IS216 Sample Mini Lab Test

General Instructions:

- This is an open-book, take-home and **individual** test.
- You must test your web pages using **Google Chrome Web Browser** (Version 85.0.4183 or later). Your graders will be using only **Google Chrome Web Browser** (Version 85.0.4183 or later) to test your web pages.
- No questions will be entertained by the IS216 teaching team (faculty/instructor/Teaching Assistants) during the test period. If necessary, make your own assumptions.
- You are allowed to use only standard HTML5, CSS, Bootstrap and JavaScript (do NOT use jQuery) in your solutions. Do not use any other third party libraries.
- Use meaningful names for HTML class/id and JavaScript variables and functions. You must indent your code (HTML/CSS/JavaScript) correctly. Use 4 spaces for indentation. Failure to do so will attract a penalty of up to **20%** of your score for the corresponding question.
- You **MUST** include your name as author in the comments of all your submitted source files. Failure to do so will attract a penalty of up to **20%** of your score for the corresponding question.
  For example, if your registered name is "KIM Jong Un" and email ID is kim.jongun.2019, include the following comment at the beginning of each source file you write.

HTML files	CSS, JavaScript files

`<!--`
`  Name:  KIM Jong Un`
`  Email: kim.jongun.2019`
`-->`
	`/*`
	`  Name:  KIM Jong Un`
	`  Email: kim.jongun.2019`
	`*/`

- You may wish to comment out the parts in your code which cause errors. But commented code will not be marked.
- All student submissions will be thoroughly checked by an SMU-approved source code plagiarism checker software program AND an additional external software program. The source code checking will be conducted across all submissions from all 9 sections of IS216. Suspected plagiarism cases will be communicated immediately to the IS216 faculty in charge and SIS Dean's Office for further investigation. Students in the suspected cases will be informed accordingly by their section faculty, and the incident will be escalated to the SMU University Council of Student Conduct. More information about the SMU Student Code of Conduct can be found **HERE**.

Question 1: Savings (Difficulty Level: */**) [8 marks]

Given:

```
|--- q1
      |-- savings.html
      |-- savings.js
```

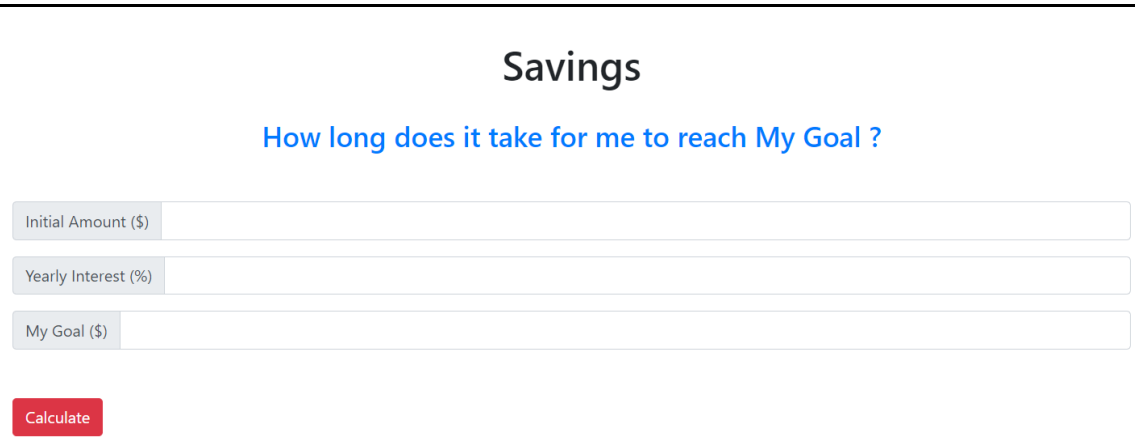
IMPORTANT
You are free to define the HTML elements and CSS style required to match the screenshot to the best of your interpretation ***where the requirement is not stated explicitly***. This includes margin, padding, font style, font size, etc. **`savings.html`** must only contain HTML and CSS code. You must write **ALL** JavaScript code inside **`savings.js`**. JavaScript code written inside **`savings.html`** will ***NOT*** be considered for grading.

Part A: Complete `savings.html` (3 marks)

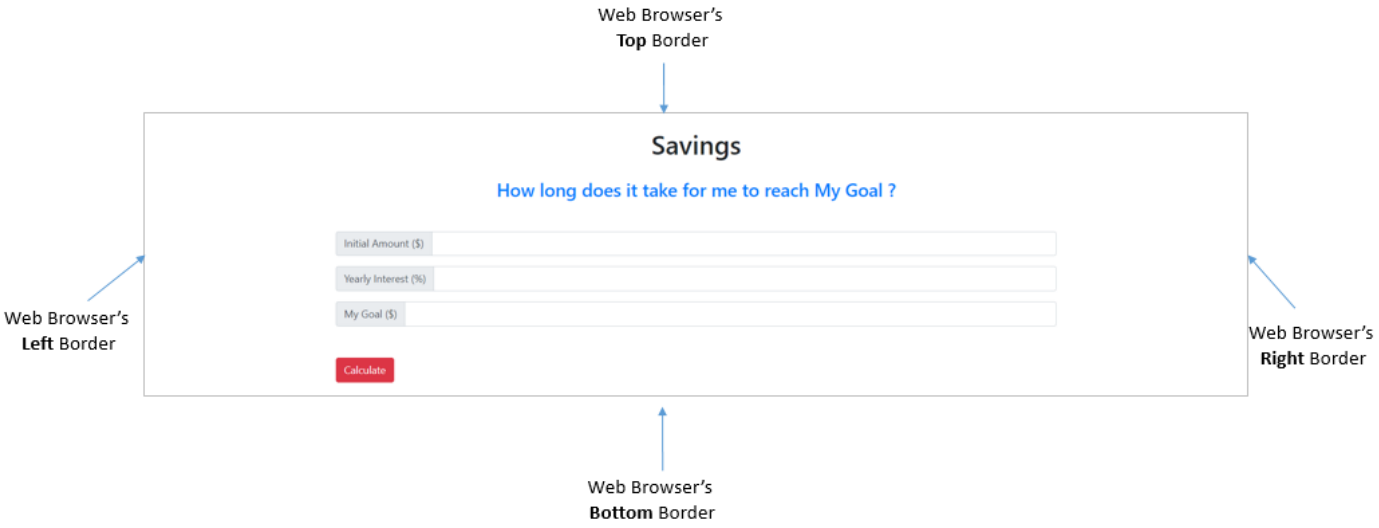
Complete implementation of **`savings.html`** such that when rendered in a web browser, **`savings.html`** displays the following.

NOTE: The three **input fields** must only allow for **numeric values** (not text). You may **ASSUME** that the user will always enter numeric values greater than 0 (thus, no negative values).

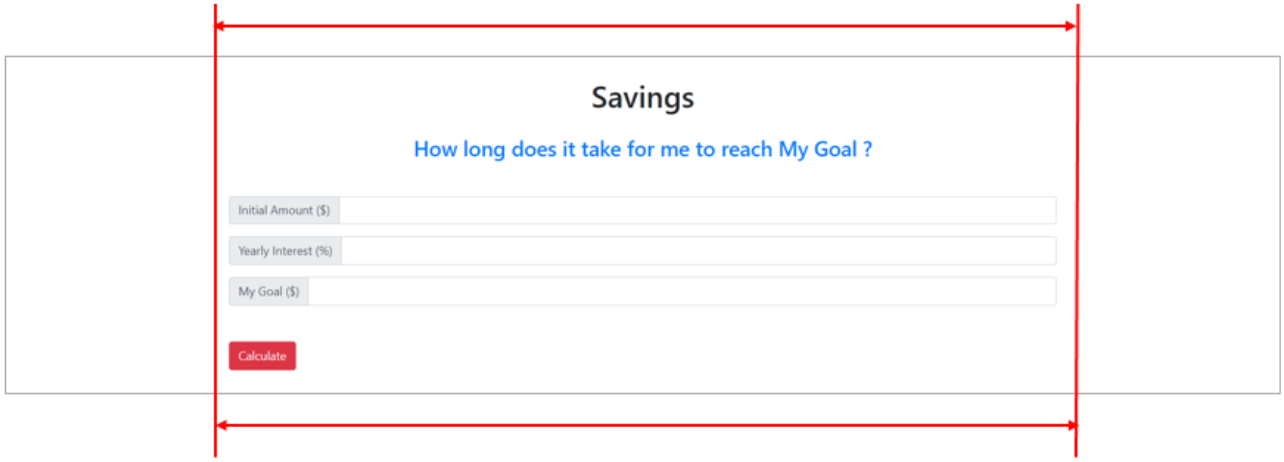
`savings.html`



Please take note of the following **user interface layout** guidelines:



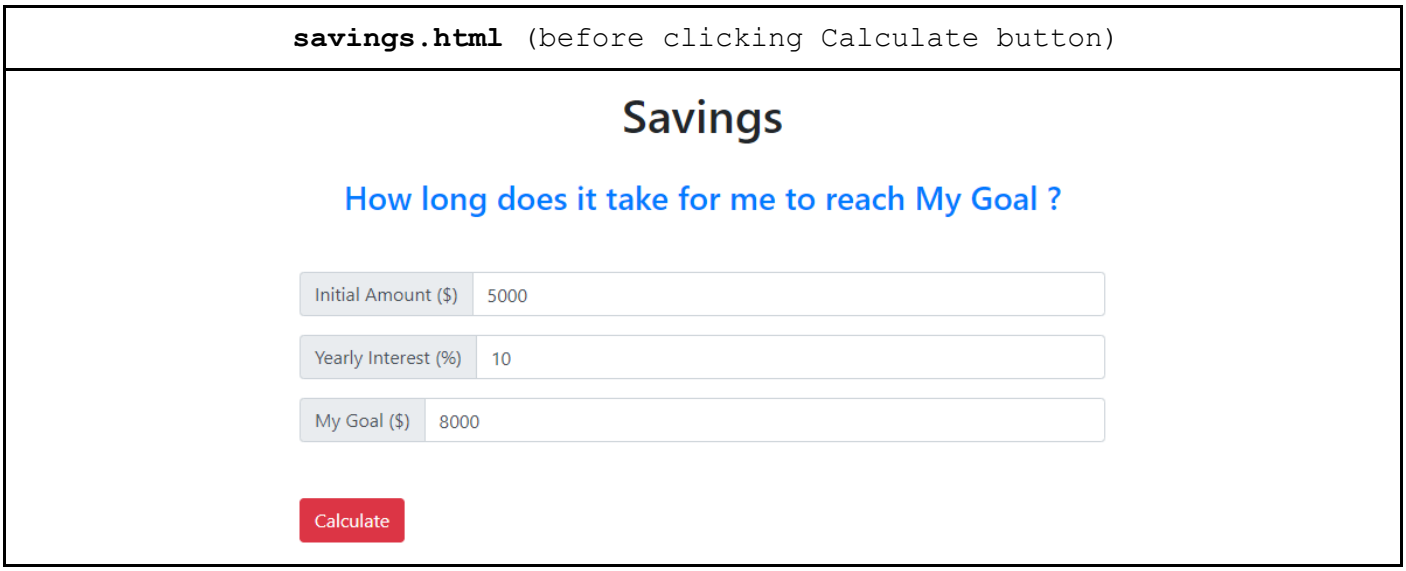
The **main content** is centered and spans **70%** of the web browser **width.**



Part B: Complete function `calculate()` (5 marks)

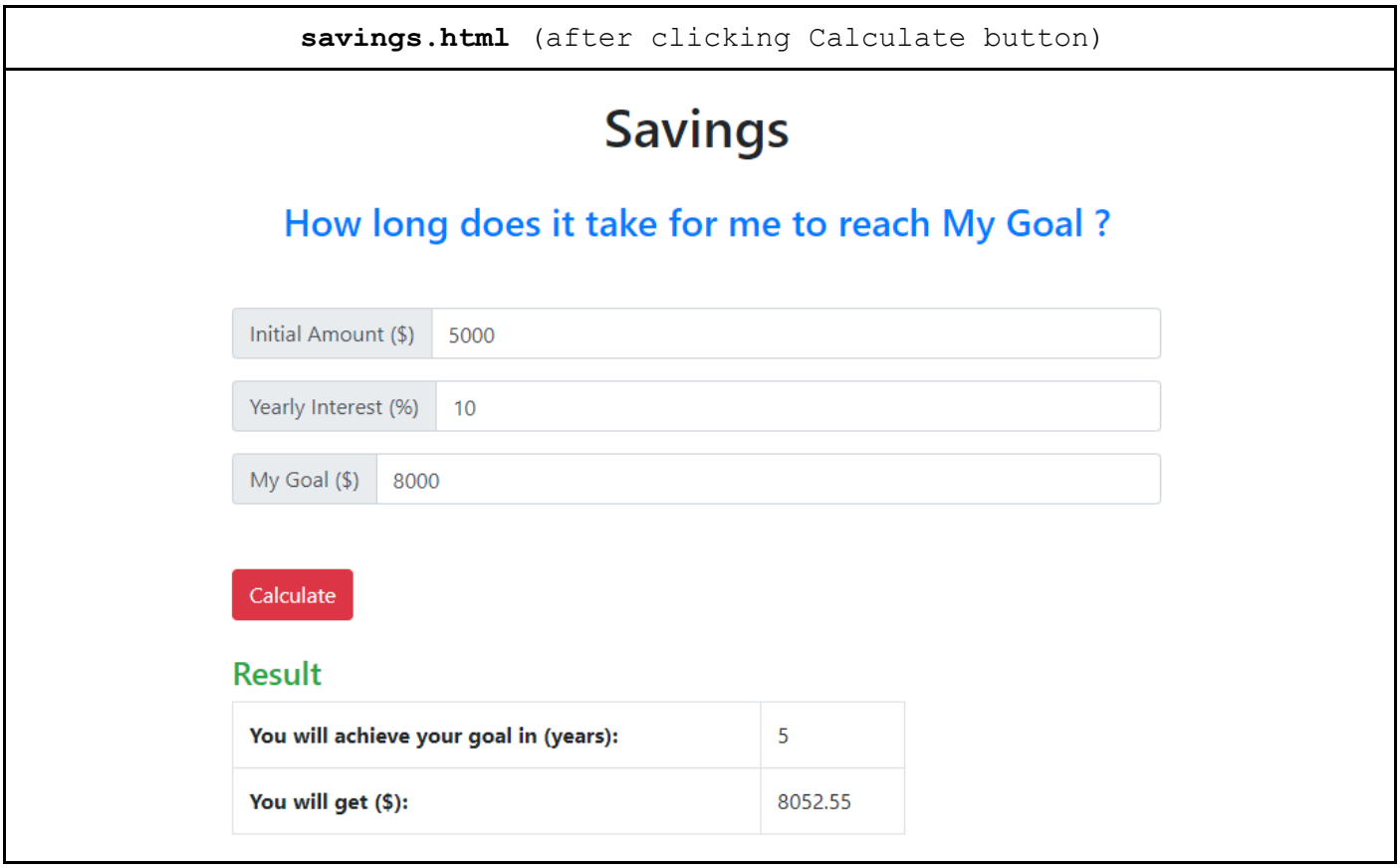
Complete the function **`calculate()`** in **`savings.js`** such that it calculates the **number of years** it will take for the user to achieve **My Goal** (user's goal) given 1) an **Initial Amount ($)** and 2) **Yearly interest (%)**.

`savings.html` (before clicking Calculate button)



1. The user enters Initial Amount = **5000**, Yearly Interest = **10** and a Goal of **8000** in the three input text fields.
2. The user clicks the **Calculate** button.
3. The **results** will be displayed below the **Calculate** button. It will display the **number of years** to achieve the goal (***as an Integer***) and the **amount** you will get in **two decimal places**.

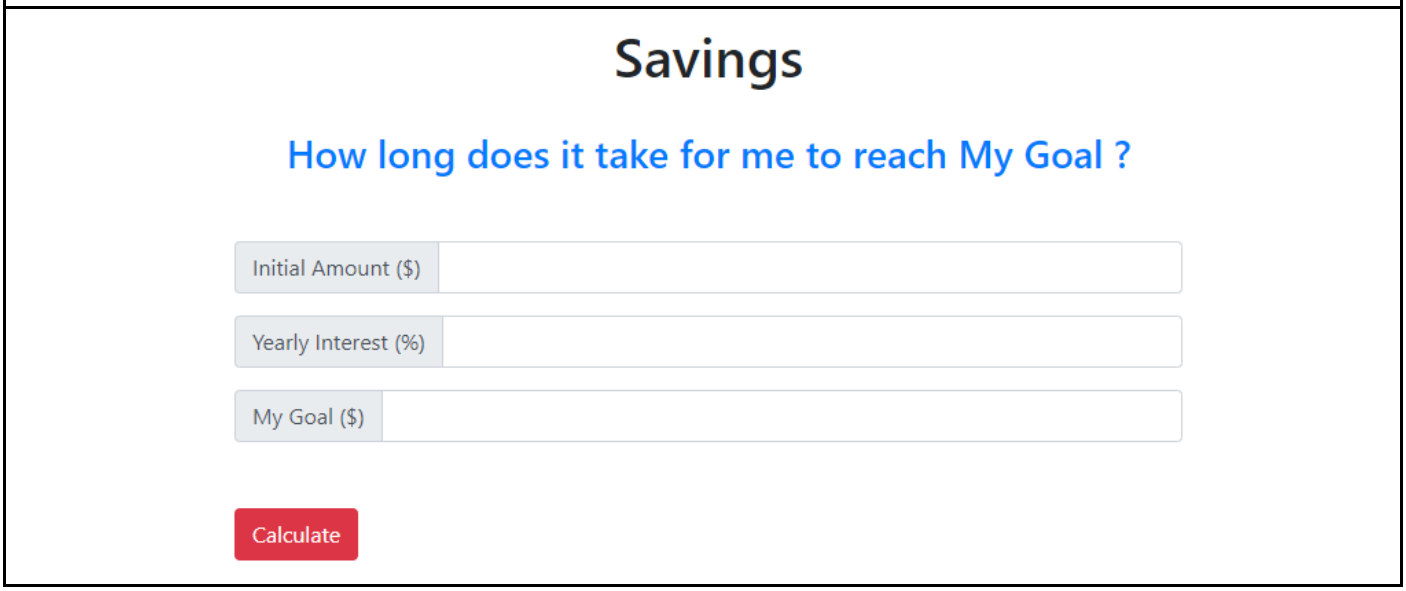
`savings.html` (after clicking Calculate button)



For the **"My Goal" calculation** to work, all three **input field values** are necessary.

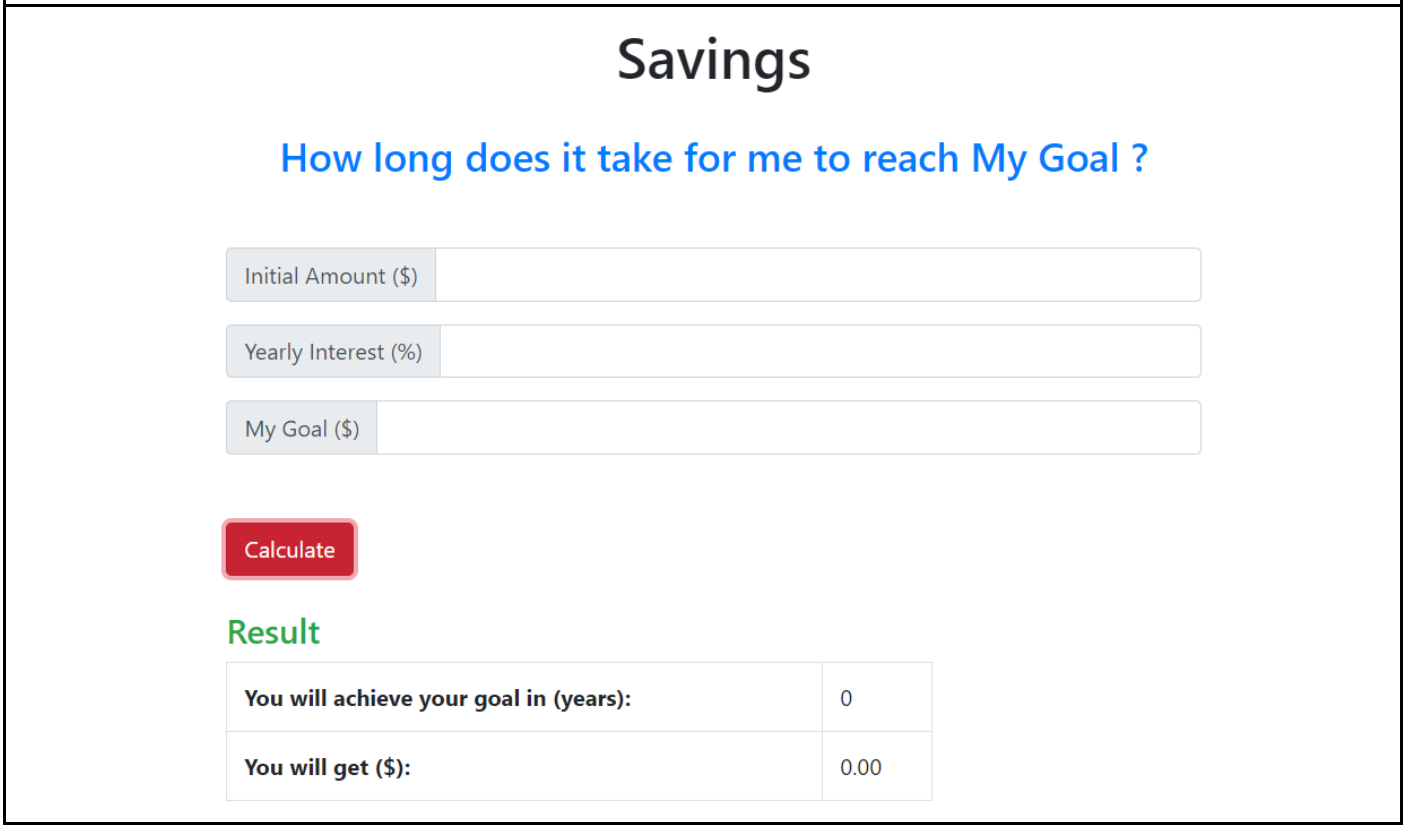
For example, below, we demonstrate a scenario where the user leaves all three input fields EMPTY.

savings.html (before clicking Calculate button)



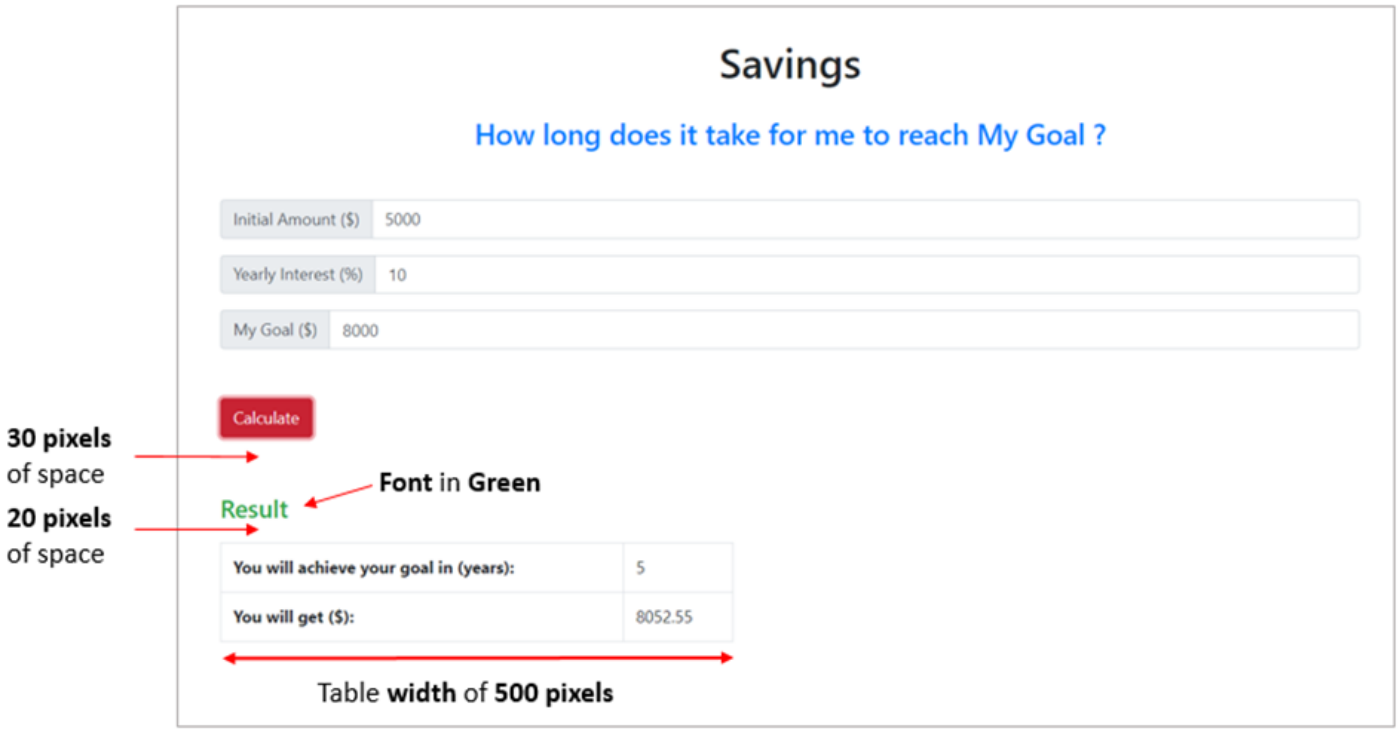
1. The user leaves all three **input fields** empty (no values).
2. The user clicks the **Calculate** button.

savings.html (after clicking Calculate button)



3. The **results** will be displayed below the **Calculate** button (indicating zero).

Please take note of the following **user interface layout** guidelines:



You may refer to the following **test cases** to test your code.

	User Input	Result after clicking Calculate button
1	Initial Amount: **1000** Yearly Interest: **10** My Goal: **1100**	Result You will achieve your goal in (years): 1 You will get ($): 1100.00
2	Initial Amount: **1500** Yearly Interest: **5.2** My Goal: **2000**	Result You will achieve your goal in (years): 6 You will get ($): 2033.23
3	Initial Amount: **1000.50** Yearly Interest: **7.8** My Goal: **800**	Result You will achieve your goal in (years): 0 You will get ($): 1000.50
4	Leaving **any** of the three input fields **EMPTY**	Result You will achieve your goal in (years): - You will get ($): 0.00

Question 2: Grocery [11 marks]

Given:

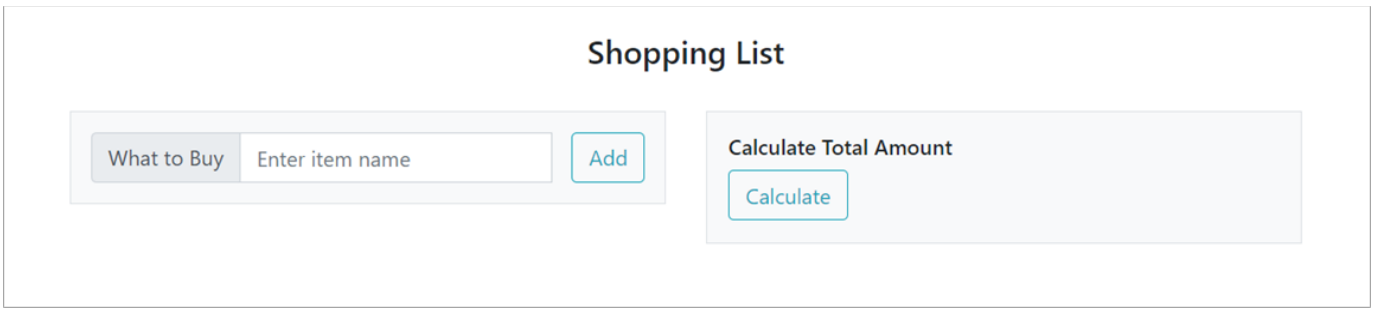
```
|--- q2
      |-- grocery.html
      |-- grocery.js
```

IMPORTANT
You are free to define the HTML elements and CSS style required to match the screenshot to the best of your interpretation ***where the requirement is not stated explicitly***. This includes margin, padding, font style, font size, etc. **`grocery.html`** must only contain HTML and CSS code. You must write **ALL** JavaScript code inside **`grocery.js`**. JavaScript code written inside **`grocery.html`** will ***NOT*** be considered for grading.

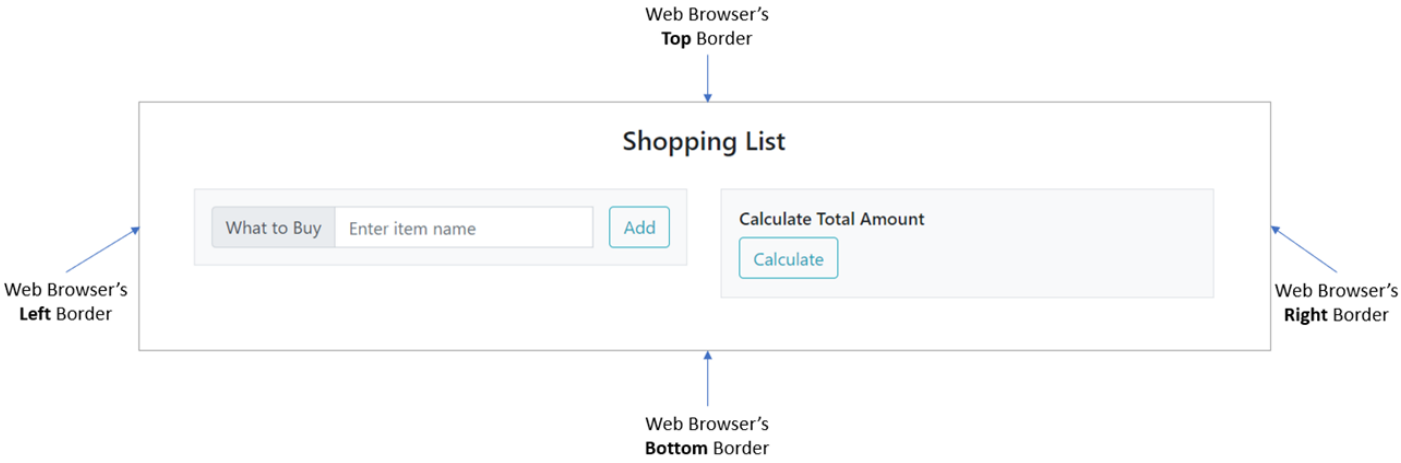
Part A: Complete `grocery.html` (3 marks)

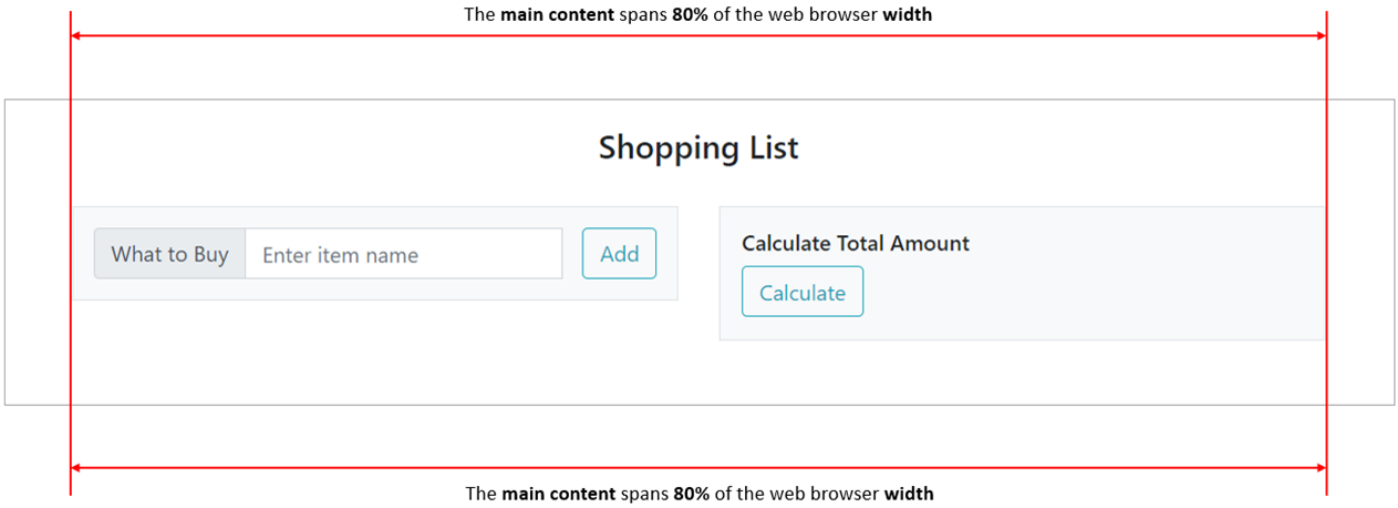
Complete the implementation of **`grocery.html`** such that when rendered in a web browser, **`grocery.html`** displays the following:

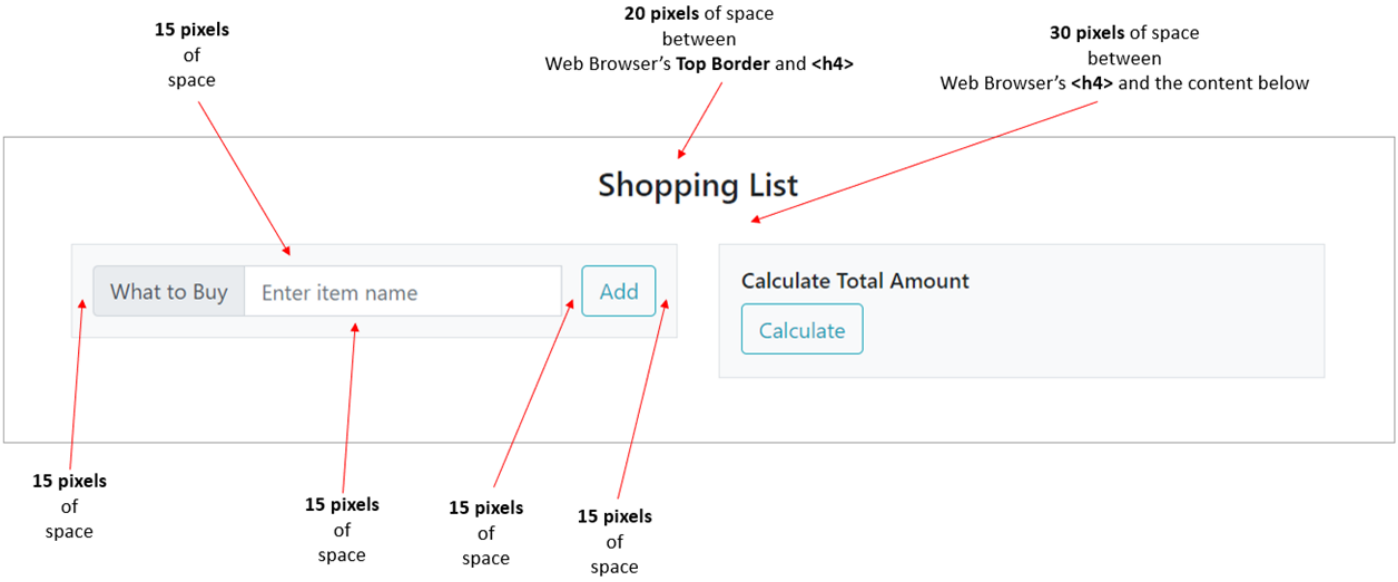
`grocery.html`

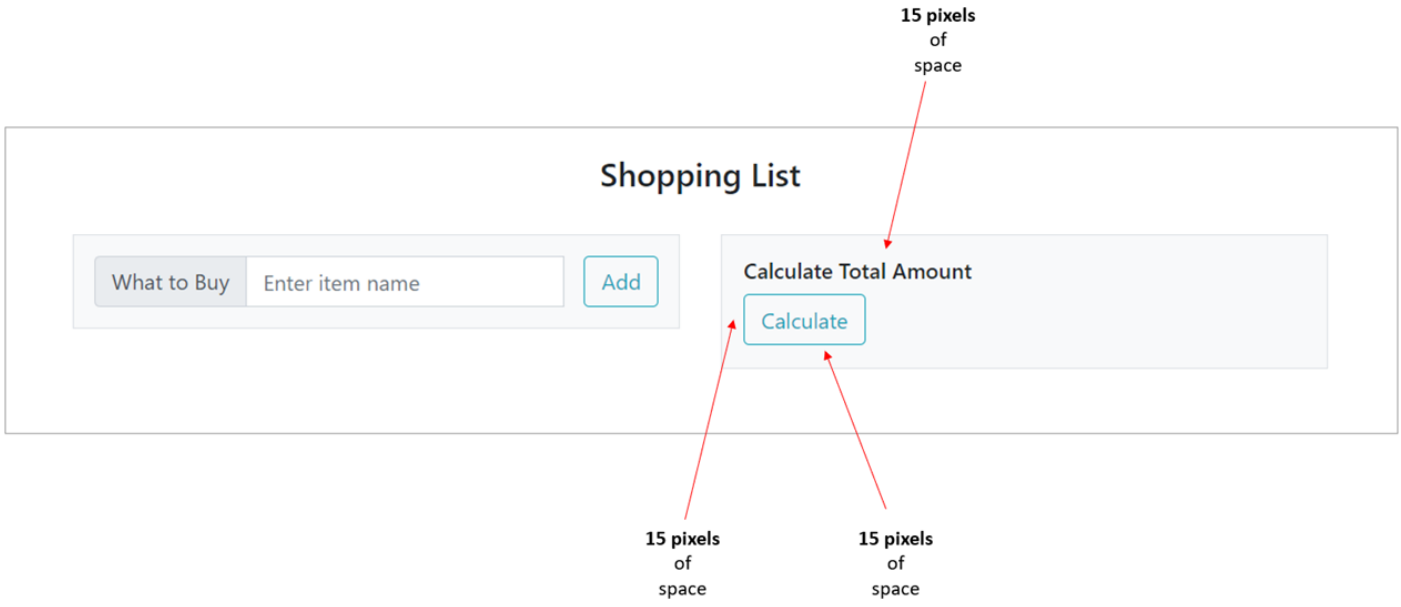


Please take note of the following **user interface layout** guidelines:




The **main content** spans **80%** of the web browser **width**
Shopping List
What to Buy
Enter item name
Add
Calculate Total Amount
Calculate
The **main content** spans **80%** of the web browser **width**

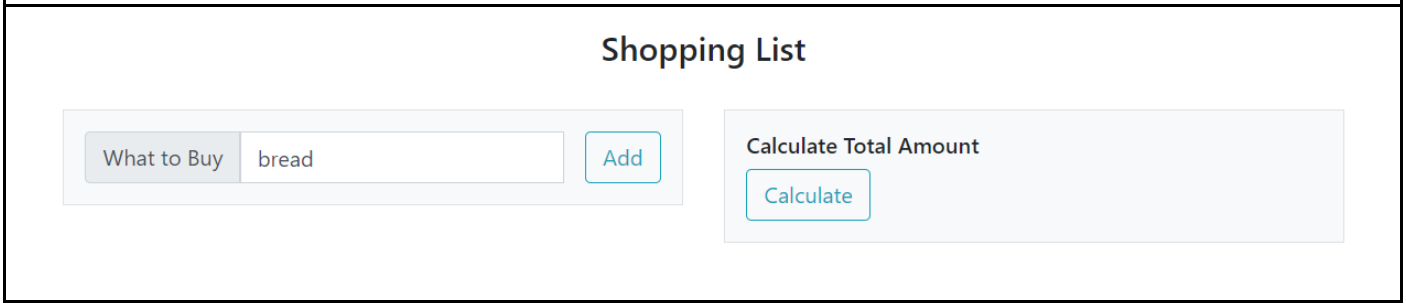

15 pixels of space
20 pixels of space between Web Browser's **Top Border** and **<h4>**
30 pixels of space between Web Browser's **<h4>** and the content below
Shopping List
What to Buy
Enter item name
Add
Calculate Total Amount
Calculate
15 pixels of space
15 pixels of space
15 pixels of space
15 pixels of space


15 pixels of space
Shopping List
What to Buy
Enter item name
Add
Calculate Total Amount
Calculate
15 pixels of space
15 pixels of space

Part B: Complete `grocery.js` (4 marks)

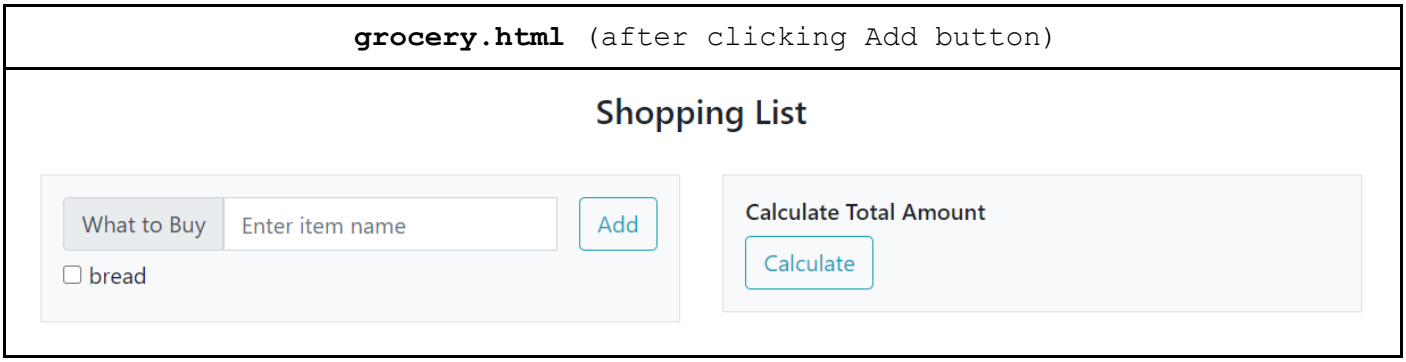
Implement **`addItem()`** and **`processItems()`** such that the following **user scenario** is fulfilled.

`grocery.html` (before clicking Add button)



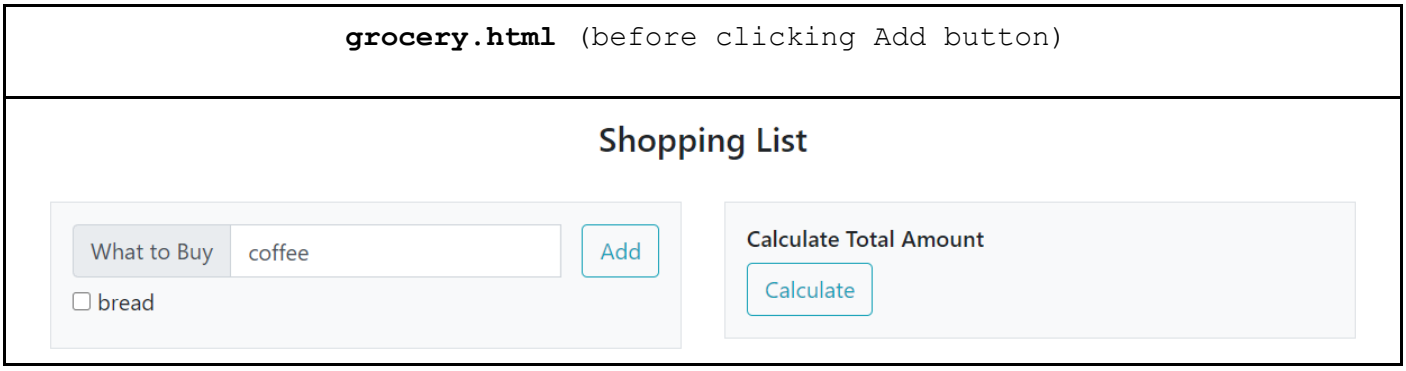
1. The user keys in string/text **bread** in the input text field.
2. The user **clicks** the **Add** button.

`grocery.html` (after clicking Add button)



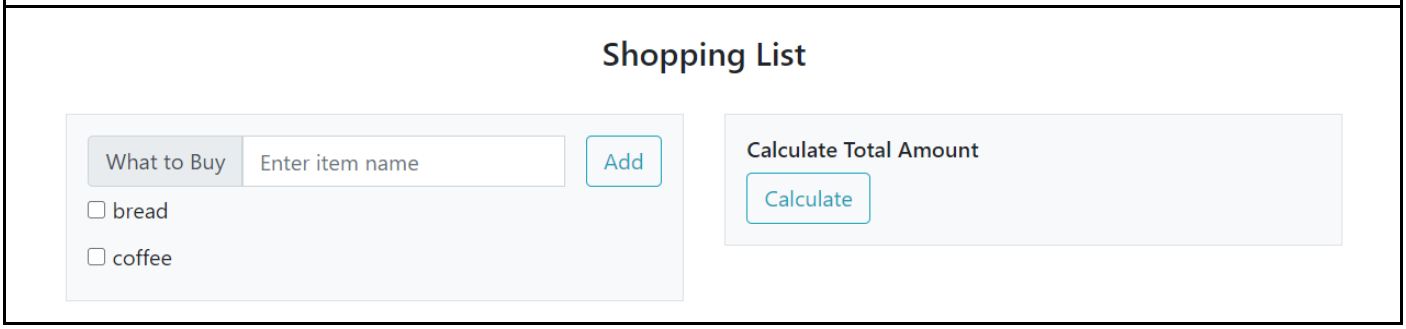
Item **bread** has been **added** with a **checkbox**.
Next, the user will add **one more item**.

`grocery.html` (before clicking Add button)



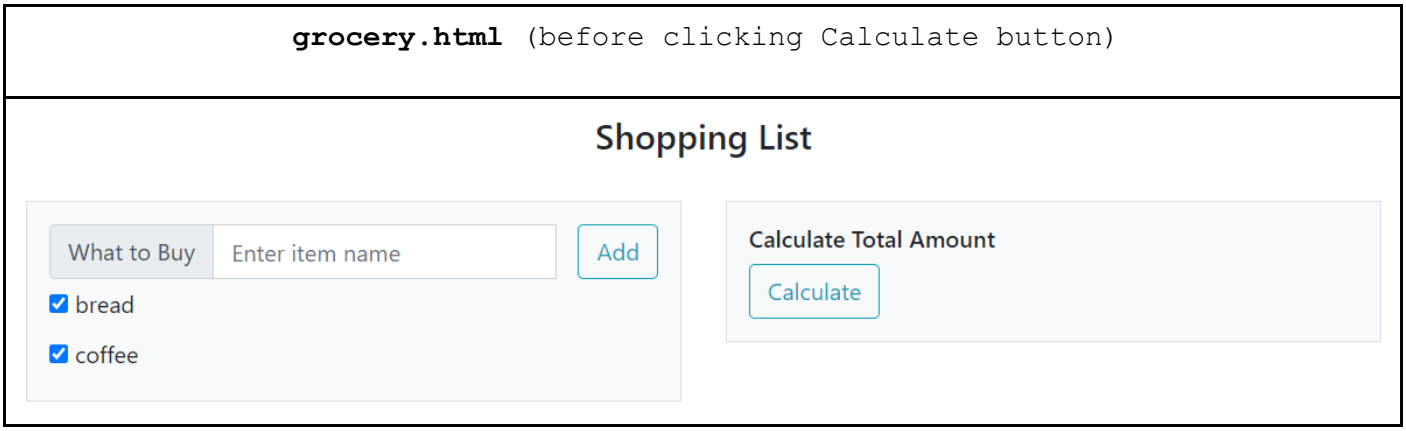
3. The user keys in string/text **coffee** in the input text field.
4. The user **clicks** the **Add** button.

`grocery.html` (after clicking Add button)



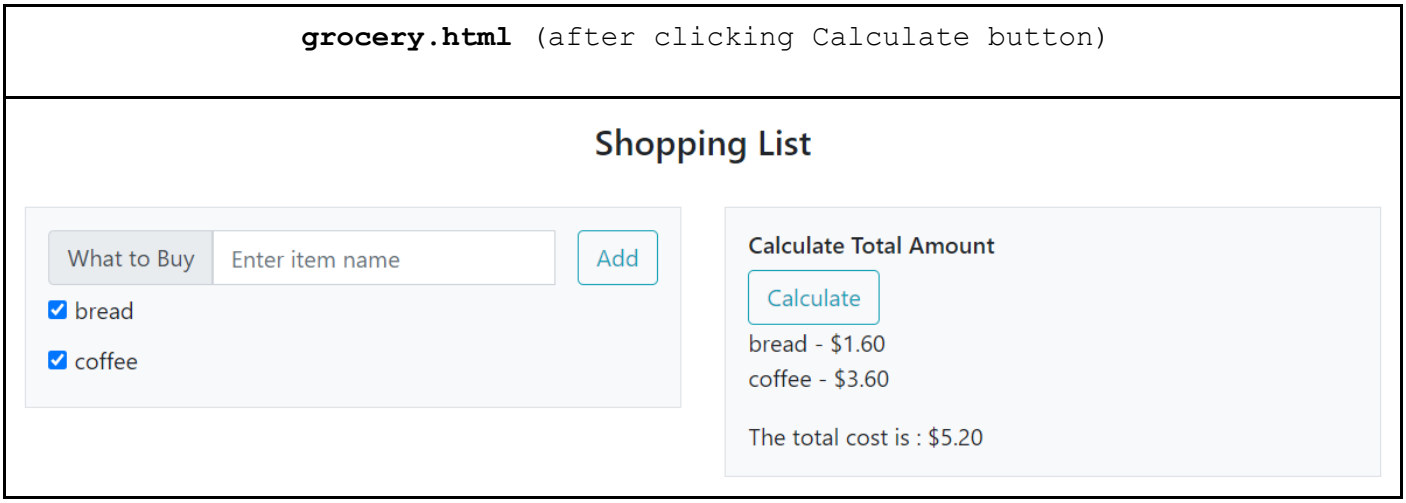
Item **coffee** has been **added** with a **checkbox**.

`grocery.html` (before clicking Calculate button)



5. The user selects both **bread checkbox** and **coffee checkbox**.
6. The user **clicks** the **Calculate** button.

`grocery.html` (after clicking Calculate button)



Below the **Calculate** button, each item's name and its price are listed.

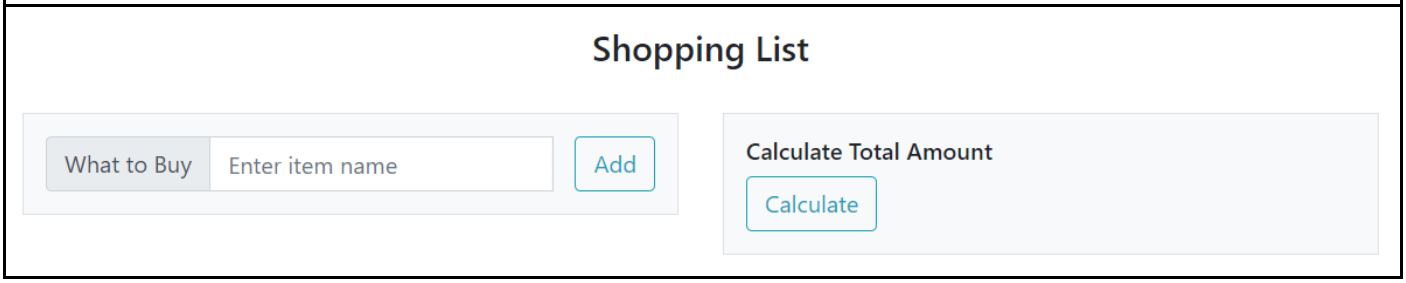
- Each item's **price** information can be found in the **`shopList`** array in **`grocery.js`**.

Furthermore, the **total cost** is calculated and displayed below the item list.

Part C: Complete `grocery.js` (4 marks)

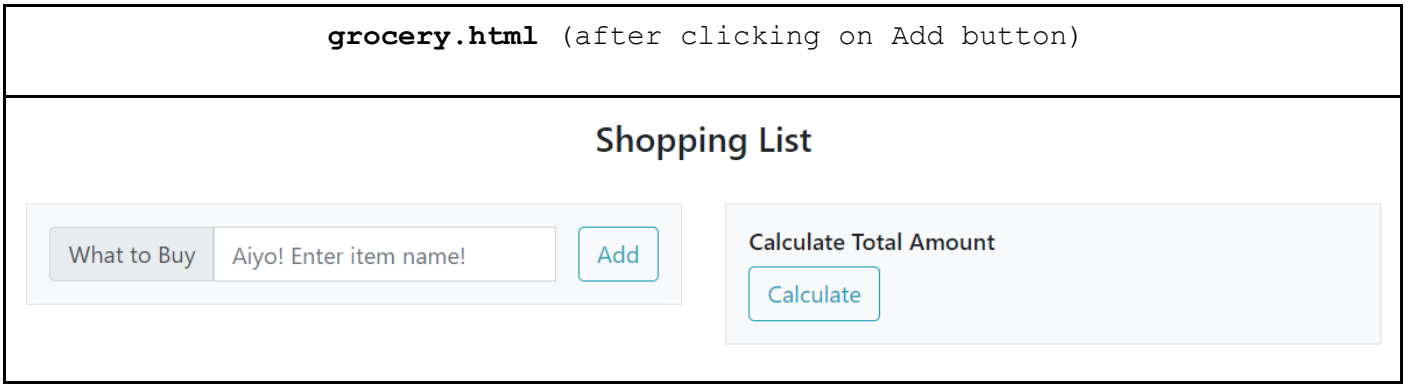
Modify **`addItem()`** and **`processItems()`** such that the following **user scenario** is fulfilled.

`grocery.html` (before clicking Add button)



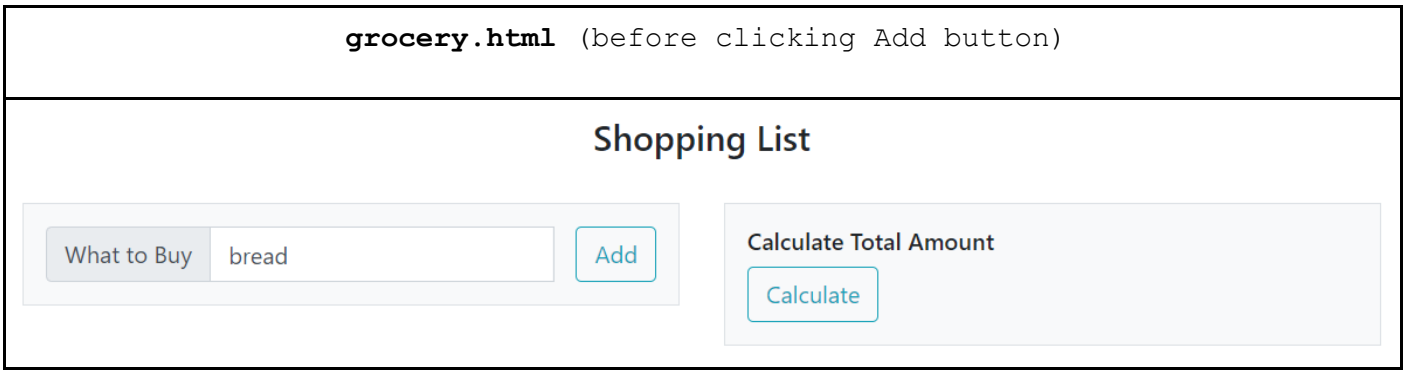
1. The user does NOT key in anything in the input text field (leaving it empty).
2. The user **clicks** the **Add** button.

`grocery.html` (after clicking on Add button)



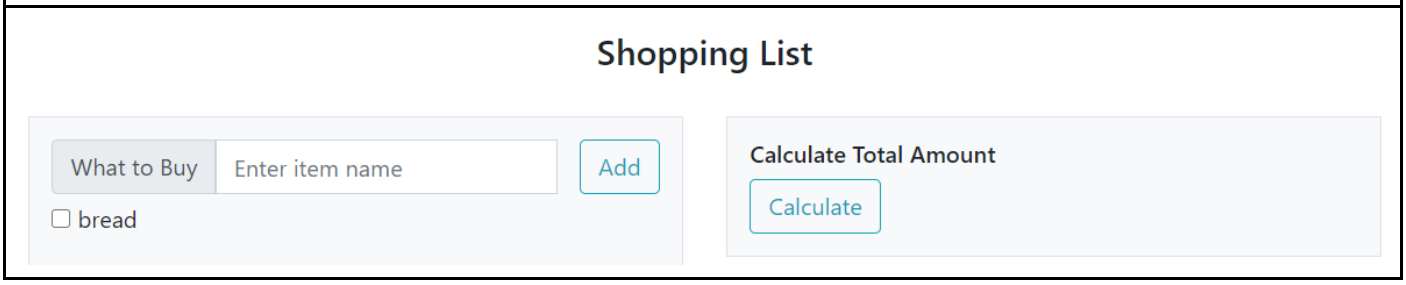
The input text field's **placeholder** value has been updated to **Aiyo! Enter item name!**
Next, the user will add **one item**.

`grocery.html` (before clicking Add button)



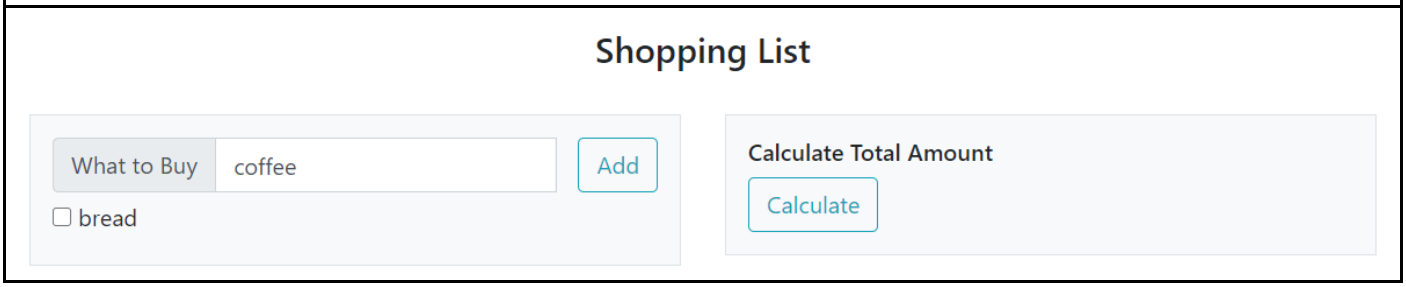
3. The user keys in string/text **bread** in the input text field.
4. The user **clicks** the **Add** button.

grocery.html (after clicking Add button)



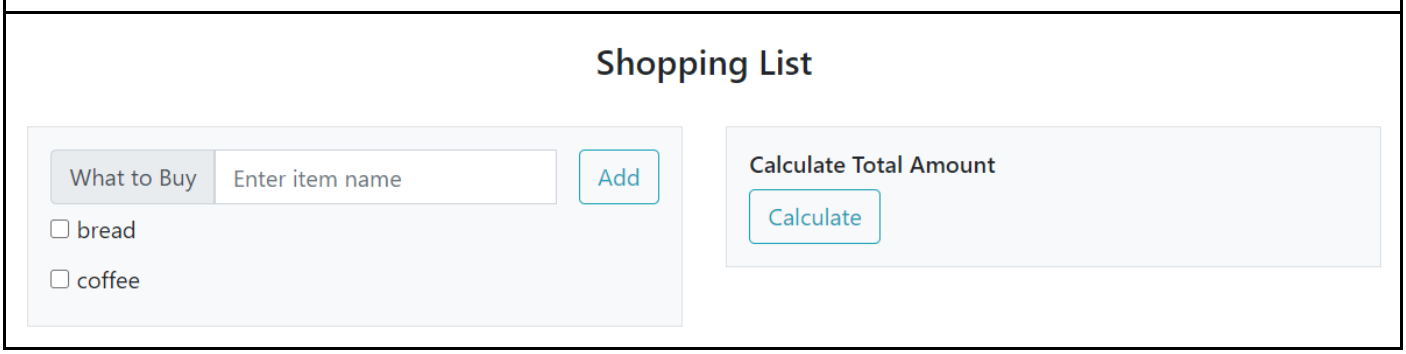
Item **bread** has been **added** with a **checkbox** button.
Next, the user will add **one more item**.

grocery.html (before clicking Add button)



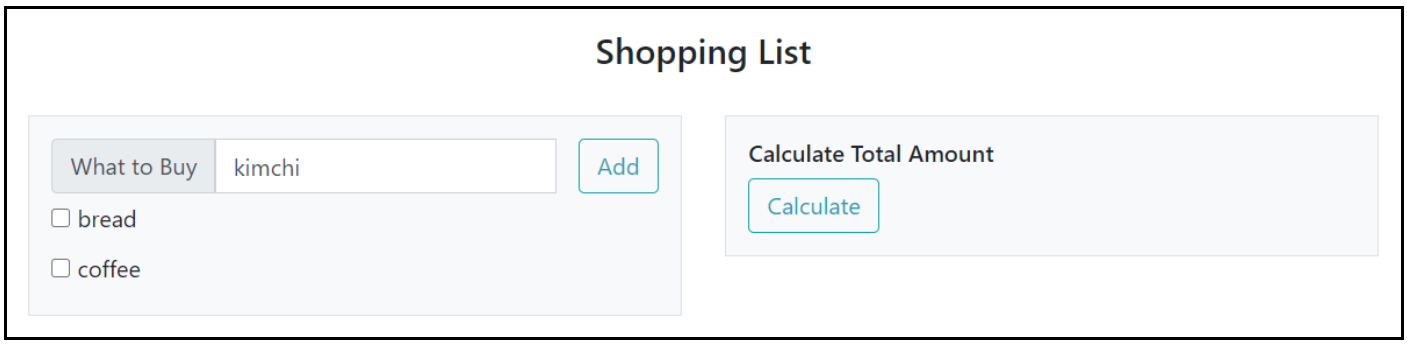
5. The user keys in string/text **coffee** in the input text field.
6. The user **clicks** the **Add** button.

grocery.html (after clicking Add button)



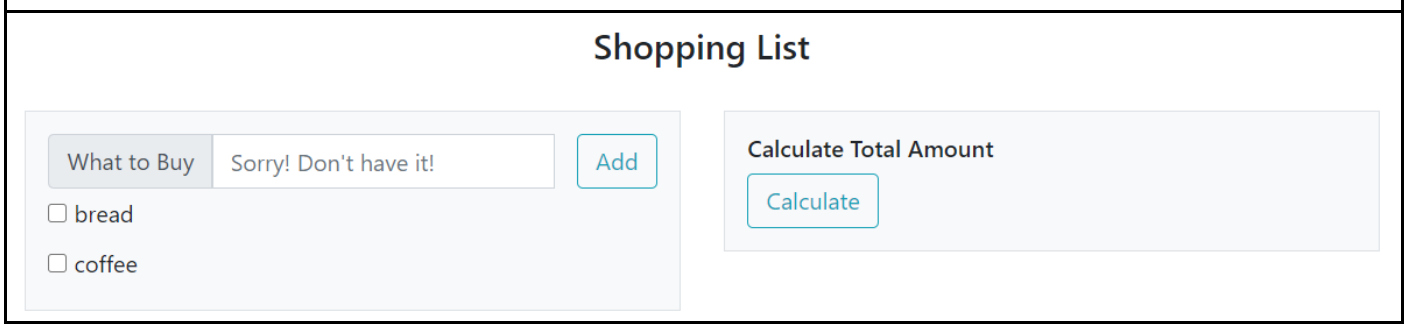
Item **coffee** has been **added** with a **checkbox** button.

grocery.html (before clicking Add button)



7. The user keys in string/text **kimchi** in the input text field.
8. The user **clicks** the **Add** button.

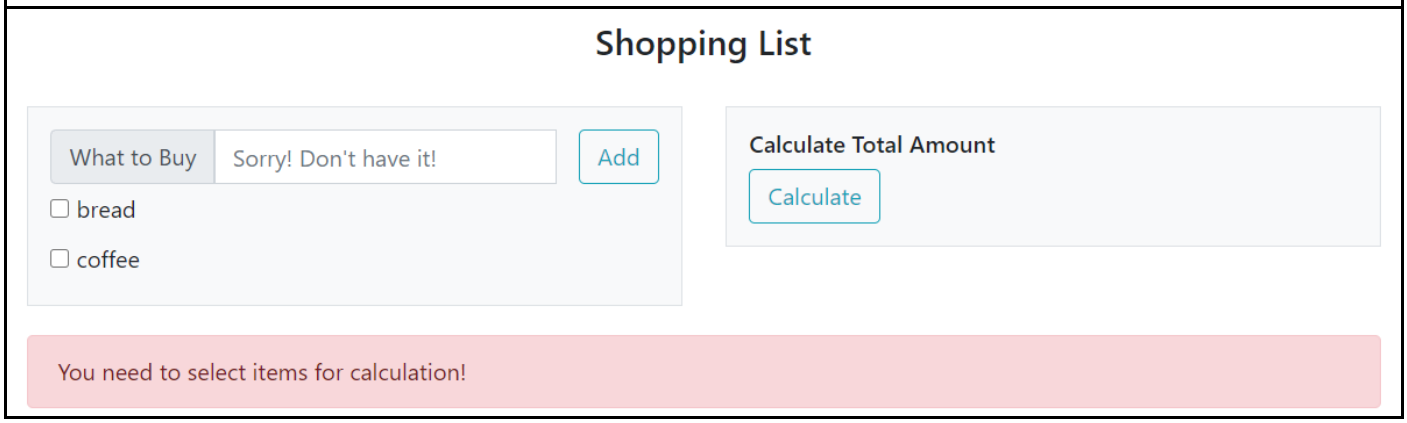
`grocery.html` (after clicking Add button)



Item **kimchi** is ***NOT*** available in this shop (refer to the variable **`shopList`** in **`grocery.js`** file).
The input text field's **placeholder** value has been updated to **Sorry! Don't have it!**

9. The user does NOT select any item **checkboxes**.
10. The user **clicks** the **Calculate** button.

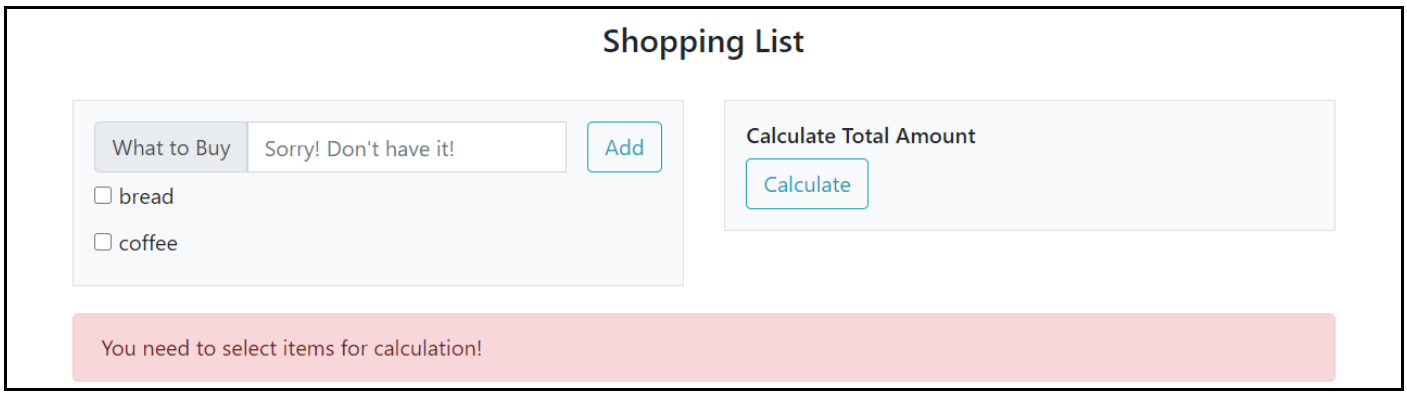
`grocery.html` (after clicking Calculate button)



At the bottom of the webpage, an **alert** message is displayed with text **You need to select items for calculation!**

11. Again, the user does NOT select any item **checkboxes**.
12. The user **clicks** the **Calculate** button.

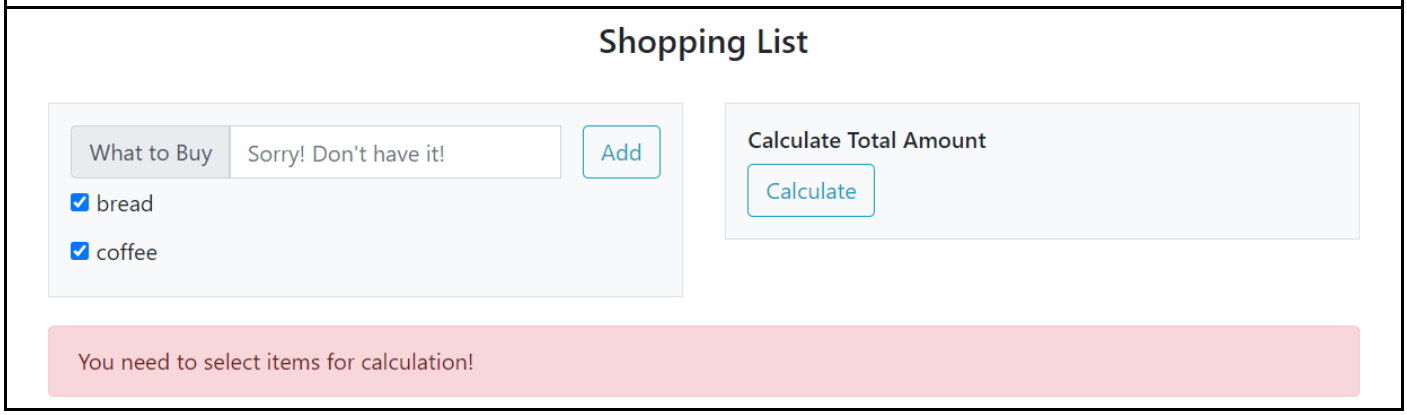
`grocery.html` (after clicking Calculate button for the 2nd time)



At the bottom of the webpage, an **alert** message is displayed with text **You need to select items for calculation!**

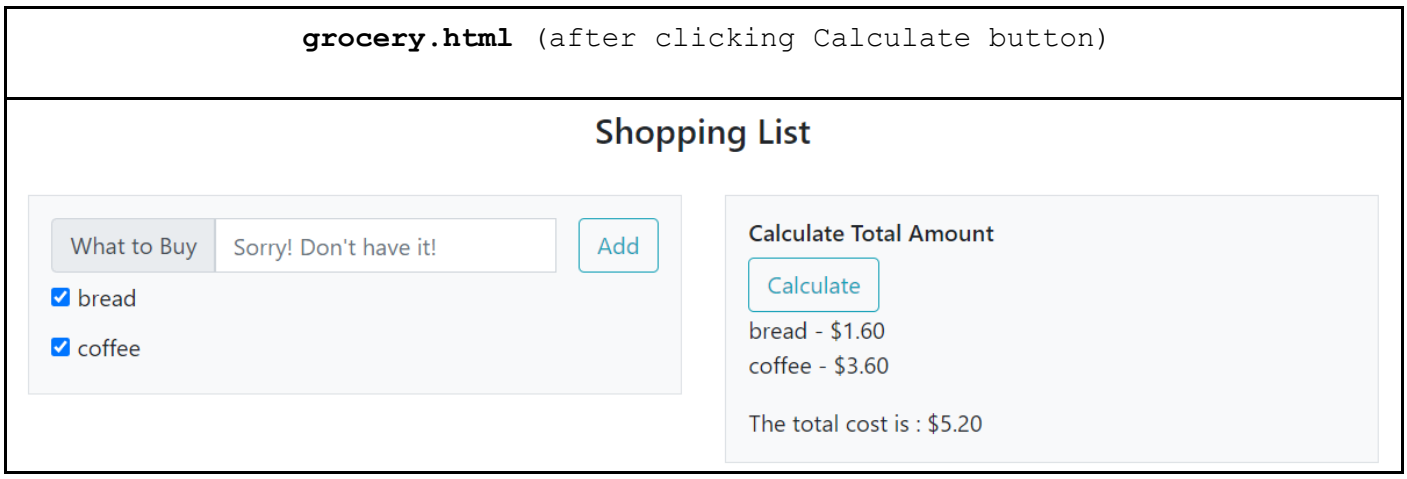
- If the user repeatedly **clicks** on the **Calculate** button without any items selected, **`grocery.html`** must display only **ONE (1) alert notification** at all times.

`grocery.html` (before clicking Calculate button)



13. Finally, the user selects both **bread checkbox** and **coffee checkbox**.
14. The user **clicks** the **Calculate** button.

`grocery.html` (after clicking Calculate button)



Below the **Calculate** button, each item and its price are listed. Furthermore,

- The **total cost** is calculated and displayed, and;
- The **alert notification** is removed from the web page.