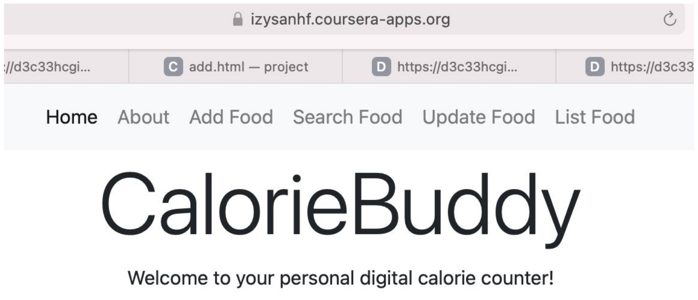
D3: List of requirements: for each sub-requirement (R1A → R6C) state how this was achieved or if it was not achieved. Explain where it can be found in the code. Use focused, short code extracts if they make your explanation clearer.

- **R1: Home page:**
  - R1A

As seen in Figure 1, on the main page we can view the name of the application CalorieBuddy, from the index.html file. The code is found in the index.html file, present in the views folder. The Home page is rendered through the corresponding route from the main.js file.



CalorieBuddy

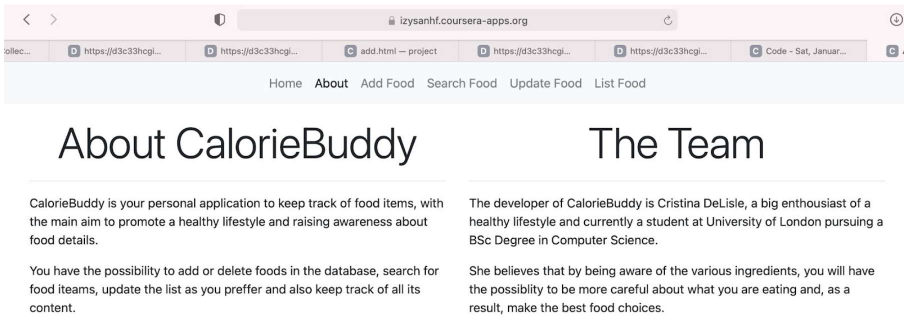
Welcome to your personal digital calorie counter!

Figure 1: Home Page

  - R1B

A navigation bar is present displaying links to other pages from the application, including the Home page: About, Add Food, Search Food, Update Food and List Food. The text is marked as active in the index.html file when the user is present on one of these pages, such that it becomes accentuated when the user is present on one of these pages.

- **R2: About page:**
  - R2A



About CalorieBuddy

CalorieBuddy is your personal application to keep track of food items, with the main aim to promote a healthy lifestyle and raising awareness about food details.

You have the possibility to add or delete foods in the database, search for food iteams, update the list as you preffer and also keep track of all its content.

The Team

The developer of CalorieBuddy is Cristina DeLisle, a big enthousiast of a healthy lifestyle and currently a student at University of London pursuing a BSc Degree in Computer Science.

She believes that by being aware of the various ingredients, you will have the possiblity to be more careful about what you are eating and, as a result, make the best food choices.

Figure 2: About Page

This page shows information about the application CalorieBuddy and also about the developer of the application, as seen in Figure 2. The code is found in the about.html file, present in the views folder. The About Page is rendered through the corresponding route from the main.js file.

A navigation bar is present displaying links to pages from the application: Home page, About, Add Food, Search Food, Update Food and List Food. The text is marked as active in the about.html file when the user is present on one of these pages, such that it becomes accentuated when the user is present on one of these pages.

- **R3: Add food page:**
    - R3A

This page displayed a form with the following Items to fill in and an Add Food Item, as soon in Figure 3: Name of food, Typical values per, Unit of the typical value, Calories (kilocalories), Carbs, Fat, Protein, Salt, and Sugar. The code is found in the add.html file, present in the views folder. The Add Food Page is rendered through the corresponding route from the main.js file.

A navigation bar is present displaying links to pages from the application: Home page, About, Add Food, Search Food, Update Food and List Food. The text is marked as active in the add.html file, such that it becomes accentuated when the user is present on one of these pages.

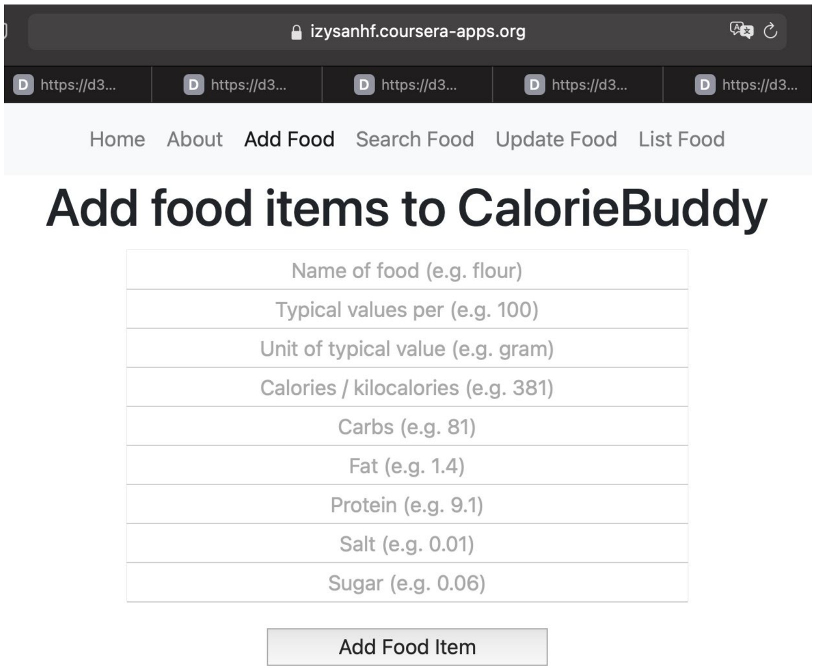
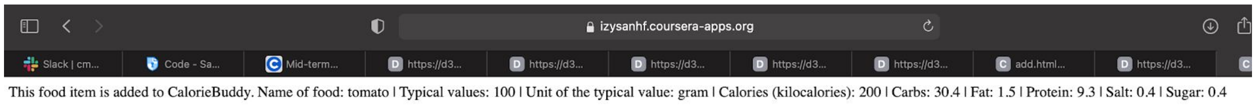

izysanhf.coursera-apps.org
Home
About
Add Food
Search Food
Update Food
List Food
Add food items to CalorieBuddy
Name of food (e.g. flour)
Typical values per (e.g. 100)
Unit of typical value (e.g. gram)
Calories / kilocalories (e.g. 381)
Carbs (e.g. 81)
Fat (e.g. 1.4)
Protein (e.g. 9.1)
Salt (e.g. 0.01)
Sugar (e.g. 0.06)
Add Food Item

Figure 3: Add food items form



This food item is added to CalorieBuddy. Name of food: tomato | Typical values: 100 | Unit of the typical value: gram | Calories (kilocalories): 200 | Carbs: 30.4 | Fat: 1.5 | Protein: 9.3 | Salt: 0.4 | Sugar: 0.4

Figure 4: Add items

- R3B

The form collects the data inserted inside the database calorieBuddy, using the POST method in the corresponding route from the main.js file.

The data currently is passed, therefore, to the back-end (database) and stored there.

As seen in Figure 3, each food item consists of the following fields: name, typical values, unit of the typical value, calories, carbs, fat, protein, salt, and sugar.

The new food item inserted can be seen in the List food section, ordered accordingly.

- R3C

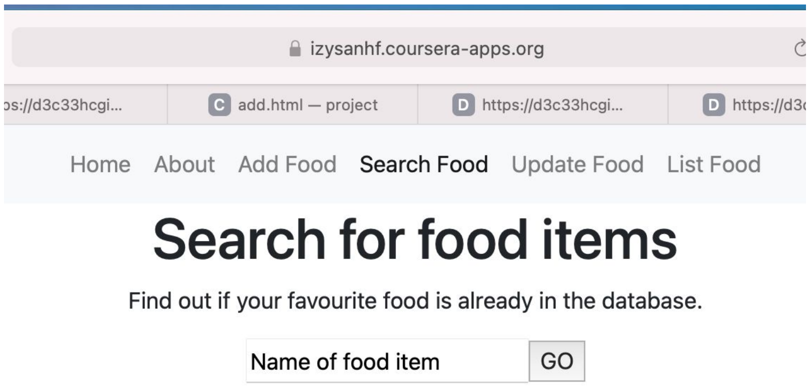
As seen in Figure 4, after the new food item was passed and stored in the database calorieBuddy, a message is displayed confirming that "This food item is added to CalorieBuddy.", together with some extra information about the details inserted.

This confirms to the user that the add operation has been done.

- **R4: Search food page:**
  - R4A

As seen in the figure 5, the Search Food page is displaying a simple form which helps the user search the Name of food item, by pressing the GO button. The code is found in the search.html file, present in the views folder. The Search Page is rendered through the corresponding route from the main.js file.

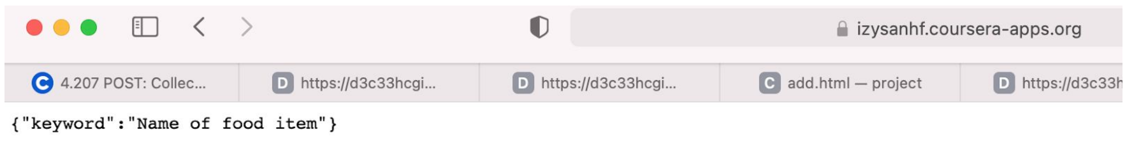
A navigation bar is present displaying links to pages from the application: Home page, About, Add Food, Search Food, Update Food and List Food. The text is marked as active in the search.html file, such that it becomes accentuated when the user is present on one of these pages.



Search for food items

Find out if your favourite food is already in the database.

Figure 5: Search food Page



```
{"keyword":"Name of food item"}
```

Figure 6: Search items

- R4B

The form collects the data inserted inside req.query, as a temporary solution. As seen in Figure 6, the information is collected and is displayed.

Using the GET method in the route present in the main.js file from the routes folder, the next step would be to implement searching the data in the database calorieBuddy.

The data currently is not passed to the back-end (database) and this will be further implemented in a second version of the application.

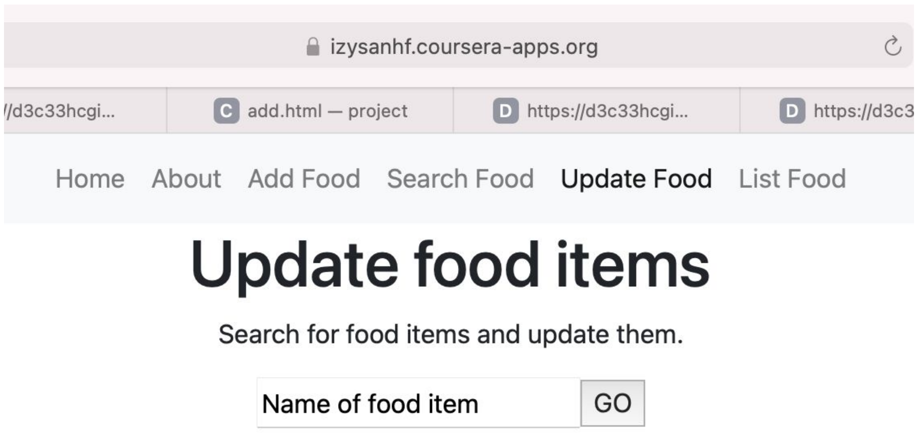
- R4C

This will be further implemented, once the requirements from R4B will be executed.

- **R5: Update food page:**
  - R5A

As seen in figure 7, the Update Food page is displaying a simple form which helps the user search the Name of food item, by pressing the GO button. The code is found in the update.html file, present in the views folder. The Update Page is rendered through the corresponding route from the main.js file.

A navigation bar is present displaying links to pages from the application: Home page, About, Add Food, Search Food, Update Food and List Food. The text is marked as active in the update.html file, such that it becomes accentuated when the user is present on one of these pages.


izysanhf.coursera-apps.org

Update food items

Search for food items and update them.

Figure 7: Update food items Search bar

- R5B

The form collects the data inserted inside req.query, as a temporary solution. As seen in Figure 6, the information is collected and is displayed.

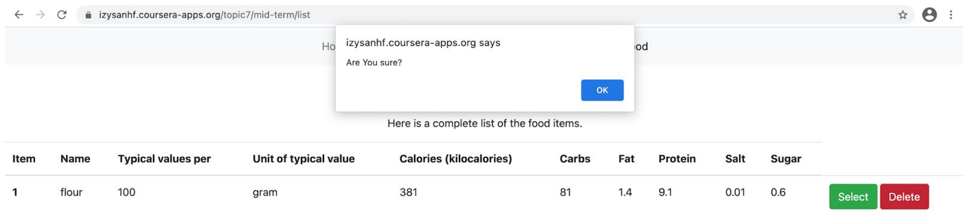
Using the GET method in the route present in the main.js file from the routes folder, the next step would be to implement searching the data in the database calorieBuddy.

The data currently is not passed to the back-end (database) and this will be further implemented in a second version of the application.

This is a blocker for the requirement of storing the details collected in the database, displaying the message that the searched item was not found and also that the details were updated in the database calorieBuddy.

- R5C

This requirement will be further implemented. In Figure 8, at the List page, there is a Delete button present which displays the best practice of asking if the user is sure about deleting the item. If pressed, currently it displays the Error: ENOENT: no such file or directory, open '/home/travis/build/codercom/code-server/packages/server/build/web/topic7/data/delete/3'.


izysanhf.coursera-apps.org/topic7/mid-term/list

izysanhf.coursera-apps.org says

Are You sure?

Here is a complete list of the food items.

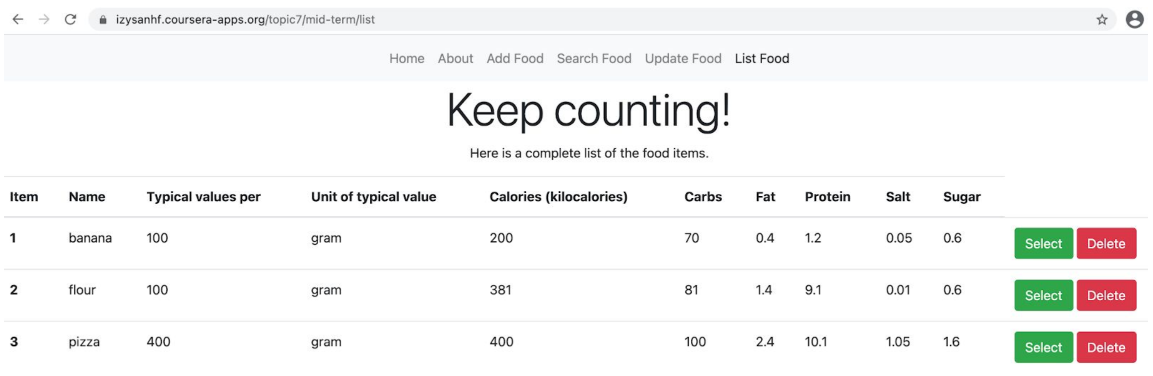
Item	Name	Typical values per	Unit of typical value	Calories (kilocalories)	Carbs	Fat	Protein	Salt	Sugar	
1	flour	100	gram	381	81	1.4	9.1	0.01	0.6	Select Delete

Figure 8: Delete button message

- **R6: List foods page**
  - R6A

As seen in Figure 9, on the List Food page all foods stored in the database including name, typical values, unit of the typical value, calories, carbs, fat, protein, salt, and sugar. The code is found in the list.html file, present in the views folder. The List Page is rendered through the corresponding route from the main.js file.

Even if they've received an Item number to be displayed, they are ordered by name, in alphabetical order. This is achieved through the operation SELECT * FROM food ORDER BY name ASC, present in the main.js file from the routes folder.



Keep counting!

Here is a complete list of the food items.

Item	Name	Typical values per	Unit of typical value	Calories (kilocalories)	Carbs	Fat	Protein	Salt	Sugar	
1	banana	100	gram	200	70	0.4	1.2	0.05	0.6	Select Delete
2	flour	100	gram	381	81	1.4	9.1	0.01	0.6	Select Delete
3	pizza	400	gram	400	100	2.4	10.1	1.05	1.6	Select Delete

Figure 9: List Page

- R6B

A navigation bar is present displaying links to pages from the application: Home page, About, Add Food, Search Food, Update Food and List Food. The text is marked as active in the list.html file (from the views folder), such that it becomes accentuated when the user is present on one of these pages.

- R6C

This requirement will be further implemented. As seen in Figure 9, a Select button is displayed with the intention of selecting the items to be further used for the calculations. If pressed, currently it displays the error: “Error: ENOENT: no such file or directory, open '/home/travis/build/codercom/code-server/packages/server/build/web/topic7/food/edit/3'”.

D4: Database structure: tables including purpose, field names, and data types for each table.

The database calorieBuddy has one table called food. The purpose is to store the various information needed to be used in the application: name, typical values, unit of the typical value, calories, carbs, fat, protein, salt, and sugar.

In order to achieve this, the table food consists of the following fields:

- id
- name
- typical_values

- unit_of_typical_value
- calories
- carbs
- fat
- protein
- salt
- sugar
- primary key (id)

The data types for each item present in the food table are the following:

- **id INT AUTO_INCREMENT**

The AUTO_INCREMENT attribute can be used to generate a unique identity for new rows.

- **name VARCHAR(50)**

This is a VARCHAR type of the form varchart(n), requiring a storage of 2 bytes + number of chars. Values in VARCHAR columns are variable-length strings.

- **typical_values DECIMAL(5, 2) unsigned**

This is a decimal data type (numeric data type) of the form DECIMAL(size, d). The total number of digits is specified in size (5). The number of digits after the decimal point is specified in the d parameter (2).

- **unit_of_typical_value VARCHAR(50)**

This is a VARCHAR type of the form varchart(n), requiring a storage of 2 bytes + number of chars. Values in VARCHAR columns are variable-length strings.

- **calories DECIMAL(5, 2) unsigned**

This is a decimal data type (numeric data type) of the form DECIMAL(size, d). The total number of digits is specified in size (5). The number of digits after the decimal point is specified in the d parameter (2).

- **carbs DECIMAL(5, 2) unsigned**

This is a decimal data type (numeric data type) of the form DECIMAL(size, d). The total number of digits is specified in size (5). The number of digits after the decimal point is specified in the d parameter (2).

- **fat DECIMAL(5, 2) unsigned**

This is a decimal data type (numeric data type) of the form DECIMAL(size, d). The total number of digits is specified in size (5). The number of digits after the decimal point is specified in the d parameter (2).

- **protein DECIMAL(5, 2) unsigned**

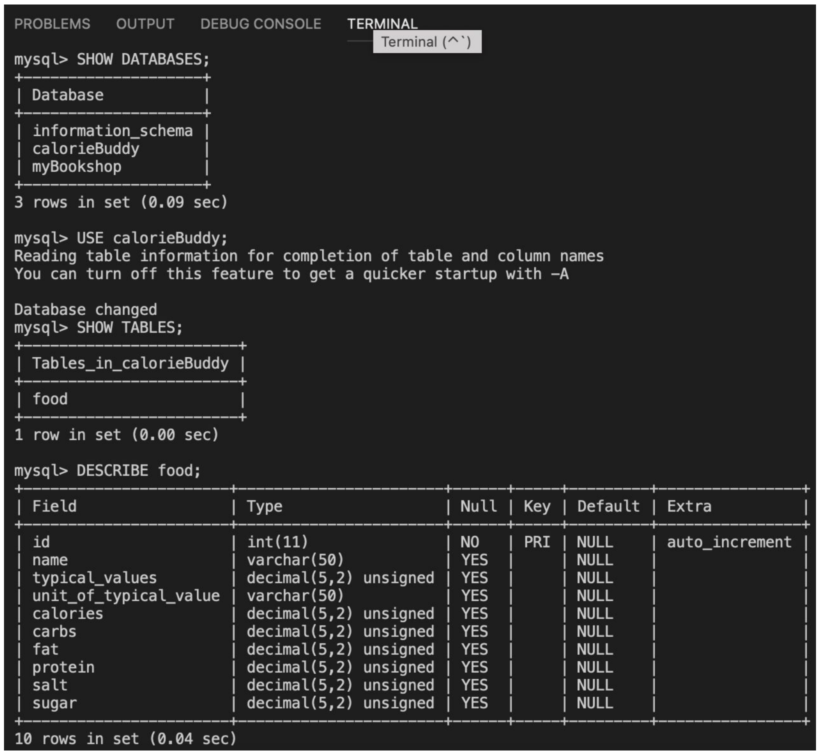
This is a decimal data type (numeric data type) of the form DECIMAL(size, d). The total number of digits is specified in size (5). The number of digits after the decimal point is specified in the d parameter (2).

- **salt DECIMAL(5, 2) unsigned**
  This is a decimal data type (numeric data type) of the form DECIMAL(size, d). The total number of digits is specified in size (5). The number of digits after the decimal point is specified in the d parameter (2).

- **sugar DECIMAL(5, 2) unsigned**
  This is a decimal data type (numeric data type) of the form DECIMAL(size, d). The total number of digits is specified in size (5). The number of digits after the decimal point is specified in the d parameter (2).

- The **PRIMARY KEY(id)** constraint uniquely identifies each record in a table. It contains UNIQUE values (the id for the food table), and cannot contain NULL values. A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

The details analyzed in this requirement can also be seen in figure 10, which shows the calorieBuddy Database, with the food table and its fields.



```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| calorieBuddy       |
| myBookshop         |
+--------------------+
3 rows in set (0.09 sec)

mysql> USE calorieBuddy;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+------------------------+
| Tables_in_calorieBuddy |
+------------------------+
| food                   |
+------------------------+
1 row in set (0.00 sec)

mysql> DESCRIBE food;
+-----------------------+-----------------------+------+-----+---------+----------------+
| Field                 | Type                  | Null | Key | Default | Extra          |
+-----------------------+-----------------------+------+-----+---------+----------------+
| id                    | int(11)               | NO   | PRI | NULL    | auto_increment |
| name                  | varchar(50)           | YES  |     | NULL    |                |
| typical_values        | decimal(5,2) unsigned | YES  |     | NULL    |                |
| unit_of_typical_value | varchar(50)           | YES  |     | NULL    |                |
| calories              | decimal(5,2) unsigned | YES  |     | NULL    |                |
| carbs                 | decimal(5,2) unsigned | YES  |     | NULL    |                |
| fat                   | decimal(5,2) unsigned | YES  |     | NULL    |                |
| protein               | decimal(5,2) unsigned | YES  |     | NULL    |                |
| salt                  | decimal(5,2) unsigned | YES  |     | NULL    |                |
| sugar                 | decimal(5,2) unsigned | YES  |     | NULL    |                |
+-----------------------+-----------------------+------+-----+---------+----------------+
10 rows in set (0.04 sec)
```

Figure 10: calorieBuddy Database