

Requirements:

R1: Home Page

In this requirement, we need to display the name of the web application. I created a home.html file under the views folder as seen below. Inside the home.html file, I included the name of the web application, CalorieBuddy, which can be seen under `<h1> CalorieBuddy </h1>` and this will display the name when being executed.

```
1  <!doctype html>
2  <html>
3  <head>
4    <title>Home Page! - Welcome to CalorieBuddy</title>
5  </head>
6  <body>
7    <h1> CalorieBuddy </h1>
8    <h2> Welcome to CalorieBuddy - We will help you meet your nutritional goals today! </h2>
9
10   <p style="padding: 10px; border: 2px solid red;"> <a href="/topic7/mid-term/about">About</a> </p>
11   <p style="padding: 10px; border: 2px solid orange;"> <a href="/topic7/mid-term/addfood">Add Food</a> </p>
12   <p style="padding: 10px; border: 2px solid green;"> <a href="/topic7/mid-term/search">Search Food</a> </p>
13   <p style="padding: 10px; border: 2px solid blue;"> <a href="/topic7/mid-term/update">Update Food</a> </p>
14   <p style="padding: 10px; border: 2px solid violet;"> <a href="/topic7/mid-term/list">View all the food</a> </p>
15
16 </body>
17 </html>
```

To display the links to other pages, for instance, the link to the About page, I used `About`.

This applies for other pages as well.

`Add Food` will display Add Food! Page.

`Search Food` will display Search Food! Page

`Update Food` will display Update Food! Page

`View all the food ` will display Update Food! Page

This will allow the user to click the link and will be redirected to the desired page. This can be achieved by using the GET method on the main.js file.

On the main.js file located on the routes folder, I render the home.html file by using

```
app.get("/home",function(req, res){
  res.render("home.html")
})
```

JS main.js



<> list.html

<> home.html

```
1 module.exports = function(app) {  
2   app.get("/about",function(req, res){  
3     res.render("about.html")  
4   });  
5  
6   app.get("/home",function(req, res){  
7     res.render("home.html")  
8   });  
9 }
```

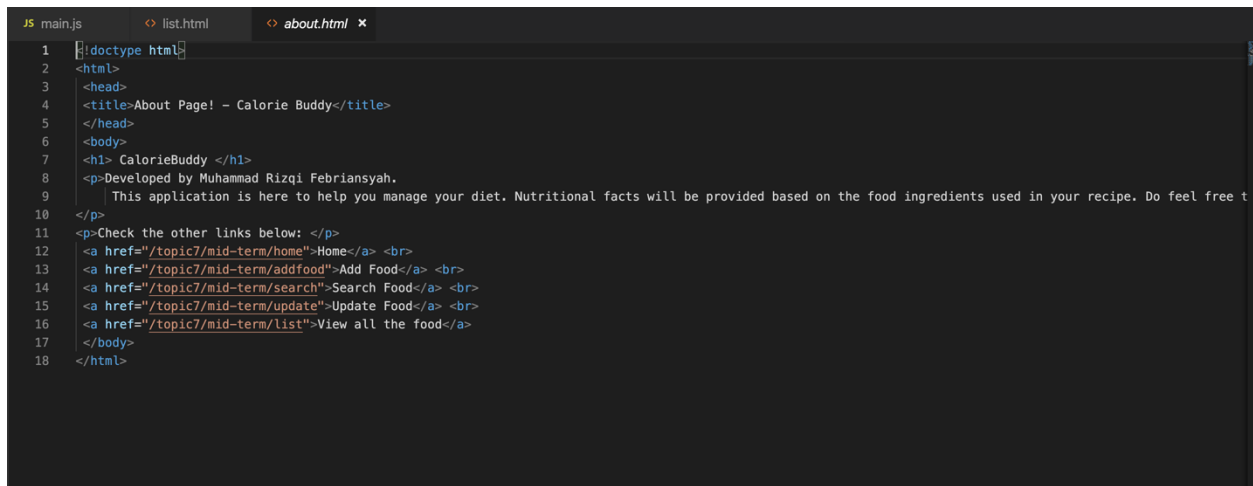
R2: About Page

In this requirement, we need to display information regarding our web application, CalorieBuddy, and also to display my name as the developer. I created an about.html file under the views folder as seen below. Inside the home.html file, I included the name of the web application, CalorieBuddy, which can be seen under `<h1> CalorieBuddy </h1>` and this will display the name when being executed. Under the `<p> </p>`, this is where I would include my name and a short description of my web application as seen below.

`<p>`Developed by Muhammad Rizqi Febriansyah.

This application is here to help you manage your diet. Nutritional facts will be provided based on the food ingredients used in your recipe. Do feel free to add, update, search and delete food.

`</p>`



```
1 <!doctype html>
2 <html>
3 <head>
4 <title>About Page! - Calorie Buddy</title>
5 </head>
6 <body>
7 <h1> CalorieBuddy </h1>
8 <p>Developed by Muhammad Rizqi Febriansyah.
9   This application is here to help you manage your diet. Nutritional facts will be provided based on the food ingredients used in your recipe. Do feel free t
10 </p>
11 <p>Check the other links below: </p>
12 <a href="/topic7/mid-term/home">Home</a> <br>
13 <a href="/topic7/mid-term/addfood">Add Food</a> <br>
14 <a href="/topic7/mid-term/search">Search Food</a> <br>
15 <a href="/topic7/mid-term/update">Update Food</a> <br>
16 <a href="/topic7/mid-term/list">View all the food</a>
17 </body>
18 </html>
```

To display the links to other pages, for instance, the link to the Home page, I used `Home`.

This applies for other pages as well.

`Add Food` will display Add Food! Page.

`Search Food` will display Search Food! Page

`Update Food` will display Update Food! Page

`View all the food` will display Update Food! Page

This will allow the user to click the link and will be redirected to the desired page. This can be achieved by using the GET method on the main.js file.

On the main.js file located on the routes folder, I render the about.html file by using

```
app.get("/about",function(req, res){
  res.render("about.html")
})
```

JS main.js



<> list.html

<> home.html

```
1 module.exports = function(app) {  
2   app.get("/about",function(req, res){  
3     res.render("about.html")  
4   });  
5  
6   app.get("/home",function(req, res){  
7     res.render("home.html")  
8   });  
9 }
```

R3: Add Food Page

In this requirement, we need to allow users to add a new food item to the database. The form should include the food name, typical values per, unit of measurement, calories, carbs, fat, protein, salt and sugar.

This can be achieved in the addfood.html under views folder.

For the form to include all the names, I labelled each and adding `<input>`.

For instance, for Food name,

`<input id="Food name: " type="text" name="name" value="foodname">`

This applies to the rest as well.

To display the links to other pages, for instance, the link to the Home page, I used `Home`.

This applies for other pages as well. Once clicked, user will be redirected to the desired page.

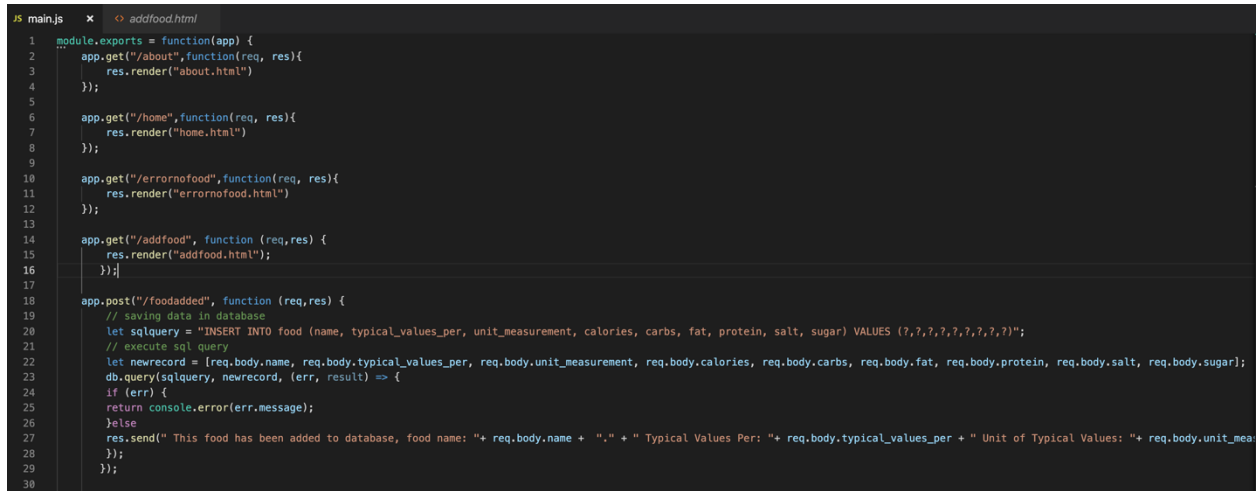
```
J5 main.js    addfood.html x
1  <!doctype html>
2  <html>
3    <head>
4      <title>Add Food Page!</title>
5    </head>
6
7    <body>
8      <h1> This is the Add Food Page </h1>
9      <p>Please fill in related data to add food to the database</p>
10
11      <form method= "POST" action= "/topic7/mid-term/foodadded">
12        Food name:
13        <input id="Food name: " type="text" name="name" value="foodname"> <br></br>
14        Typical Values per:
15        <input id="Typical values per: " type="number" step="any" name="typical_values_per" value="0"> <br></br>
16        Units of Typical Value – Measurement:
17        <input id="Units of measurement: " type="text" name="unit_measurement" value="unitOfMeasurement"> <br></br>
18        Calories:
19        <input id="Calories: " type="number" step="any" name="calories" value="0"> <br></br>
20        Carbs:
21        <input id="Carbs: " type="number" step="any" name="carbs" value="0"> <br></br>
22        Fat:
23        <input id="Fat: " type="number" step="any" name="fat" value="0"> <br></br>
24        Protein:
25        <input id="Protein: " type="number" step="any" name="protein" value="0"> <br></br>
26        Salt:
27        <input id="Salt: " type="number" step="any" name="salt" value="0"> <br></br>
28        Sugar:
29        <input id="Sugar: " type="number" step="any" name="sugar" value="0"> <br></br>
30        <input type="submit" value="Submit">
31      </form>
32
33      <br>
34      <a href="/topic7/mid-term/home">Home</a> <br>
35      <a href="/topic7/mid-term/addfood">Add Food</a> <br>
36      <a href="/topic7/mid-term/search">Search Food</a> <br>
37      <a href="/topic7/mid-term/update">Update Food</a> <br>
38      <a href="/topic7/mid-term/list">View all the food</a>
39
40    </body>
41  </html>
```

This can be achieved by using the GET method on the main.js file.

On the main.js file located on the routes folder, I render the addfood.html file by using `app.get("/addfood",function(req, res){`
`res.render("addfood.html")`

To add and store food inside the database, I use POST method under addfood.html.

<form method= "POST" action= "/topic7/mid-term/foodadded"> </form>



```
1 module.exports = function(app) {
2   ...
3   app.get("/about",function(req, res){
4     res.render("about.html")
5   });
6   app.get("/home",function(req, res){
7     res.render("home.html")
8   });
9   app.get("/errornofood",function(req, res){
10    res.render("errornofood.html")
11  });
12
13  app.get("/addfood", function (req,res) {
14    res.render("addfood.html");
15  });
16
17  app.post("/foodadded", function (req,res) {
18    // saving data in database
19    let sqlquery = "INSERT INTO food (name, typical_values_per, unit_measurement, calories, carbs, fat, protein, salt, sugar) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
20    // execute sql query
21    let newrecord = [req.body.name, req.body.typical_values_per, req.body.unit_measurement, req.body.calories, req.body.carbs, req.body.fat, req.body.protein, req.body.salt, req.body.sugar];
22    db.query(sqlquery, newrecord, (err, result) => {
23      if (err) {
24        return console.error(err.message);
25      }else
26        res.send(" This food has been added to database, food name: "+ req.body.name + ". " + " Typical Values Per: "+ req.body.typical_values_per + " Unit of Typical Values: "+ req.body.unit_measurement + " Calories: " + req.body.calories + " Carbs: " + req.body.carbs + " Fat: " + req.body.fat + " Protein: "+ req.body.protein + " Salt: " + req.body.salt + " Sugar: "+ req.body.sugar);
27      });
28    });
29  });
30 }
```

On the main.js file located on the routes folder, you call the POST method.

```
app.post("/foodadded", function (req,res) {
  // saving data in database
  let sqlquery = "INSERT INTO food (name, typical_values_per, unit_measurement, calories,
carbs, fat, protein, salt, sugar) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
```

This code above will save the data in the database.

```
let newrecord = [req.body.name, req.body.typical_values_per, req.body.unit_measurement,
req.body.calories, req.body.carbs, req.body.fat, req.body.protein, req.body.salt, req.body.sugar];
db.query(sqlquery, newrecord, (err, result) =>
```

This code above will execute the sql query.

```
res.send(" This food has been added to database, food name: "+ req.body.name + ". " + " Typical
Values Per: "+ req.body.typical_values_per + " Unit of Typical Values: "+
req.body.unit_measurement + " Calories: " + req.body.calories + " Carbs: " + req.body.carbs + "
Fat: " + req.body.fat + " Protein: "+ req.body.protein + " Salt: " + req.body.salt + " Sugar: "+
req.body.sugar);
});
```

This code above will display message that food has been added.

R4: Search Food Page

In this requirement, we need to allow users to search food in the database. The form should include only one field, which is the name of the food.

This can be achieved in the search.html under views folder.

For the form to only display one field, I labelled and added only one <input>.

For instance,

```
<input id="search-box" type="text" name="keyword" value="Default">
```

will display only one field, which is to search for the name of the food.

```
JS main.js    <> search.html x
1  <!doctype html>
2  <html>
3    <head>
4      <title>Search Food Page!</title>
5    </head>
6    <body>
7      <h1> Let's look for your food! Yummy </h1>
8      <p>Do type below the food that you are looking for:</p>
9      <form action="/topic7/mid-term/search-result-db" method="GET">
10     <input id="search-box" type="text" name="keyword" value="Default">
11     <input type="submit" value="Submit" >
12   </form>
13
14   <br>
15   <a href="/topic7/mid-term/home">Home</a> <br>
16   <a href="/topic7/mid-term/addfood">Add Food</a> <br>
17   <a href="/topic7/mid-term/about">About Food</a> <br>
18   <a href="/topic7/mid-term/update">Update Food</a> <br>
19   <a href="/topic7/mid-term/list">View all the food</a>
20
21 </body>
22 </html>
```

To display the links to other pages, for instance, the link to the Home page, I used Home".

This applies for other pages as well. Once clicked, user will be redirected to the desired page.

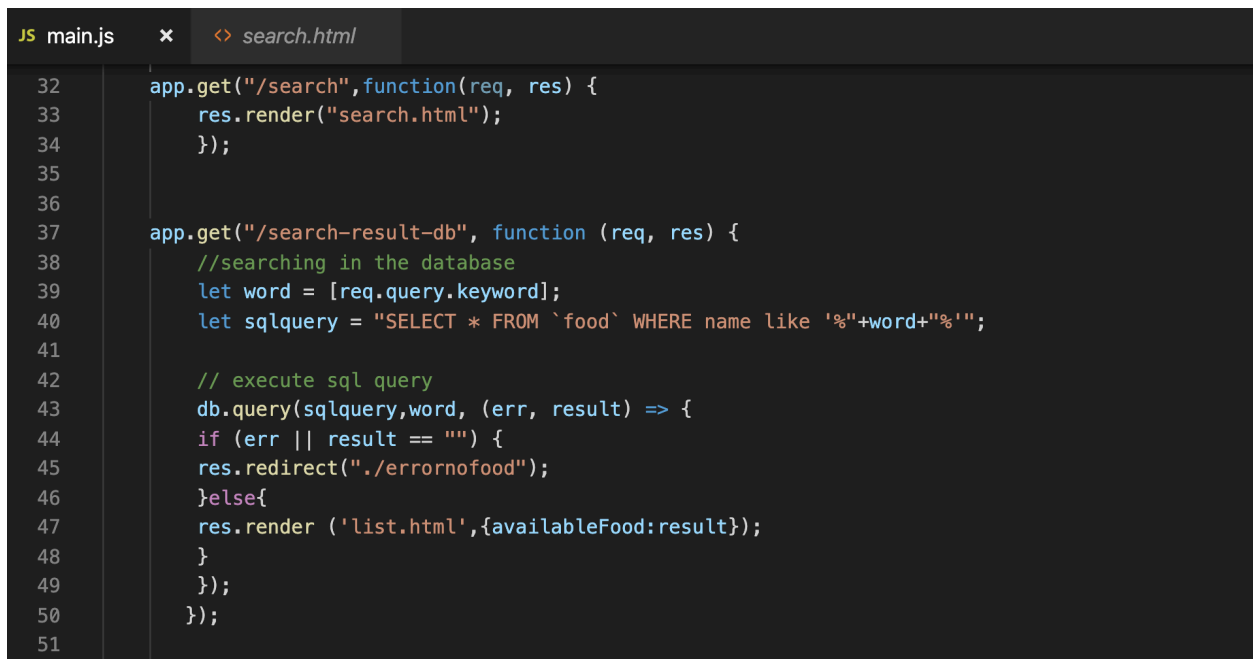
This can be achieved by using the GET method on the main.js file.

On the main.js file located on the routes folder, I render the search.html file by using

```
app.get("/search",function(req, res){  
    res.render("search.html")  
})
```

To display the search result of the food, I use GET method from the search.html form action="/topic7/mid-term/search-result-db" method="GET">

This code syntax above will initiate and call the GET method located on the main.js file.

A screenshot of a code editor with two tabs: 'main.js' and 'search.html'. The 'main.js' tab is active, showing a JavaScript file with line numbers 32 to 51. The code implements two GET routes. The first route at line 32 is for '/search', which renders 'search.html'. The second route at line 37 is for '/search-result-db', which performs a database query. It extracts the keyword from the request, constructs an SQL query to search for food items with names containing the keyword, executes the query, and then either redirects to an error page or renders 'list.html' with the search results.

```
32 app.get("/search",function(req, res) {  
33     res.render("search.html");  
34 });  
35  
36  
37 app.get("/search-result-db", function (req, res) {  
38     //searching in the database  
39     let word = [req.query.keyword];  
40     let sqlquery = "SELECT * FROM `food` WHERE name like '%" + word + "%'";  
41  
42     // execute sql query  
43     db.query(sqlquery,word, (err, result) => {  
44         if (err || result == "") {  
45             res.redirect("./errornofood");  
46         }else{  
47             res.render ('list.html',{availableFood:result});  
48         }  
49     });  
50 }  
51
```

```
pp.get("/search-result-db", function (req, res) {  
    //searching in the database  
    let word = [req.query.keyword];  
    let sqlquery = "SELECT * FROM `food` WHERE name like '%" + word + "%'";
```

This code syntax above will search for the food in the database.

```
let sqlquery = "SELECT * FROM `food` WHERE name like '%" + word + "%'";
```

This code syntax above will allow user to search all food containing similar names stored in the database.


```
db.query(sqlquery,word, (err, result) => {  
  if (err || result == "") {  
    res.redirect("./errornofood");  
  }else{  
    res.render ('list.html',{availableFood:result});  
  }  
});  
});
```

This code syntax above will execute the sql query.

```
if (err || result == "") {  
  res.redirect("./errornofood");  
}
```

This code syntax above will display a message to the user that food is not found in the database.

R5: Update Food Page

In this requirement, we need to display the search food form.

```
<input id="search-box" type="text" name="keyword" value="Default">
```

This syntax code above will display search food form.

```
JS main.js  <> update.html x

1  <!doctype html>
2  <html>
3  <head>
4  <title>Update Food Page!</title>
5  </head>
6  <body>
7  <h1> Let's look for your food! </h1>
8  <p>Do type below the food you are looking for:</p>
9  <form action="/topic7/mid-term/update-result-db" method="GET">
10 <input id="search-box" type="text" name="keyword" value="Default">
11 <input type="submit" value="Submit" >
12 </form>
13
14 <br>
15 <a href="/topic7/mid-term/home">Home</a>
16 <a href="/topic7/mid-term/addfood">Add Food</a>
17 <a href="/topic7/mid-term/about">About Food</a>
18 <a href="/topic7/mid-term/update">Update Food</a>
19 <a href="/topic7/mid-term/list">View all the food</a>
20
21 </body>
22 </html>

JS main.js x  <> update.html
65
66 app.get("/update", function (req,res) {
67   res.render("update.html");
68 });
69
70 app.get("/update-result-db", function (req, res) {
71   //searching in the database
72   let word = [req.query.keyword];
73   let sqlquery = "SELECT * FROM `food` WHERE name Like ?";
74
75   // execute sql query
76   db.query(sqlquery,word, (err, result) => {
77     if (err || result == "") {
78       res.redirect("/errornofood");
79       //res.redirect("/search"); this can also be used in case of an error instead of the above line
80     }else{
81       res.render ('updatesearch.html',{availableFood:result});
82     }
83   });
84 });
85
86
87 app.post("/foodupdated", function (req,res) {
88   // saving data in database
89   let sqlquery = "UPDATE food SET name = ?, typical_values_per = ?, unit_measurement = ?, calories = ?, carbs = ?, fat = ?, protein = ?, salt = ?, sugar =? WHERE name = ?";
90   // execute sql query
91   let newrecord = [req.body.name, req.body.typical_values_per, req.body.unit_measurement, req.body.calories, req.body.carbs, req.body.fat, req.body.protein, req.body.salt, req.body.sugar, req
92   db.query(sqlquery, newrecord, (err, result) => {
93     if (err) {
94       return console.error(err.message);
95     }else
96     res.send(" This food has been updated to database, food name: "+ req.body.name + + ".");
97   });
98 });
99
```

```

JS main.js  updatesearch.html x
1  <!doctype html>
2  <html>
3    <head>
4      <title>Update Food Page!</title>
5    </head>
6
7    <body>
8
9      <h1> This is the Update Food Page </h1>
10     <p>Feel free to update or delete your food.</p>
11
12     <style>
13       table, th, td {
14         border: 1px solid black;
15         border-collapse: collapse;
16       }
17       th, td {
18         padding: 5px;
19       }
20     </style>
21
22     <h2>You can see names and nutritional facts of the searched food here:</h2>
23
24     <table style="width:70%">
25
26       <tr>
27         <th>Food Name</th>
28         <th>Typical Values Per</th>
29         <th>Unit of Typical Values </th>
30         <th>Calories</th>
31         <th>Carbs</th>
32         <th>Fat</th>
33         <th>Protein</th>
34         <th>Salt</th>
35         <th>Sugar</th>
36       </tr>
37
38       <%
39         availableFood.forEach(function(food){
40           %>
41
42           <tr>
43             <td><%= food.name %></td>
44             <td><%= food.typical_values_per %></td>
45             <td><%= food.unit_measurement %></td>
46             <td><%= food.calories %></td>

```

<form method= "POST" action= "/topic7/mid-term/deletefood" onsubmit="return confirm('Are you sure?') ">

Food name:

<input id="Food name: " type="text" name="name" value="foodname">
</br>

<button>

<input type="hidden" name="delete" value="">

Delete

</button>

</form>

This code syntax above in updatesearch.html will display the delete message.

```

JS main.js updatesearch.html x
45         <td><%= food.unit_measurement %></td>
46         <td><%= food.calories %></td>
47         <td><%= food.carbs %></td>
48         <td><%= food.fat %></td>
49         <td><%= food.protein %></td>
50         <td><%= food.salt %></td>
51         <td><%= food.sugar %></td>
52     </tr>
53
54     <% }) %>
55 </table>
56 <br></br>
57 <p>Please key in related data field to update food with new values: </p>
58 <form method= "POST" action= "/topic7/mid-term/foodupdated">
59     Food name:
60     <input id="Food name: " type="text" name="name" value="foodname"> <br></br>
61     Typical Values per:
62     <input id="Typical values per: " type="number" step="any" name="typical_values_per" value="0"> <br></br>
63     Units of Typical Value ~ Measurement:
64     <input id="Units of measurement: " type="text" name="unit_measurement" value="unitOfMeasurement"> <br></br>
65     Calories:
66     <input id="Calories: " type="number" step="any" name="calories" value="0"> <br></br>
67     Carbs:
68     <input id="Carbs: " type="number" step="any" name="carbs" value="0"> <br></br>
69     Fat:
70     <input id="Fat: " type="number" step="any" name="fat" value="0"> <br></br>
71     Protein:
72     <input id="Protein: " type="number" step="any" name="protein" value="0"> <br></br>
73     Salt:
74     <input id="Salt: " type="number" step="any" name="salt" value="0"> <br></br>
75     Sugar:
76     <input id="Sugar: " type="number" step="any" name="sugar" value="0"> <br></br>
77     <input type="submit" value="Submit">
78 </form>
79
80 <br></br>
81 <p>Please key in the desired food to be deleted from database</p>
82 <form method= "POST" action= "/topic7/mid-term/deletefood" onsubmit="return confirm('Are you sure?') ">
83     Food name:
84     <input id="Food name: " type="text" name="name" value="foodname"> <br></br>
85     <button>
86     <input type="hidden" name="delete" value="1">
87     Delete
88     </button>
89 </form>
90

```

```

JS main.js updatesearch.html x
64     <input id= "Units of measurement: " type= "text" name= "unit_measurement" value= "unitOfMeasurement" > <br></br>
65     Calories:
66     <input id="Calories: " type="number" step="any" name="calories" value="0"> <br></br>
67     Carbs:
68     <input id="Carbs: " type="number" step="any" name="carbs" value="0"> <br></br>
69     Fat:
70     <input id="Fat: " type="number" step="any" name="fat" value="0"> <br></br>
71     Protein:
72     <input id="Protein: " type="number" step="any" name="protein" value="0"> <br></br>
73     Salt:
74     <input id="Salt: " type="number" step="any" name="salt" value="0"> <br></br>
75     Sugar:
76     <input id="Sugar: " type="number" step="any" name="sugar" value="0"> <br></br>
77     <input type="submit" value="Submit">
78 </form>
79
80 <br></br>
81 <p>Please key in the desired food to be deleted from database</p>
82 <form method= "POST" action= "/topic7/mid-term/deletefood" onsubmit="return confirm('Are you sure?') ">
83     Food name:
84     <input id="Food name: " type="text" name="name" value="foodname"> <br></br>
85     <button>
86     <input type="hidden" name="delete" value="">
87     Delete
88     </button>
89 </form>
90
91 <br>
92 <p>Other links:</p>
93 <a href="/topic7/mid-term/home">Home</a>
94 <a href="/topic7/mid-term/addfood">Add Food</a>
95 <a href="/topic7/mid-term/search">Search Food</a>
96 <a href="/topic7/mid-term/list">View all the food</a>
97
98 </body>
99 </html>

```

R6: List Food Page

In this requirement, we need to display all the food stored in the database. Database should also be in tabular format.

To display data in tabular format, need to include `<table> </table>`.

Then need to separate into rows `<tr> </tr>` and headers `<th></th>`

```
JS main.js  <> list.html  x
1  <!doctype html>
2  <html>
3  <head>
4  <style>
5    table, th, td {
6      border: 1px solid black;
7      border-collapse: collapse;
8    }
9    th, td {
10   padding: 5px;
11   }
12 </style>
13
14 <title>Food List Page!</title>
15 </head>
16 <body>
17 <h1> This is Food List Page </h1>
18 <h3>You can see names and nutritional facts of all available foods stored here:</h3>
19
20 <table style="width:70%">
21
22   <tr>
23     <th>Select</th>
24     <th>Quantity</th>
25     <th>Food Name</th>
26     <th>Typical Values Per</th>
27     <th>Unit of Typical Values </th>
28     <th>Calories</th>
29     <th>Carbs</th>
30     <th>Fat</th>
31     <th>Protein</th>
32     <th>Salt</th>
33     <th>Sugar</th>
34   </tr>
35
36   <%
37     availableFood.forEach(function(food){
38   %>
```

SQL Table

```
Database changed
mysql> describe food;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
name	varchar(50)	YES		NULL	
typical_values_per	decimal(15,2) unsigned	YES		NULL	
unit_measurement	varchar(50)	YES		NULL	
calories	decimal(10,2) unsigned	YES		NULL	
carbs	decimal(8,2) unsigned	YES		NULL	
fat	decimal(7,2) unsigned	YES		NULL	
protein	decimal(7,2) unsigned	YES		NULL	
salt	decimal(7,2) unsigned	YES		NULL	
sugar	decimal(7,2) unsigned	YES		NULL	

```
10 rows in set (0.00 sec)
```

```
mysql>
```

The purpose of this table is to store each and respective field into the database. Each field has different types. For instance, for name, the type would be varchar(50).

For each data type,

id: this will increase by one every time a food is stored.

name: we use varchar for this case as it is indeterminate length string data type. It can hold numbers, letters and special characters.

typical_values_per: we use decimal for this case as we will be storing numbers. decimal (15,2) means that it can store up to 15 digits, and up to 2 decimal points. Unsigned is included so it will not accept negative values.