data* != NULL) ⇒ Display component with user data | (*data* == NULL) ⇒ Display component displaying String stating "No user data available"
- exception: None

9.4.5 Local Functions

None

10 MIS of Food Entry Module

10.1 Template Module

FoodEntry

10.2 Uses

N/A

10.3 Syntax

10.3.1 Exported Types

FoodEntry = ?

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
FoodEntry	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)	FoodEntry	-
getFoodName	-	String	-
getFoodInfo	-	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)	-

10.4 Semantics

10.4.1 State Variables

$food\_name$: String
$food\_info$: 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)

10.4.2 Environment Variables

None

10.4.3 Assumptions

The FoodEntry(food_name, food_info) constructor is called for each object instance before any other access routine is called for that object. The constructor can only be called once.

10.4.4 Access Routine Semantics

FoodEntry($item, info$):

- transition: $food\_name, food\_info := item, info$
- output: $out := self$
- exception: None

getFoodName():

- output: $out := food\_name$
- exception: None

getFoodInfo():

- output: $out := food\_info$
- exception: None

10.4.5 Local Functions

None

11 MIS of Upload Page Module

11.1 Module

Upload

11.2 Uses

ImageUpload, ManualUpload, VoiceUpload

11.3 Syntax

11.3.1 Exported Types

Upload = ?

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
Upload	-	Upload	-

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

Upload():

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

11.4.5 Local Functions

None

12 Image Upload Module

12.1 Module

ImageUpload

12.2 Uses

BackComm

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) ⇒ BadResponseError

12.3.7 Local Functions

setImage(s)

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

setResponseData(r)

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

13 Manual Upload Module

13.1 Module

ManualUpload

13.2 Uses

BackComm

13.3 Syntax

13.3.1 Exported Types

ManualUpload = ?

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
ManualUpload	-	ManualUpload	-
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

ManualUpload():

- 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

BackComm

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) ⇒ 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 Backend Communication Module

16.1 Module

BackComm

16.2 Uses

InputPreProcess, NutritionalData

16.3 Syntax

16.3.1 Exported Constants

None

16.3.2 Exported Access Programs

Name	In	Out	Exceptions
displayIndex	-	JSON	None

16.4 Semantics

16.4.1 State Variables

None

16.4.2 Environment Variables

None

16.4.3 Assumptions

It is assumed that the filename of the user inputted file will be of String format.

16.4.4 Access Routine Semantics

displayIndex():

- output: *out* := JSON consisting of the food classified from the user input
- exception: None

16.4.5 Local Functions

None

17 MIS of Input Pre-Processing Module

17.1 Module

InputPreProcess

17.2 Uses

ImageClassification

17.3 Syntax

17.3.1 Exported Constants

None

17.3.2 Exported Access Programs

Name	In	Out	Exceptions
open	String	String	-

17.4 Semantics

17.4.1 State Variables

$filePath$: String
$foodIdentified$: String

17.4.2 Environment Variables

None

17.4.3 Assumptions

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

17.4.4 Access Routine Semantics

open(path):

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

17.4.5 Local Functions

None

18 MIS of Training Dataset Module

18.1 Module

TrainingDataset

18.2 Uses

N/A

18.3 Syntax

18.3.1 Exported Constants

None

18.3.2 Exported Access Programs

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

18.4 Semantics

18.4.1 State Variables

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

18.4.2 Environment Variables

None

18.4.3 Assumptions

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

18.4.4 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

18.4.5 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.

19 MIS of Image Classification Module

19.1 Module

ImageClassification

19.2 Uses

TrainingDataset

19.3 Syntax

19.3.1 Exported Constants

None

19.3.2 Exported Access Programs

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

19.4 Semantics

19.4.1 State Variables

$weights$: seq of (seq of $\mathbb{Z}$)
$imageArray$: seq of (seq of $\mathbb{Z}$)
$foodItem$: String

19.4.2 Environment Variables

None

19.4.3 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.

19.4.4 Access Routine Semantics

startModel(array):

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

19.4.5 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.

20 MIS of Nutritional Data Retriever Module

20.1 Module

NutritionalData

20.2 Uses

N/A

20.3 Syntax

20.3.1 Exported Constants

None

20.3.2 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

20.4 Semantics

20.4.1 State Variables

result: tuple of Strings

20.4.2 Environment Variables

None

20.4.3 Assumptions

None

20.4.4 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

20.4.5 Local Functions

None

21 MIS of User Log Data Structure Module

21.1 Module

UserLogData

21.2 Uses

FoodEntry

21.3 Syntax

21.3.1 Exported Constants

None

21.3.2 Exported Access Programs

Name	In	Out	Exceptions
addData	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)	-	-
getData	-	seq of FoodEntry	-

21.4 Semantics

21.4.1 State Variables

userData: seq of FoodEntry

21.4.2 Environment Variables

None

21.4.3 Assumptions

None

21.4.4 Access Routine Semantics

addData($food\_item$, $food\_info$):

- transition: $userData$ += new FoodEntry($food\_item$, $food\_info$)
- exception: None

getData():

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

21.4.5 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.

22 Appendix