Shatha Salem Alkulaib

Project: Kitchen Story

Program: Full Stack Developer .NET

Phase: two

GitHub link: https://github.com/ShathaAlk/Kitchen-Story-System

Email: shatha.alkulaib@gmail.com

The Kitchen Story system consists of three JavaScript files that contains the functions linked to the HTML files:

<u>First file: AccountFunctions.js</u>

The file contains all the functions associated with the account such as logging in, changing the password, logging out.

<u>Login function</u>

- Declaring the required variables.
- Using several conditions, the first of which is to verify the existence of the correct inputs.
- The second condition is to make sure that the browser supports the localStorage and sessionStorage.
- If the condition is proven, the system starts to check if the localStorage contains the unamekey key which responsible for storing the username.
- If it does not exist, the admin user must enter the preliminary data <u>(username = admin, password = admin)</u>

The key and values for the username and the password will be stored in both localStorage and sessionStorage. Then, directing the admin user to the admin page.

```
function login() {
    // Creating new variables that take the values from the input text
    var username = document.getElementById("username").value;
    var password = document.getElementById("password").value;
    // Creating new variables that take the values from the localStorage
    var storedUsername = localStorage.getItem('unameKey');
    var storedPassword = localStorage.getItem('PasswordKey');
    var emptyFields = document.getElementById("emptyFields");
    var wrongFields = document.getElementById("wrongFields");
  //If the username or password is empty, the validation message will be shown
    if (username == "" || password == "") {
        emptyFields.style.display = "block";
        wrongFields.style.display = "none";
    }
    else {
        //Check if the browser support localStorage and sessionStorage
        if (localStorage && sessionStorage) {
            //Check if the localStorage key exists
            if (!localStorage.getItem('unameKey')) {
                if (username == "admin" && password == "admin") {
```

```
   //Use setItem method to save the values to localStorage and sessionStorage
for the first time
                    localStorage.setItem('unameKey', username);
                    localStorage.setItem('PasswordKey', password);
                    sessionStorage.setItem('unameKey', username);
                    sessionStorage.setItem('PasswordKey', password);
                    location.href = './AdminPage.html';
                }
//If the username or password is wrong, the validation message will be shown
                else {
                    wrongFields.style.display = "block";
                    emptyFields.style.display = "none";
                }
            }
```

The other condition is the availability of the key in the localStorage, it is used if the admin user changed the password in the localStorage.

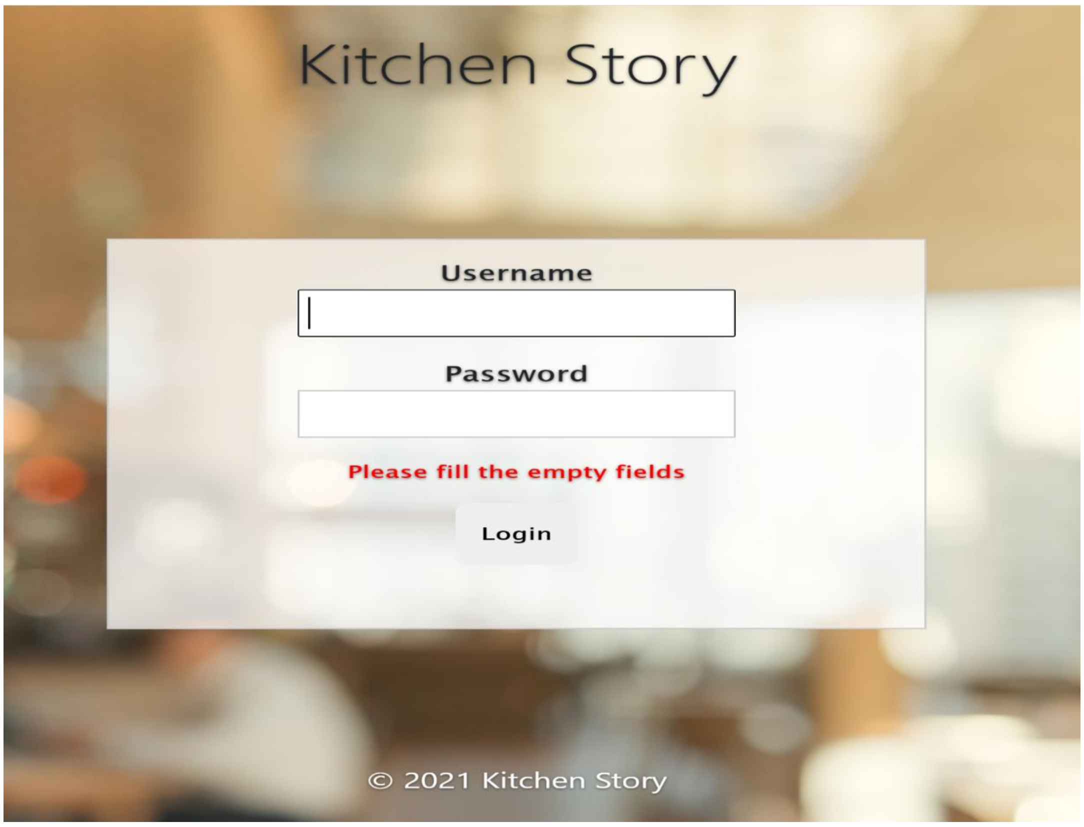
After verifying the compatibility of the entered data with the stored data, it will be stored in the sessionStorage and directed the user to the admin page.

<u>Note:</u> There is no need to store the data in the localStorage in this code because it is stored in it already after using the change password function.

```
 //If there are already stored keys and values in both storages,
            //it will use the other condition
            else {
                if (username == storedUsername && password == storedPassword)
{
 //Use setItem method to update the new values to
//localStorage and sessionStorage
                    sessionStorage.setItem('unameKey', username);
                    sessionStorage.setItem('PasswordKey', password);
                    location.href = './AdminPage.html';
                }
                else {
                    wrongFields.style.display = "block";
                    emptyFields.style.display = "none";
                }
            }
        }
        else {

            alert("localStorage is disabled");
        }
    }
}
```

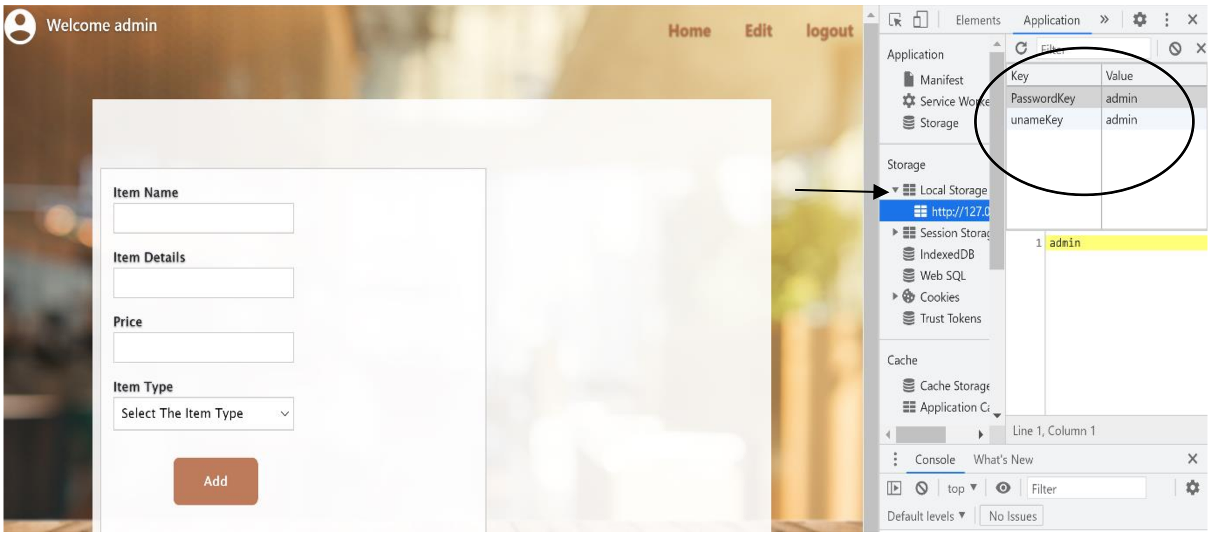
Outputs:

Case 1: If the text inputs are empty, a validation message will be shown.



Please fill the empty fields

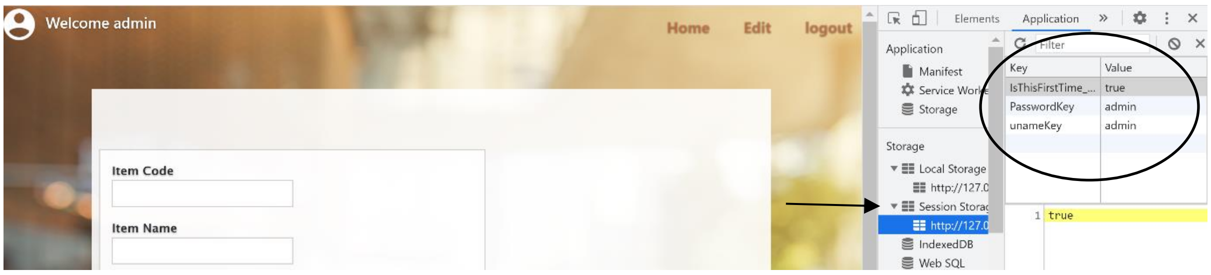
Case 2: If the text inputs are filled correctly, the admin page will be opened, and the values will be stored in both storages.



Key	Value
PasswordKey	admin
unameKey	admin

```
1 admin
```

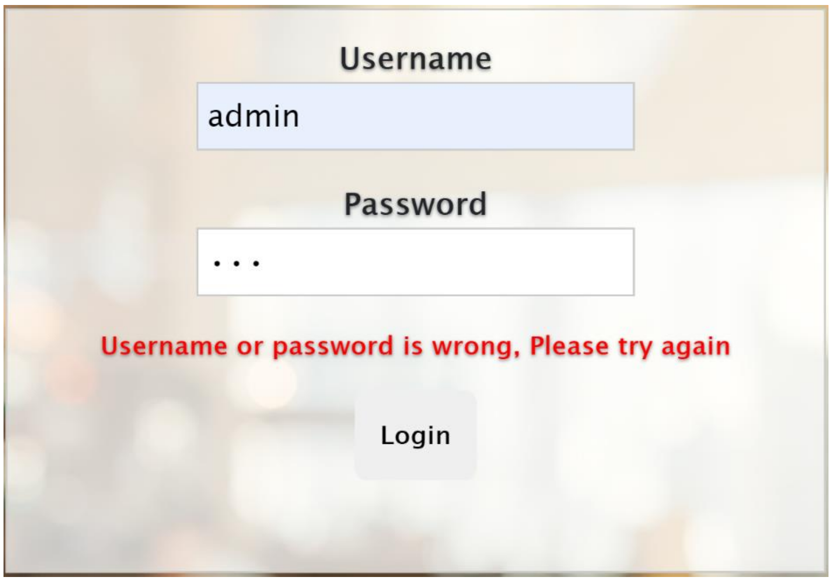
Line 1, Column 1



Key	Value
IsThisFirstTime_...	true
PasswordKey	admin
unameKey	admin

```
1 true
```

Case 3: If the text inputs are filled wrongly, a validation message will be shown.



Username: admin

Password: •••

Username or password is wrong, Please try again

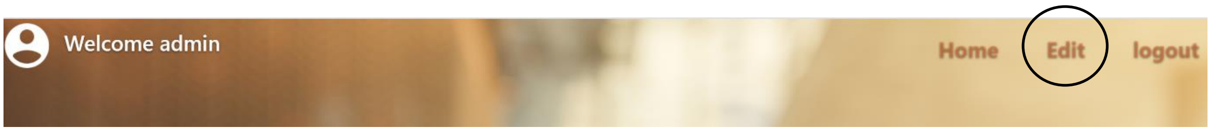
<u>changePassword function</u>

This function changes the password by obtaining the new password from the text input and then storing it in the PasswordKey (overwrite the old value).

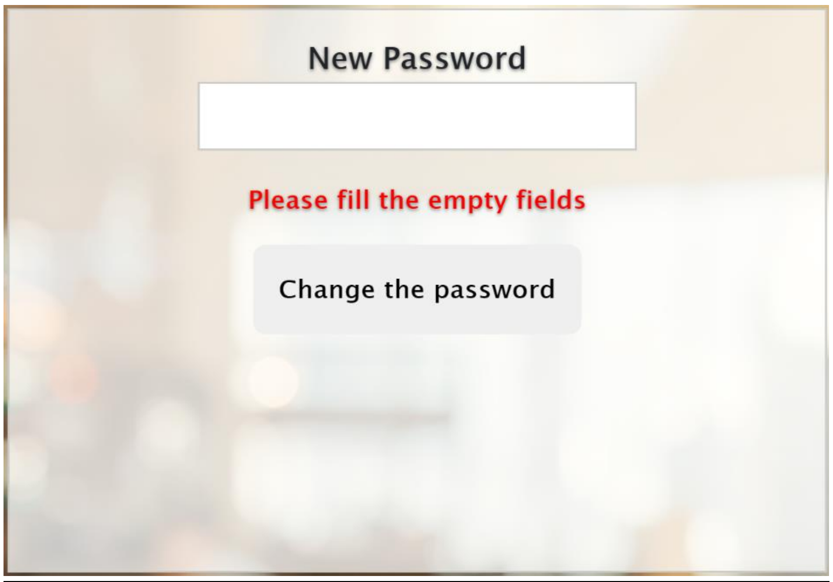
```
//Change the password by getting the new value from the input text
function changePassword() {
    var NewPassword = document.getElementById("NewPassword").value;
    var lblPassChangeMsg = document.getElementById("passChangeMsg");
    var emptyFields = document.getElementById("emptyFields");
    var AdminPageLink = document.getElementById("AdminPageLink");

    if (NewPassword == "") {
        emptyFields.style.display = "block";
    }
    else {
        localStorage.setItem('PasswordKey', NewPassword);
        lblPassChangeMsg.style.display = "block";
        AdminPageLink.style.display = "block";
        emptyFields.style.display = "none";
    }
}
```

Outputs:

Click Edit to open the change password page.

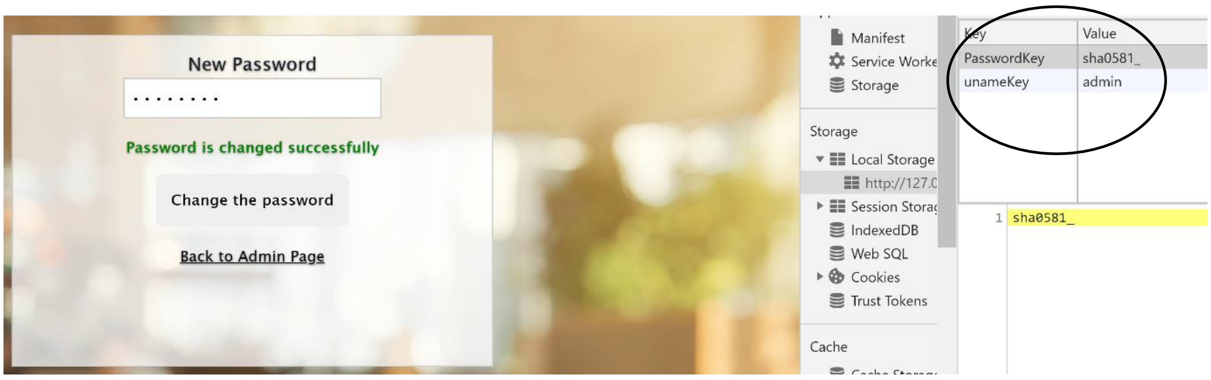


Case 1: If the text input is empty, a validation message will be shown.



Please fill the empty fields

Case 2: If the text input is filled correctly, the new value will be stored to the localStorage.



Password is changed successfully

Key	Value
PasswordKey	sha0581_
unameKey	admin

```
sha0581_
```

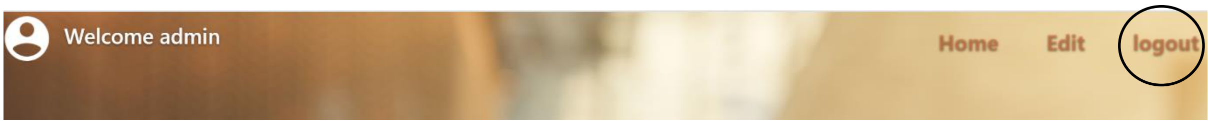
logOut function

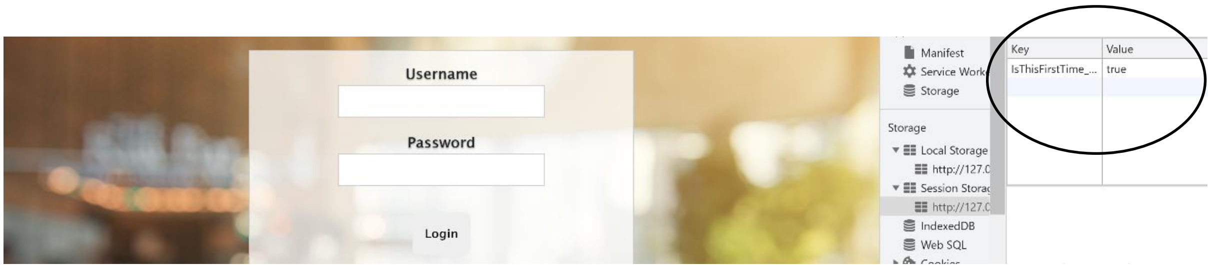
It clears the sessionStorage and redirects the admin user to the login page.

```
//Clear the sessionStorage and redirect the admin to the loginPage
function logOut() {
    sessionStorage.clear();
    location.href = "./loginPage.html"
}
```

Outputs:

Click logout to leave the page and clear the sessionStorage.





Key	Value
IsThisFirstTime_...	true

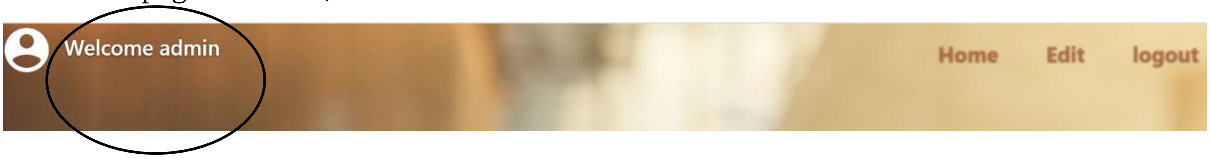
<u>Second file: AdminPage.js</u>

When the page is loaded and the sessionStorage key is checked correctly, a welcome message will be displayed with the username. Otherwise, the admin user will be redirected to the login page.

```
//-------------- Admin Page --------------
//Check if the sessionStorage key exists, or the login page will be opened.
window.onload = function () {
    if (sessionStorage.getItem('unameKey')) {
        //Show the localStorage value (unameKey) on the page
        document.getElementById('welcomeMsg').textContent = "Welcome " +
            localStorage.getItem('unameKey');
    }
    else {
        location.href = "./loginPage.html"
    }
}
```

Outputs:

When the page is loaded, the admin username will be shown.



<u>addItem function</u>

This function is used to add several food items to the localStorage by the admin user.

- Variables are declared outside the function to be global to all other functions.

```
var submit = document.getElementById("submit");
var itemCodeInput = document.getElementById("itemCode")
var itemNameInput = document.getElementById("itemName")
var ItemDetailsInput = document.getElementById("ItemDetails")
var priceInput = document.getElementById("price")
var ItemTypeInput = document.getElementById("ItemType")
var emptyFields = document.getElementById("emptyFields");
var noSelect = document.getElementById("noSelect");
```

- Firstly, the admin user enters the correct values for each variable in text fields.
- Getting the values from the text inputs.
- Creating the FoodData object to have all five variables. In order to store the data to the localStorage as an object, not as individual variables.

```
// addItem() function to add item to local storage by the admin
function addItem() {
    let StoredFoodData = JSON.parse(localStorage.getItem("itemInStorage"));
    //Use trim to remove spaces from the text
    var itemCode = itemCodeInput.value.trim()
    var itemName = itemNameInput.value.trim()
    var ItemDetails = ItemDetailsInput.value.trim()
    var price = priceInput.value.trim()
    var ItemType = ItemTypeInput.value;
    //Creating an object for the five variables.
    var FoodData = {
        itemCode: itemCode,
        itemName: itemName,
        ItemDetails: ItemDetails,
        price: price,
        ItemType: ItemType
    }
```

The conditions begin with checking all fields.

```
 // Validate if all the values are entered
    if (!itemCode || !itemName || !ItemDetails || !price) {
        emptyFields.style.display = "block";
        noSelect.style.display = "none";
    }
    else if (!ItemType) {
        noSelect.style.display = "block";
        emptyFields.style.display = "none";
    }
```

- The following condition starts with checking StoredFoodData key existence, which means that there is already added value.
- If the previous condition is met, the next condition will start, which is to verify that the **itemCode** in the array is undefined and not repeated (unique).
- if it is met, then it will start adding another array.
- Afterward, updating the localStorage values by storing another new value.
- If there is no StoredFoodData key, the system will start by entering the first array into it.

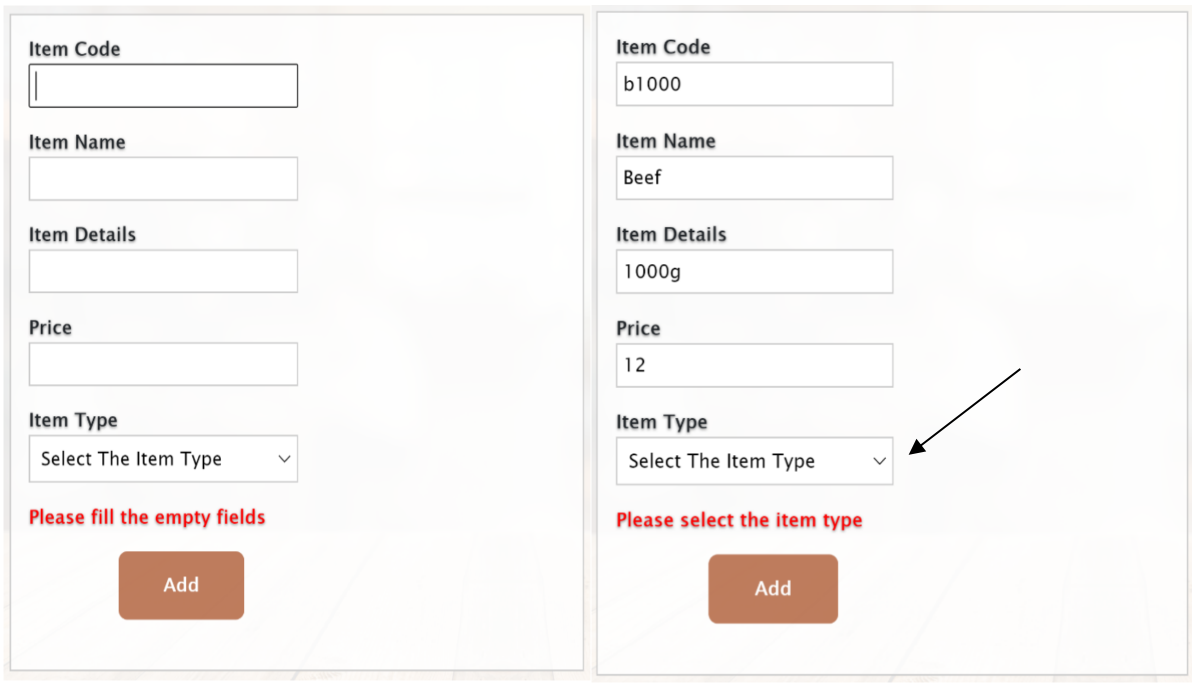
```
   else {
        // Validate if localStorage key (StoredFoodData) exists
        if (StoredFoodData != null) {
            //To store more than one array
            if (StoredFoodData[FoodData.itemCode] == undefined) {
                StoredFoodData = {
                 //use ... 3 dots to insert new item array to StoredFoodData.
                    ...StoredFoodData,
                    [FoodData.itemCode]: FoodData
                }
            }
        }
       //If there is no localStorage key, create it by adding the first array.
        else {

            StoredFoodData = {
                [FoodData.itemCode]: FoodData
            }
        }
```

- Using **JSON.stringify()** which converts JavaScript object to a string stringify to store data in the localStorage.

```
 //use setItem method to save the values inside the localStorage
        //JSON.stringify to convert the JS object to a string
        localStorage.setItem("itemInStorage", JSON.stringify(StoredFoodData));
        //Clear the inputs after submitting
        itemCodeInput.value = "";
        itemNameInput.value = "";
        ItemDetailsInput.value = "";
        priceInput.value = "";
        ItemTypeInput.selectedIndex = 0;
        emptyFields.style.display = "none";
        noSelect.style.display = "none";

        //call the function
        ShowItemsIntheAdminPage();
    }
}
```

Outputs:

Case 1: If the text inputs are empty or there is no option selection, a validation message will be shown.



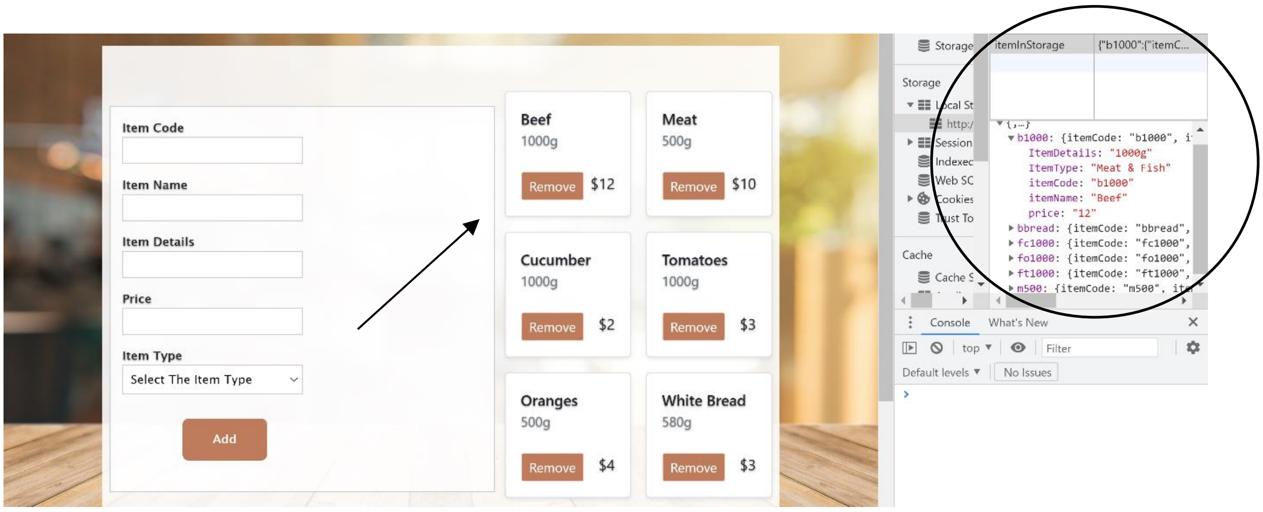
Item Code	
Item Name	
Item Details	
Price	
Item Type	Select The Item Type

Please fill the empty fields

Item Code	b1000
Item Name	Beef
Item Details	1000g
Price	12
Item Type	Select The Item Type

Please select the item type

Case 2: If the text inputs are filled correctly, the new values will be stored to the localStorage and shown on admin page.

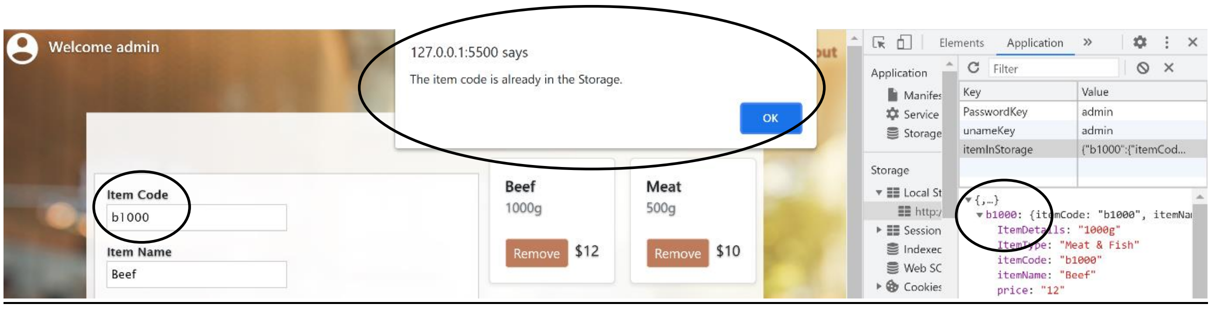


Beef	1000g	$12
Meat	500g	$10
Cucumber	1000g	$2
Tomatoes	1000g	$3
Oranges	500g	$4
White Bread	580g	$3

itemInStorage	{"b1000":{"itemC...

```
▼{,…}
  ▼b1000: {itemCode: "b1000", i
      ItemDetails: "1000g"
      ItemType: "Meat & Fish"
      itemCode: "b1000"
      itemName: "Beef"
      price: "12"
  ▶bbread: {itemCode: "bbread",
  ▶fc1000: {itemCode: "fc1000",
  ▶fo1000: {itemCode: "fo1000",
  ▶ft1000: {itemCode: "ft1000",
  ▶m500: {itemCode: "m500", ite
```

Case 3: If the same itemCode is entered, a validation message will be shown.



127.0.0.1:5500 says

The item code is already in the Storage.

Item Code	b1000
Item Name	Beef

Key	Value
PasswordKey	admin
unameKey	admin
itemInStorage	{"b1000":{"itemCod...

```
▼{,…}
  ▼b1000: {itemCode: "b1000", itemNa
      ItemDetails: "1000g"
      ItemType: "Meat & Fish"
      itemCode: "b1000"
      itemName: "Beef"
      price: "12"
```

ShowItemsIntheAdminPage Function

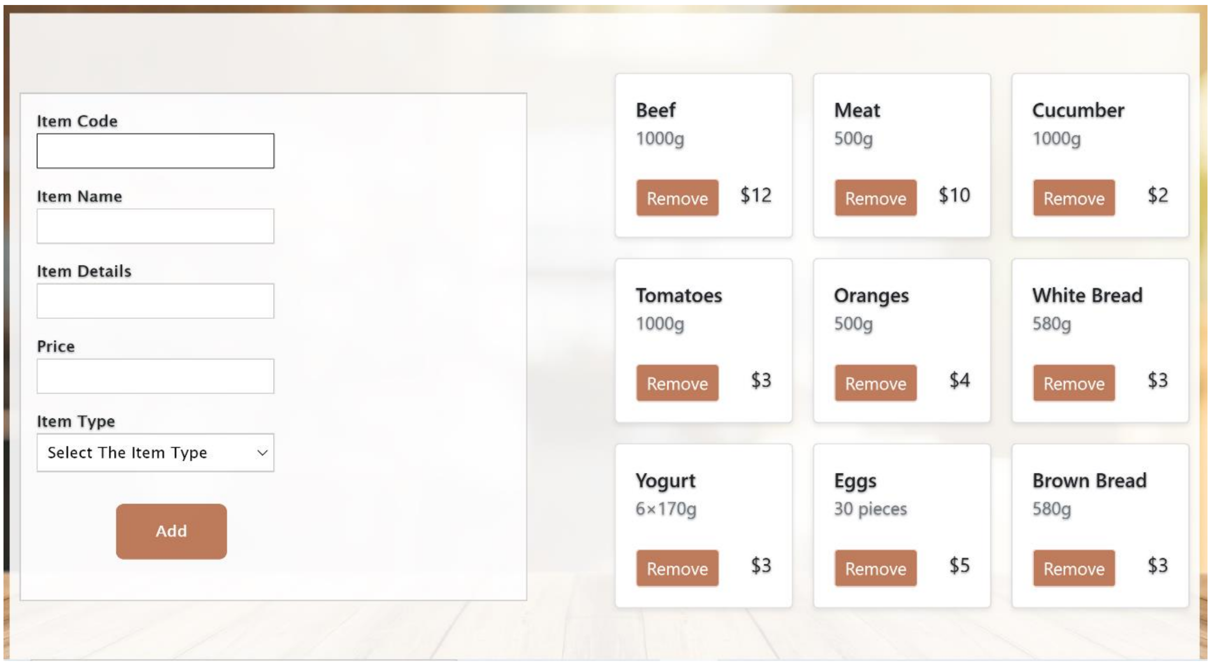
This function displays the contents of the StoredFoodData key on the HTML admin page.

- Using **JSON.parse()** which converts the string to a JavaScript object to retrieve data from localStorage.

```
// Get the items from the localStorage and show them in AdminPage.
function ShowItemsIntheAdminPage() {
    //use getItem method to retrive the values from localStorage
    //JSON.parse to convert the string to JS object
    let StoredFoodData = localStorage.getItem("itemInStorage");
    StoredFoodData = JSON.parse(StoredFoodData);
    let AdminItemsDiv = document.querySelector(".AdminItems");

    //check if the localStorage key (StoredFoodData) exists
    if (StoredFoodData) {
        AdminItemsDiv.innerHTML = '';
        //Use Object.values() to return an array of values
        //Use map() to create a new array that helps to get each array element
 and show them in HTML content.
        Object.values(StoredFoodData).map(item => {
            AdminItemsDiv.innerHTML += `
              <div class="col">
              <div class="card shadow-sm">
               <span style="display:none">${item.itemCode}</span>
                <div class="card-body">
                  <span class="card-text">${item.itemName}</span>
                  <p class="text-muted">${item.ItemDetails}</p>
                  <div class="d-flex justify-content-between align-items-
center">

                    <div class="btn-group">
                      <button type="button" class="btn btn-sm btn-outline-
secondary btnRemove">Remove</button>
                    </div>
                    <label class="price" >$${item.price}</label>
                  </div>
                </div>
              </div>
            </div>
            `
        });

    }
    //call the function
    RemoveButtons();
}
```

Output:

The stored values in StoredFoodData key are shown on the admin page.



Item Code

Item Name

Item Details

Price

Item Type

Select The Item Type

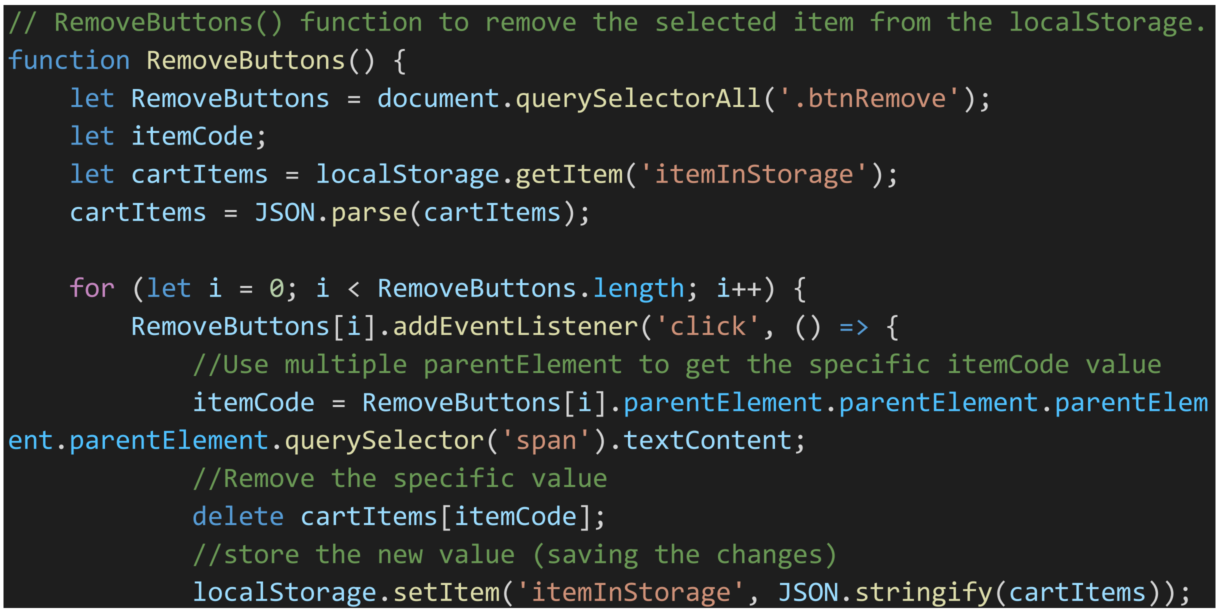
Add

Beef 1000g Remove $12	Meat 500g Remove $10	Cucumber 1000g Remove $2
Tomatoes 1000g Remove $3	Oranges 500g Remove $4	White Bread 580g Remove $3
Yogurt 6×170g Remove $3	Eggs 30 pieces Remove $5	Brown Bread 580g Remove $3

<u>RemoveButtons() function</u>

It removes the selected item by clicking the responsible button for it.

- Calling all buttons with the same class name
- Looping through the buttons
- Specifying each ItemCode with the button associated with it
- Executing the function by using **<u>delete</u>** cartItems[itemCode]

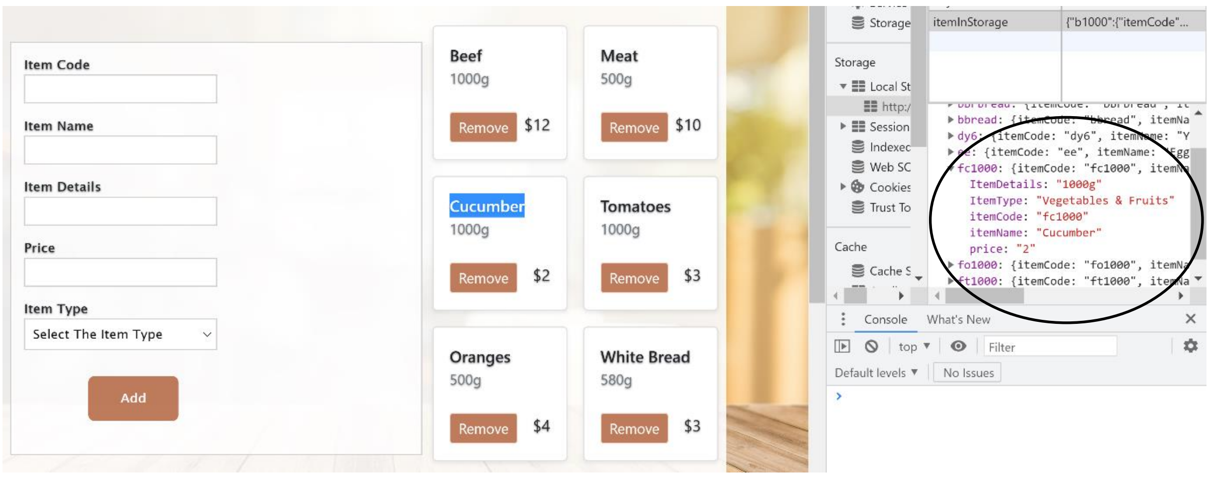


```
// RemoveButtons() function to remove the selected item from the localStorage.
function RemoveButtons() {
    let RemoveButtons = document.querySelectorAll('.btnRemove');
    let itemCode;
    let cartItems = localStorage.getItem('itemInStorage');
    cartItems = JSON.parse(cartItems);

    for (let i = 0; i < RemoveButtons.length; i++) {
        RemoveButtons[i].addEventListener('click', () => {
            //Use multiple parentElement to get the specific itemCode value
            itemCode = RemoveButtons[i].parentElement.parentElement.parentElem
ent.parentElement.querySelector('span').textContent;
            //Remove the specific value
            delete cartItems[itemCode];
            //store the new value (saving the changes)
            localStorage.setItem('itemInStorage', JSON.stringify(cartItems));
            ShowItemsIntheAdminPage();
        });
    }
}
//call the function when the page is loaded
ShowItemsIntheAdminPage();
```

Output:

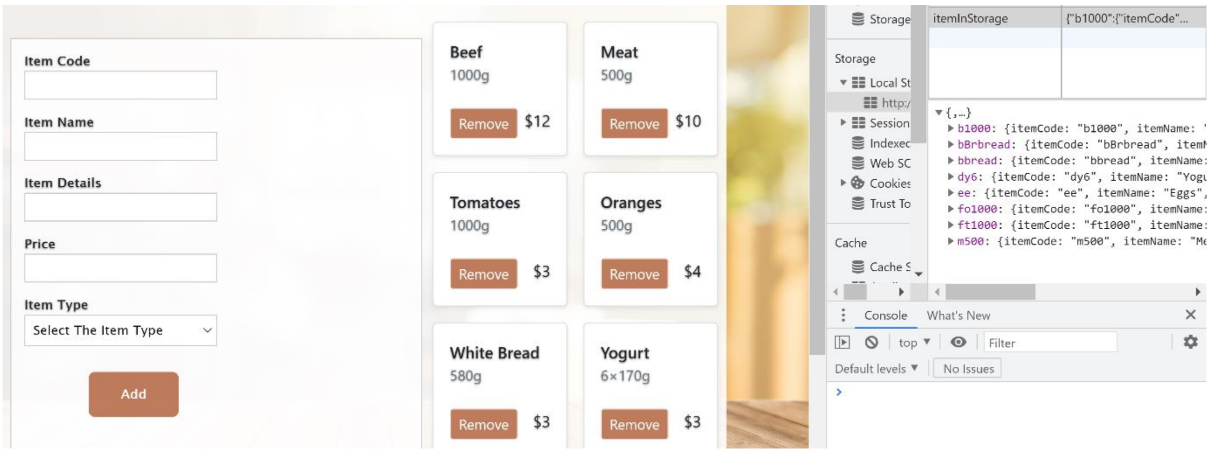
When the admin user clicks the remove button, all the values that linked to it will be deleted from the localStorage and the data disappeared from the page.

- Before removing Cucumber (<u>itemCode: fc1000</u>):



```
itemInStorage    {"b1000":{"itemCode"...

▸ bbread: {itemCode: "bbread", itemNa
▸ dy6: {itemCode: "dy6", itemName: "Y
▸ ee: {itemCode: "ee", itemName: "Egg
▾ fc1000: {itemCode: "fc1000", itemNa
    ItemDetails: "1000g"
    ItemType: "Vegetables & Fruits"
    itemCode: "fc1000"
    itemName: "Cucumber"
    price: "2"
▸ fo1000: {itemCode: "fo1000", itemNa
▸ ft1000: {itemCode: "ft1000", itemNa
```

- After removing:



```
itemInStorage    {"b1000":{"itemCode"...

▾ {,…}
  ▸ b1000: {itemCode: "b1000", itemName: '
  ▸ bBrbread: {itemCode: "bBrbread", itemN
  ▸ bbread: {itemCode: "bbread", itemName:
  ▸ dy6: {itemCode: "dy6", itemName: "Yogu
  ▸ ee: {itemCode: "ee", itemName: "Eggs",
  ▸ fo1000: {itemCode: "fo1000", itemName:
  ▸ ft1000: {itemCode: "ft1000", itemName:
  ▸ m500: {itemCode: "m500", itemName: "Me
```

<u>Third file: Customer.js</u>

The Customer.js file is linked to three HTML pages, which are the Home page, Cart Page and Confirmation Page.

The related functions to the cart page.

<u>ShowItemsIntheHomePage() function</u>

It displays the contents of the StoredFoodData key on the HTML Home page to let customers select from them.

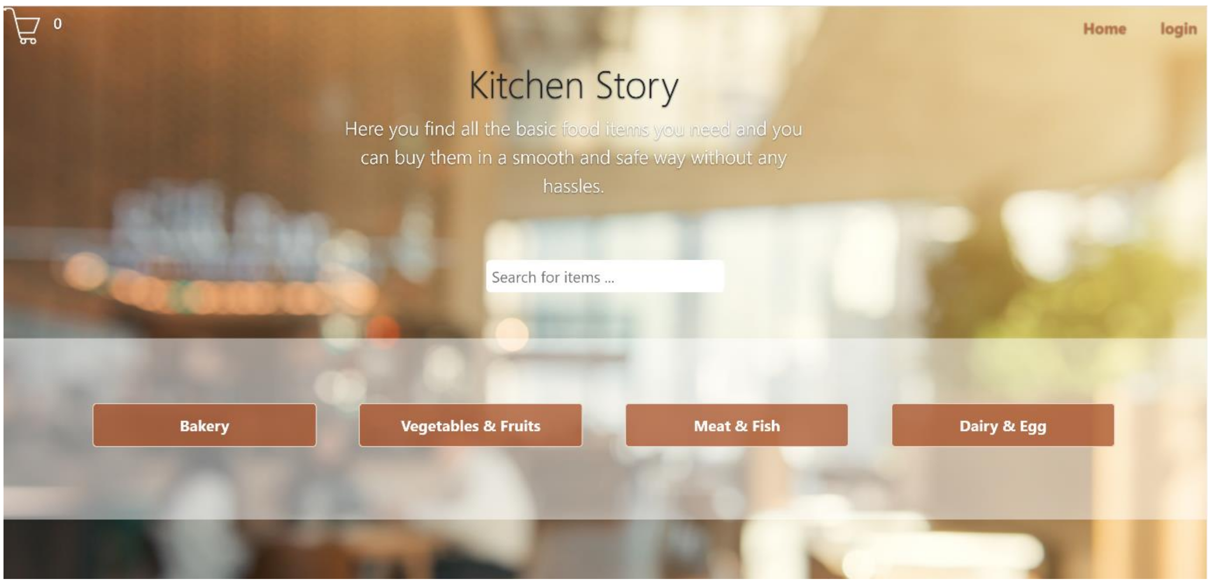
```
// Get the items from the local storage and show them in HomePage.
function ShowItemsIntheHomePage() {
    let StoredFoodData = localStorage.getItem("itemInStorage");
    StoredFoodData = JSON.parse(StoredFoodData);
    //select each button with class "itemTypeBtn"
    var itemTypeBtn = document.querySelectorAll(".itemTypeBtn");
    let itemTypeDiv = document.querySelector(".itemTypeDiv");
    //check if the HomePage is currently used, not other
    if (StoredFoodData && itemTypeDiv) {
        //Creating a loop for each button
        for (let i = 0; i < itemTypeBtn.length; i++) {
            itemTypeBtn[i].addEventListener('click', () => {
                itemTypeDiv.innerHTML = '';
                Object.values(StoredFoodData).map(item => {
                    if (item.ItemType == itemTypeBtn[i].textContent) {

                        itemTypeDiv.innerHTML += `
                      <div class="col specificItem">
              <div class="card shadow-sm">
              <span style="display:none">${item.itemCode}</span>
                <div class="card-body">
                  <span class="card-text">${item.itemName}</span>
                  <p class="text-muted">${item.ItemDetails}</p>
                  <div class="d-flex justify-content-between align-items-
center">

                    <div class="btn-group">
                    <button type="button" class="btn btn-sm btn-outline-
secondary btnAddtoCart">Add to Cart</button>
                    </div>
                    <label class="price" >$${item.price}</label>
                  </div>
                </div>
              </div>
            </div>
                    `
```

```
                        }

                    });

                    AddCartButtons();
                })
            }
        }
    }
}
```

Output:

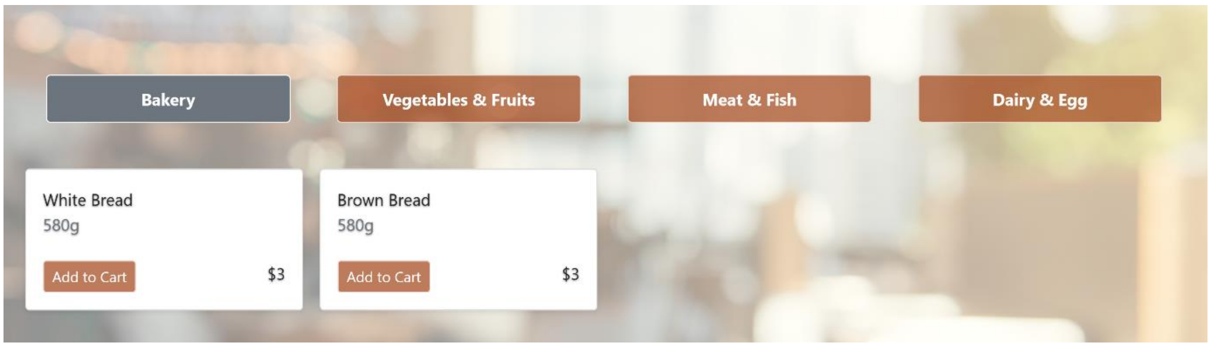
The stored values in StoredFoodData key (the admin user added them) are shown on the Home page.

- There are four sections, any section that a customer clicks on, will display only the values related to the same item type.


0
Home
login
Kitchen Story
Here you find all the basic food items you need and you can buy them in a smooth and safe way without any hassles.
Search for items ...
Bakery
Vegetables & Fruits
Meat & Fish
Dairy & Egg

For example:

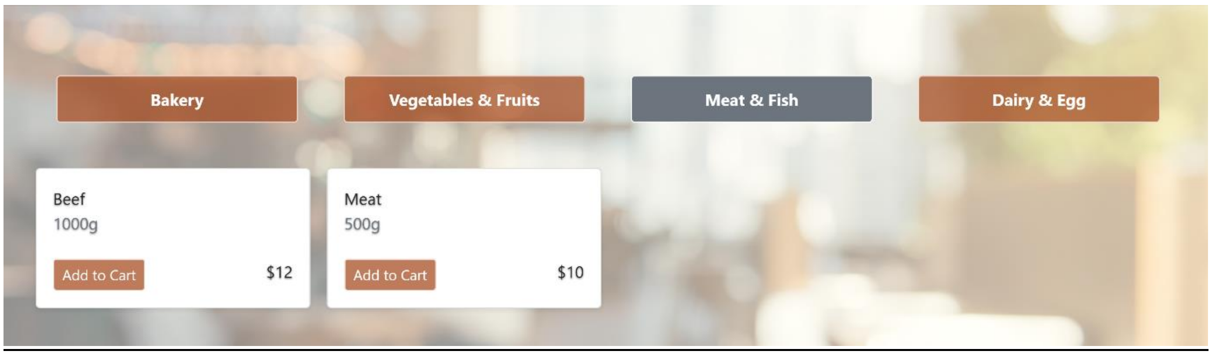
- Clicking Bakery Section



White Bread
580g
$3

Brown Bread
580g
$3

- Clicking Meat & Fish Section



Beef
1000g
$12

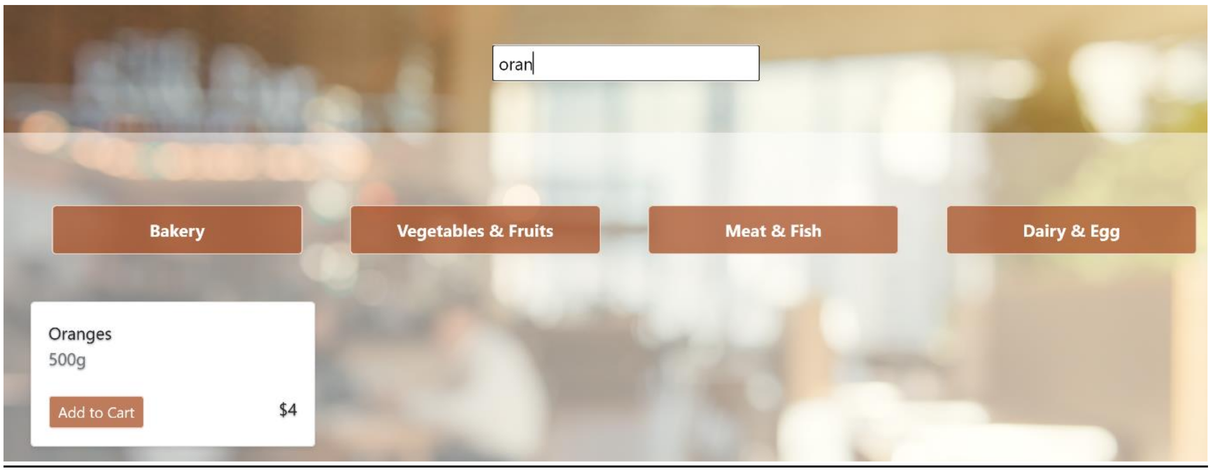
Meat
500g
$10

<u>searchAboutItems() function</u>

It searches for any item inside the div called specificItem which contains the values of the variables such as ItemName, ItemDetails, and the price by typing inside a search text input.

```
//Search function
function searchAboutItems() {
    var input = document.getElementById("Search");
    //filter the input by disabling the case-sensitive.
    var filter = input.value.toLowerCase();
    var specificItem = document.getElementsByClassName('specificItem');

    for (i = 0; i < specificItem.length; i++) {
        if (specificItem[i].innerText.toLowerCase().includes(filter)) {
            specificItem[i].style.display = "block";
        } else {
            specificItem[i].style.display = "none";
        }
    }
}
```

Output: After typing the first letters of the word oranges, the required item is shown.



Oranges
500g
$4

AddCartButtons() function

This function adds food items to the customer cart by getting the contents of the items from the HTML page and storing them in local Storage (cartItems key).

- Calling all buttons with the same class name (btnAddtoCart class)
- Looping through the buttons
- Specifying all the item variables with the button associated with them.
- Creating an object called cartItems to have all the variables besides a new variable called ItemQuantity.
- Executing the function based on the required conditions.
- The following steps were previously explained in addItem() function on the AdminPage.js.

```
// AddCartButtons() to add Items from the Html content to the local storage
//(cartItems) which belongs to customer cart.
function AddCartButtons() {
    let AddButtons = document.querySelectorAll('.btnAddtoCart');

    for (let i = 0; i < AddButtons.length; i++) {
        AddButtons[i].addEventListener('click', () => {
            //Get the specific items' values from text content by using multip
le parentElement
            //and select the html element
            let itemCode = AddButtons[i].parentElement.parentElement.parentEle
ment.parentElement.querySelector('span').textContent;
            let itemName = AddButtons[i].parentElement.parentElement.parentEle
ment.querySelector('span').textContent;
            let price = AddButtons[i].parentElement.parentElement.querySelecto
r('label').textContent.trim().replace('$', '');;
            price = parseInt(price);

            let ItemDetails = AddButtons[i].parentElement.parentElement.parent
Element.querySelector('p').textContent;
            //Create an object to deal with several inputs
            var cartItems = {
                itemCode: itemCode,
                itemName: itemName,
                price: price,
                ItemDetails: ItemDetails,
                ItemQuantity: 1
            }

            let CartFoodData = localStorage.getItem("cartItems");
            CartFoodData = JSON.parse(CartFoodData);
            if (CartFoodData != null) {

                if (CartFoodData[cartItems.itemCode] == undefined) {
                    CartFoodData = {
                   //use ... 3 dots to insert new item array to CartFoodData.
                        ...CartFoodData,
                        [cartItems.itemCode]: cartItems
                    }
```

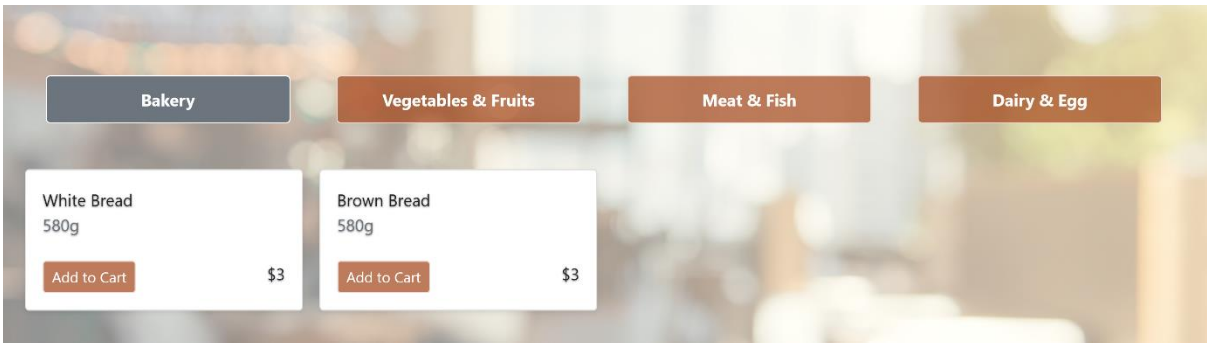
In this same function, calculating the total price of the items and store it in a localStorage key called totalPrice.

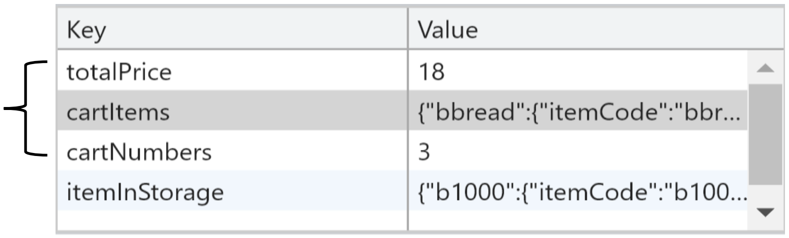
In addition, the cart counter is calculated each time the number of items increases or decreases, then stored in the cartNumbers key.

```
let totalPrice = localStorage.getItem("totalPrice");
                    //calculate the total price
                    if (totalPrice != null) {
                        totalPrice = parseInt(totalPrice);
                        localStorage.setItem("totalPrice", totalPrice + price)
;
                    }

                    //count the cart numbers
                    let cartNumbers = localStorage.getItem("cartNumbers");
                    cartNumbers = parseInt(cartNumbers);
                    //if cartNumbers exists, increment the value
                    //and show them in span element by using textContent
                    if (cartNumbers) {
                        localStorage.setItem('cartNumbers', cartNumbers + 1);
                        document.querySelector('.cartNums span').textContent =
 cartNumbers + 1;
                    }

                }
                else {
                    alert("it is already in cart")
                }
            }
            else {

                CartFoodData = {
                    [cartItems.itemCode]: cartItems
                }
                //store the new value (saving the changes)
                localStorage.setItem("totalPrice", cartItems.price);
                localStorage.setItem('cartNumbers', 1);
                document.querySelector('.cartNums span').textContent = 1;

            }
            localStorage.setItem("cartItems", JSON.stringify(CartFoodData));
            //call the function
            onLoadingCartCounter();
        });
    }
}
```

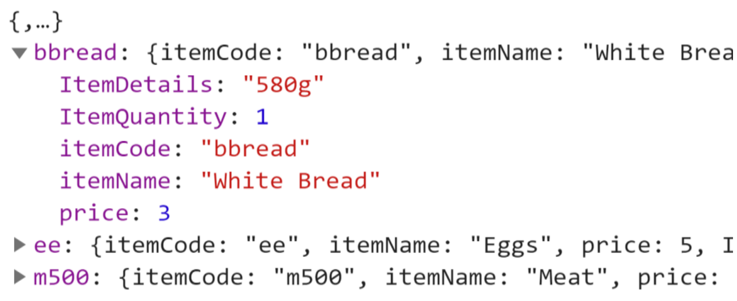
Output:

- After the customer clicks on Add to cart button, the values will be stored in cartItems key.
- Creating totalPrice key and cartNumbers key in the localStorage.



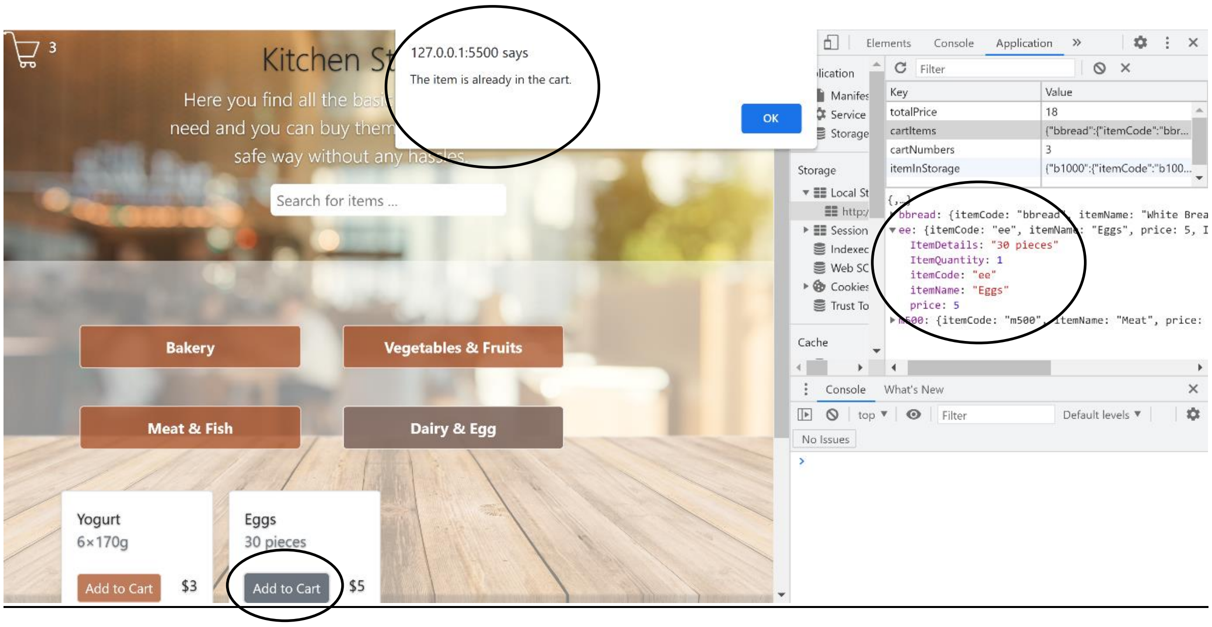


Key	Value
totalPrice	18
cartItems	{"bbread":{"itemCode":"bbr...
cartNumbers	3
itemInStorage	{"b1000":{"itemCode":"b100...



```
{,…}
▼bbread: {itemCode: "bbread", itemName: "White Brea
    ItemDetails: "580g"
    ItemQuantity: 1
    itemCode: "bbread"
    itemName: "White Bread"
    price: 3
▶ee: {itemCode: "ee", itemName: "Eggs", price: 5, I
▶m500: {itemCode: "m500", itemName: "Meat", price:
```

- If the customer clicked on the same item, a validation message will be shown.



127.0.0.1:5500 says

The item is already in the cart.

Key	Value
totalPrice	18
cartItems	{"bbread":{"itemCode":"bbr…
cartNumbers	3
itemInStorage	{"b1000":{"itemCode":"b100…

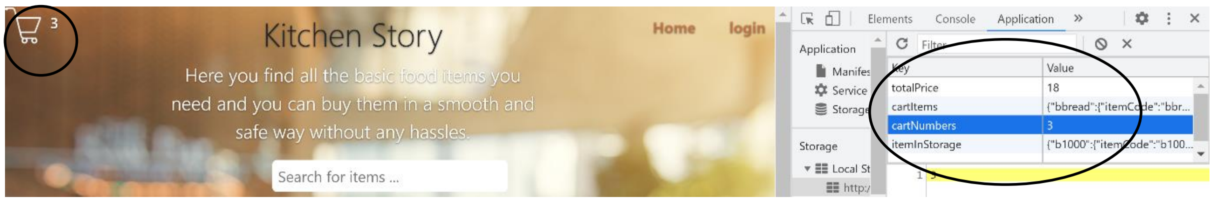
```
{,…}
▸ bbread: {itemCode: "bbread", itemName: "White Brea
▾ ee: {itemCode: "ee", itemName: "Eggs", price: 5, I
    ItemDetails: "30 pieces"
    ItemQuantity: 1
    itemCode: "ee"
    itemName: "Eggs"
    price: 5
▸ m500: {itemCode: "m500", itemName: "Meat", price:
```

onLoadingCartCounter() function

It uses to display the cart counter whenever the page is loaded to keep the value always updated.

```
//to show the cart counter when the page is loaded
function onLoadingCartCounter() {
    let cartNumbers = localStorage.getItem("cartNumbers");
    if (cartNumbers) {
        document.querySelector('.cartNums span').textContent = cartNumbers;
    }
}


//call the functions when the Home page is loaded
ShowItemsIntheHomePage();
onLoadingCartCounter();
```

Output: the cartNumbers value will be shown beside the cart icon.



Key	Value
totalPrice	18
cartItems	{"bbread":{"itemCode":"bbr…
cartNumbers	3
itemInStorage	{"b1000":{"itemCode":"b100…

The related functions to the cart page.

ShowItemsInCustomerCart() function

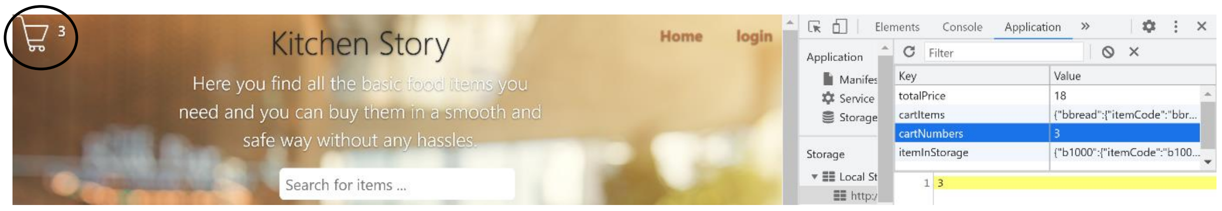
It displays the contents of the cartItems key and totalPrice key on the HTML cart page.

```
//-------------- Cart Page --------------
function ShowItemsInCustomerCart() {
    let CartFoodData = localStorage.getItem("cartItems");
    CartFoodData = JSON.parse(CartFoodData);
    let itemDiv = document.querySelector(".Customeritems");
    let totalPrice = localStorage.getItem('totalPrice');
    //check if the CartPage is currently used, not other
    if (CartFoodData && itemDiv) {
        itemDiv.innerHTML = '';
        Object.values(CartFoodData).map(item => {
            itemDiv.innerHTML += `

            <div class="rowClass row mb-5 text-center">
            <div class="col-1 themed-grid-col text-muted">
             <span style="display:none">${item.itemCode}</span>
            <button type="button" class="remove"><img class="img-
style" src="./images/x.png" /></button>
            </div>
             <div class="col-4 themed-grid-col text-muted"> <h6 class="my-
0">${item.itemName}</h6>
             <small class="text-muted">${item.ItemDetails}</small>
             </div>
             <div class="col-2 themed-grid-col text-muted"><span class="text-
muted price">$${item.price}</span></div>
             <div class="col-3 themed-grid-col text-muted">
             <button type="button" class="decrease"><img class="img-
style" src="./images/minus.png" /></button>
                  <span class="text-
muted Quantity">${item.ItemQuantity}</span>
                  <button type="button" class="increase"><img class="img-
style" src="./images/plus.png" /></button>
             </div>
             <div class="col-2 themed-grid-col text-muted"><span class="text-
muted Total">$${item.ItemQuantity * item.price}</span></div>
           </div>
            `
        });

        itemDiv.innerHTML += `
        <div class="rowClass row mb-2">
        <div class="col-10 themed-grid-col text-
muted"><h6>Total Price</h6></div>
```

```
        <div class="col-2 themed-grid-col text-
end"><strong>$${totalPrice}</strong></div>

      </div>

          `;
    }
    RemoveCartItem();
    manageQuantity();
}
```

Output:

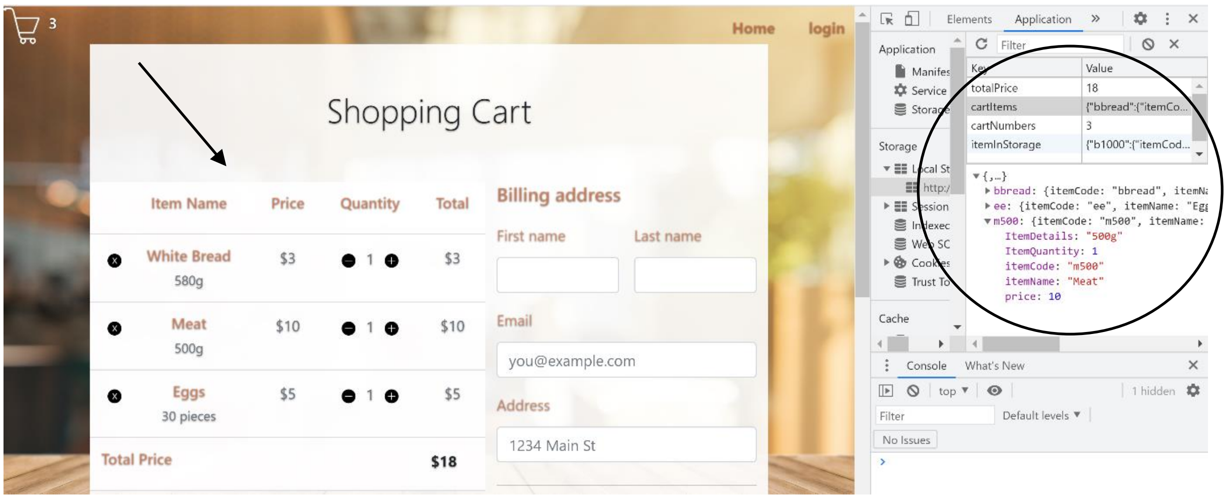
- When click on the cart icon, cart page will be opened.



Key	Value
totalPrice	18
cartItems	{"bbread":{"itemCode":"bbr...
cartNumbers	3
itemInStorage	{"b1000":{"itemCode":"b100...

```
1 3
```

- Displaying the cart contents.



	Item Name	Price	Quantity	Total
⊗	White Bread 580g	$3	⊖ 1 ⊕	$3
⊗	Meat 500g	$10	⊖ 1 ⊕	$10
⊗	Eggs 30 pieces	$5	⊖ 1 ⊕	$5
	Total Price			$18

Key	Value
totalPrice	18
cartItems	{"bbread":{"itemCo...
cartNumbers	3
itemInStorage	{"b1000":{"itemCod...

```
▼{,…}
  ▶bbread: {itemCode: "bbread", itemNa
  ▶ee: {itemCode: "ee", itemName: "Egg
  ▼m500: {itemCode: "m500", itemName:
     ItemDetails: "500g"
     ItemQuantity: 1
     itemCode: "m500"
     itemName: "Meat"
     price: 10
```

<u>manageQuantity() function</u>

This function is responsible for calculating the quantity of the items, through which the customer can increase or decrease the quantity as desired.

This function is divided into two main parts, the increase button, and the decrease button.

- Calling all buttons with the same class name
- Looping through the buttons

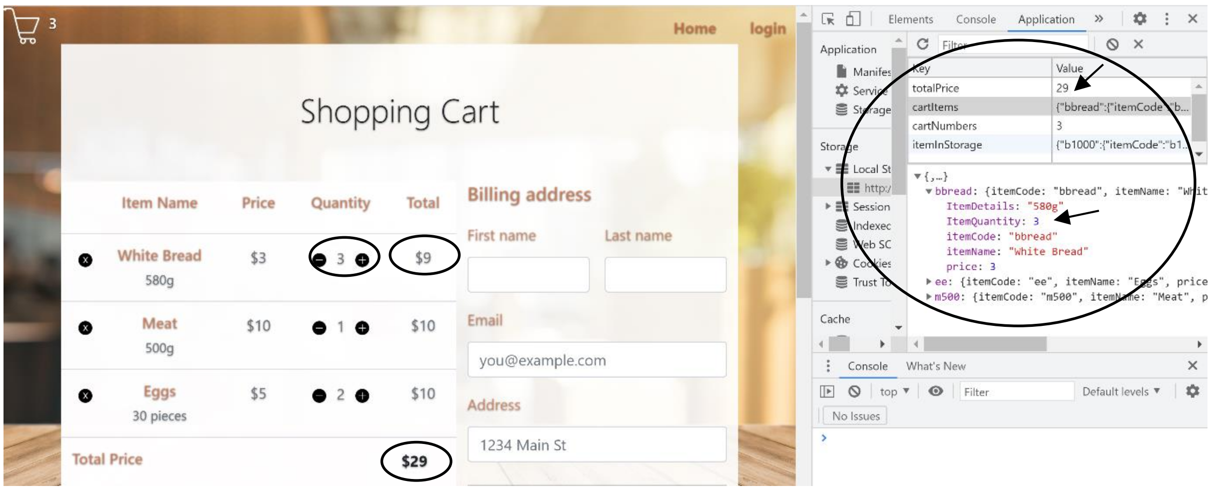
- Specifying the itemCode variable with the button associated with it.
- Executing the function based on the required conditions.

```
function manageQuantity() {
    let decreaseButtons = document.querySelectorAll('.decrease');
    let increaseButtons = document.querySelectorAll('.increase');
    let cartItems = localStorage.getItem('cartItems');
    let currentQuantity = 0;
    let itemCode = "";
    cartItems = JSON.parse(cartItems);
    let cartCost = localStorage.getItem('totalPrice');
    //parse to integer
    cartCost = parseInt(cartCost);
    //Creating a loop for each button
    for (let i = 0; i < decreaseButtons.length; i++) {
        decreaseButtons[i].addEventListener('click', () => {
          //Get the specific items' values from text content
          //by using multiple parentElement
            //or any appropriate node, then select the html element
            currentQuantity = decreaseButtons[i].parentElement.querySelector('
span').textContent;
            itemCode = decreaseButtons[i].parentElement.previousElementSibling
.previousElementSibling.previousElementSibling.querySelector('span').textConte
nt;
       //The decrease button will check that the quantity is greater than one
            // then start decreasing it.
            if (cartItems[itemCode].ItemQuantity > 1) {
                cartItems[itemCode].ItemQuantity -= 1;

                localStorage.setItem('totalPrice', cartCost - cartItems[itemCo
de].price);
                localStorage.setItem('cartItems', JSON.stringify(cartItems));
                ShowItemsInCustomerCart();
            }
        });
    }

    for (let i = 0; i < increaseButtons.length; i++) {
        increaseButtons[i].addEventListener('click', () => {
            currentQuantity = increaseButtons[i].parentElement.querySelector('
span').textContent;
            itemCode = increaseButtons[i].parentElement.previousElementSibling
.previousElementSibling.previousElementSibling.querySelector('span').textConte
nt;
            //The increase button will increase the quantity.
            cartItems[itemCode].ItemQuantity += 1;
            localStorage.setItem('totalPrice', cartCost + cartItems[itemCode].
price);
```

```
            localStorage.setItem('cartItems', JSON.stringify(cartItems));
            ShowItemsInCustomerCart();

        })
    }
}
```

Output:

Case 1: If the increase button is clicked, the quantity value, the total price for the selected item, and the total price of all items will be increased.

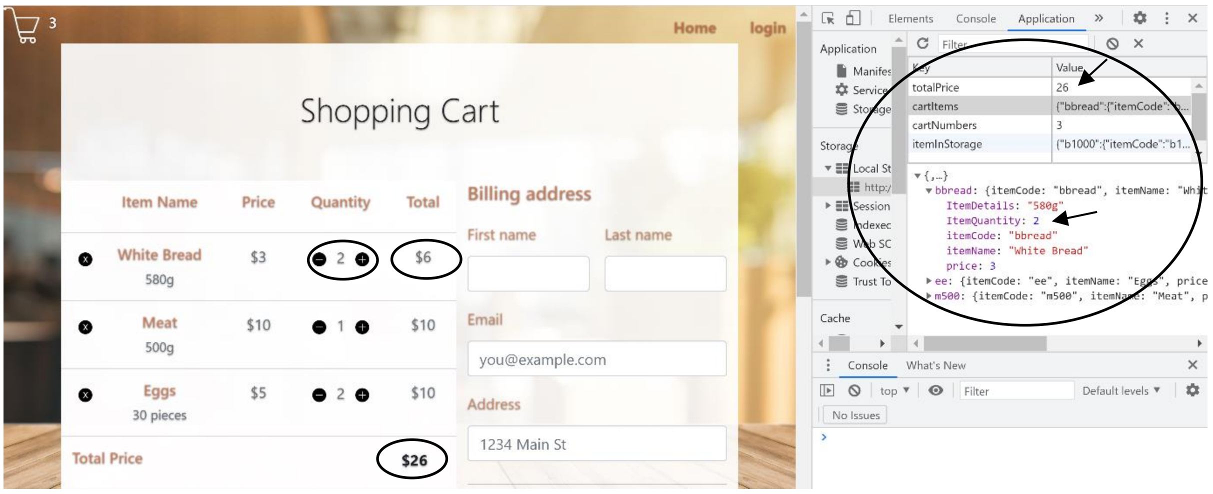


Item Name	Price	Quantity	Total
White Bread 580g	$3	3	$9
Meat 500g	$10	1	$10
Eggs 30 pieces	$5	2	$10
Total Price			$29

Key	Value
totalPrice	29
cartItems	{"bbread":{"itemCode":"b...
cartNumbers	3
itemInStorage	{"b1000":{"itemCode":"b1...

```
▼{,…}
  ▼bbread: {itemCode: "bbread", itemName: "Whit
     ItemDetails: "580g"
     ItemQuantity: 3
     itemCode: "bbread"
     itemName: "White Bread"
     price: 3
  ▶ee: {itemCode: "ee", itemName: "Eggs", price
  ▶m500: {itemCode: "m500", itemName: "Meat", p
```

Case 2: If the decrease button is clicked, the quantity value, the total price for the selected item, and the total price of all items will be decreased.



Item Name	Price	Quantity	Total
White Bread 580g	$3	2	$6
Meat 500g	$10	1	$10
Eggs 30 pieces	$5	2	$10
Total Price			$26

Key	Value
totalPrice	26
cartItems	{"bbread":{"itemCode":"b...
cartNumbers	3
itemInStorage	{"b1000":{"itemCode":"b1...

```
▼{,…}
  ▼bbread: {itemCode: "bbread", itemName: "Whit
     ItemDetails: "580g"
     ItemQuantity: 2
     itemCode: "bbread"
     itemName: "White Bread"
     price: 3
  ▶ee: {itemCode: "ee", itemName: "Eggs", price
  ▶m500: {itemCode: "m500", itemName: "Meat", p
```

RemoveCartItem() function

It removes the selected item from the cart by clicking the responsible button for it.

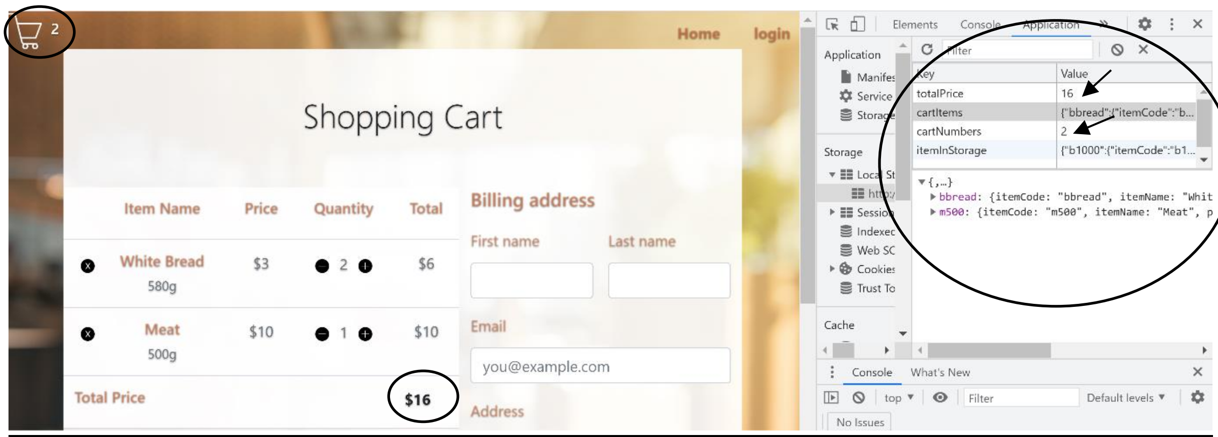
- Calling all buttons with the same class name (remove class)
- Looping through the buttons
- Specifying each ItemCode with the button associated with it
- Executing the function by using **delete** cartItems[itemCode];

During the deletion process, the total price and the cart counter will be affected (decreasing).

```
//Remove button from the customer cart
function RemoveCartItem() {
    let deleteButtons = document.querySelectorAll('.remove');
    let itemCode;
    let cartNumbers = localStorage.getItem('cartNumbers');
    let cartItems = localStorage.getItem('cartItems');
    cartItems = JSON.parse(cartItems);
    let cartCost = localStorage.getItem('totalPrice');

    for (let i = 0; i < deleteButtons.length; i++) {
        deleteButtons[i].addEventListener('click', () => {
   //Get the specific items' values from text content by using parentElement
            //then select the html element
            itemCode = deleteButtons[i].parentElement.querySelector('span').te
xtContent;
            //decrease the item quantity in the localStorage
            localStorage.setItem('cartNumbers', cartNumbers - 1);
            //decrease the item price from the total price in the localStorage
            localStorage.setItem('totalPrice', cartCost - (cartItems[itemCode]
.price * cartItems[itemCode].ItemQuantity));
            //Remove the specific item from the localStorage and store the cha
nges.
            delete cartItems[itemCode];
            localStorage.setItem('cartItems', JSON.stringify(cartItems));

            //call the functions
            ShowItemsInCustomerCart();
            onLoadingCartCounter();
        });
    }
}
```

Output:

- If the remove button is clicked, all the related values of the item will be removed, and the total price will be decreased.



Item Name	Price	Quantity	Total
White Bread 580g	$3	2	$6
Meat 500g	$10	1	$10
Total Price			$16

Key	Value
totalPrice	16
cartItems	{"bbread":{"itemCode":"b...
cartNumbers	2
itemInStorage	{"b1000":{"itemCode":"b1...

```
▼{,…}
 ▶bbread: {itemCode: "bbread", itemName: "Whit
 ▶m500: {itemCode: "m500", itemName: "Meat", p
```

payment() function

In this function cartItems values will be copied to TheOrderItems (paid) and totalPrice value to PaidtotalPrice value, then clear cartItems and totalPrice keys by using the **removeItem()** method.

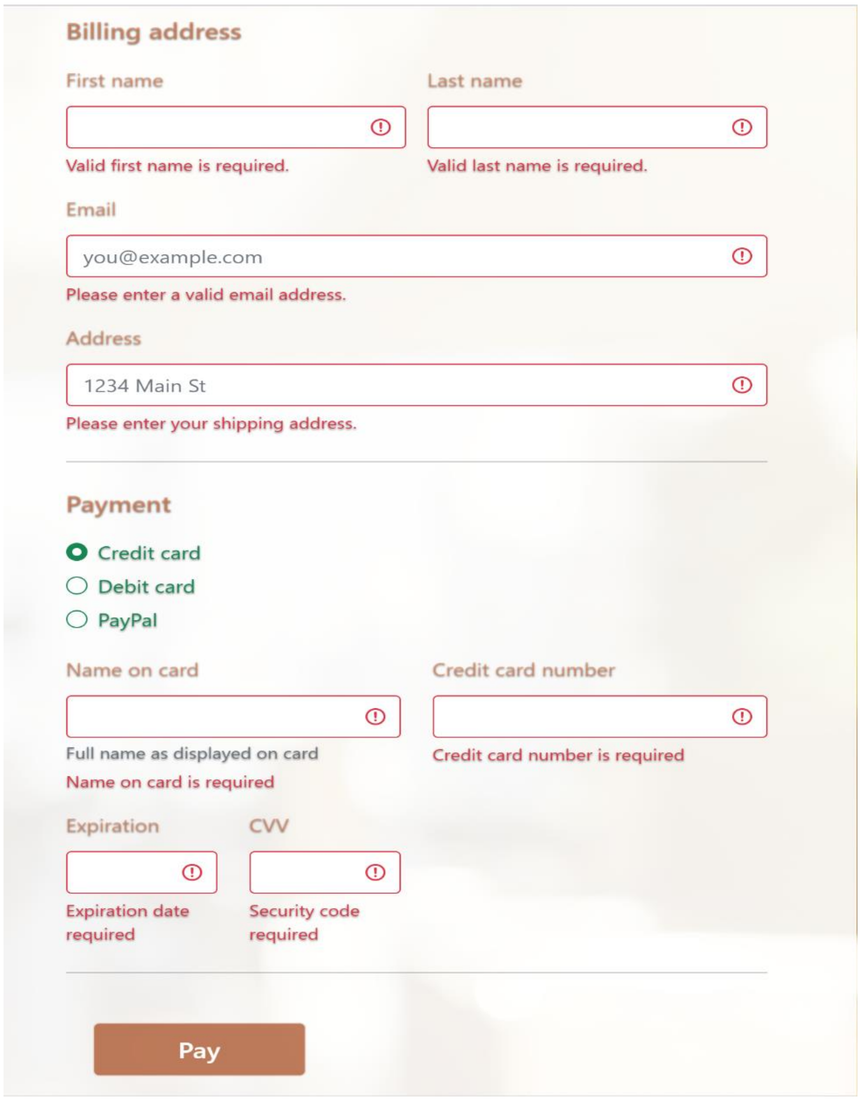
- Removing the keys means all the items in the customer cart will also be removed.

In filling out the payment form, a custom Bootstrap is used to validate the text inputs.

```
function payment() {
    let TheOrderItems = localStorage.getItem("cartItems");
    let PaidtotalPrice = localStorage.getItem("totalPrice");
    'use strict'

// Fetch all the forms we want to apply custom Bootstrap validation styles to
    var forms = document.querySelectorAll('.needs-validation')

    // Loop over them and prevent submission
    Array.prototype.slice.call(forms)
        .forEach(function (form) {
            form.addEventListener('submit', function (event) {
                if (!form.checkValidity()) {
                    event.preventDefault()
                    event.stopPropagation()
                }
                else {
                    //copy the local storage
                    localStorage.setItem("TheOrderItems", TheOrderItems);
                    localStorage.setItem("PaidtotalPrice", PaidtotalPrice);
                    //clear the cart by remove cartItems
                    localStorage.removeItem("cartItems");
                    localStorage.removeItem("totalPrice");
                    localStorage.removeItem("cartNumbers");
                    location.href = './ConfirmationPage.html';
                }
                form.classList.add('was-validated')

            }, false)
        })
}
//call the function when the page is loaded
ShowItemsInCustomerCart();
```

Outputs:

Case 1: If the text inputs are empty, validation messages will be shown.



Billing address

First name

Valid first name is required.

Last name

Valid last name is required.

Email

you@example.com

Please enter a valid email address.

Address

1234 Main St

Please enter your shipping address.

Payment

[✓ Credit card]

Debit card

PayPal

Name on card

Full name as displayed on card

Name on card is required

Credit card number

Credit card number is required

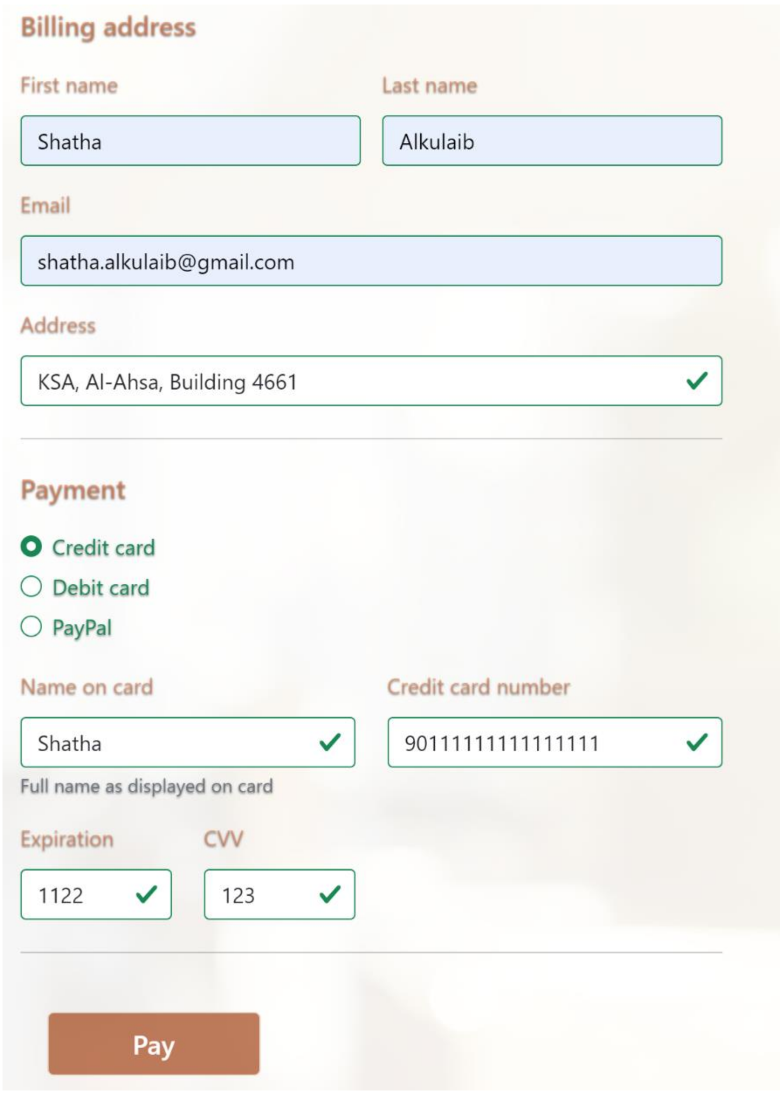
Expiration

Expiration date required

CVV

Security code required

Case 2: If the text inputs are filled correctly, the confirmation page will be opened, and the values will be stored in TheOrderItems key and PaidtotalPrice.



Billing address

First name	Last name
Shatha	Alkulaib

Email: shatha.alkulaib@gmail.com

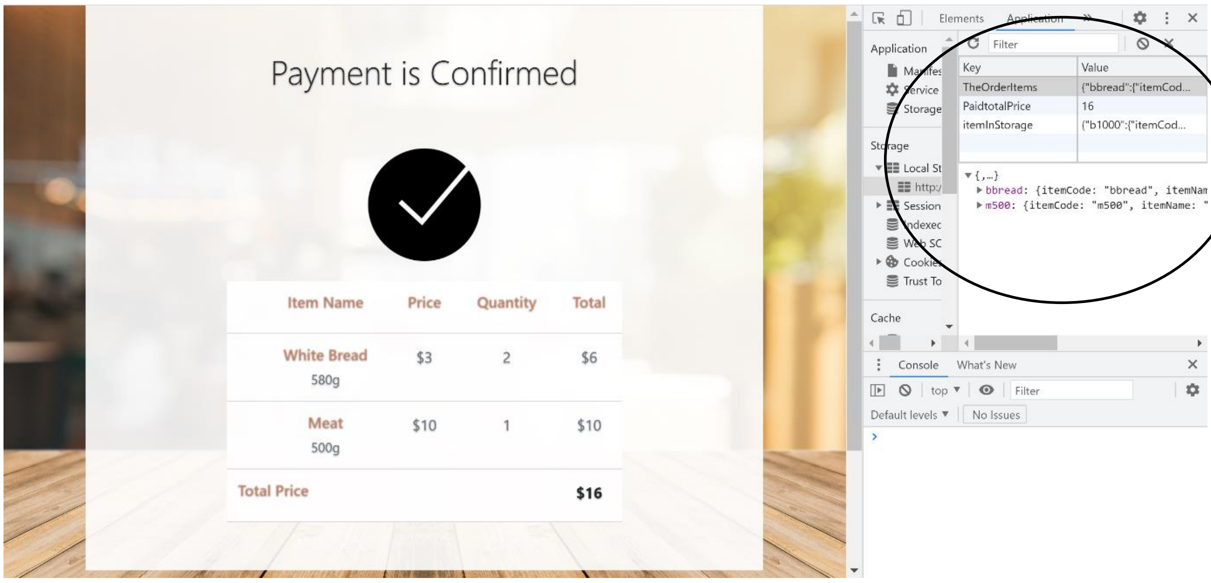
Address: KSA, Al-Ahsa, Building 4661 ✓

Payment

● Credit card
○ Debit card
○ PayPal

Name on card	Credit card number
Shatha ✓	9011111111111111 ✓
Full name as displayed on card	

Expiration	CVV
1122 ✓	123 ✓



Payment is Confirmed

Item Name	Price	Quantity	Total
White Bread 580g	$3	2	$6
Meat 500g	$10	1	$10
Total Price			$16

Key	Value
TheOrderItems	{"bbread":{"itemCod…
PaidtotalPrice	16
itemInStorage	{"b1000":{"itemCod…

```
▼{,…}
  ▶bbread: {itemCode: "bbread", itemNam
  ▶m500: {itemCode: "m500", itemName: "
```

The related functions to the Confirmation Page.

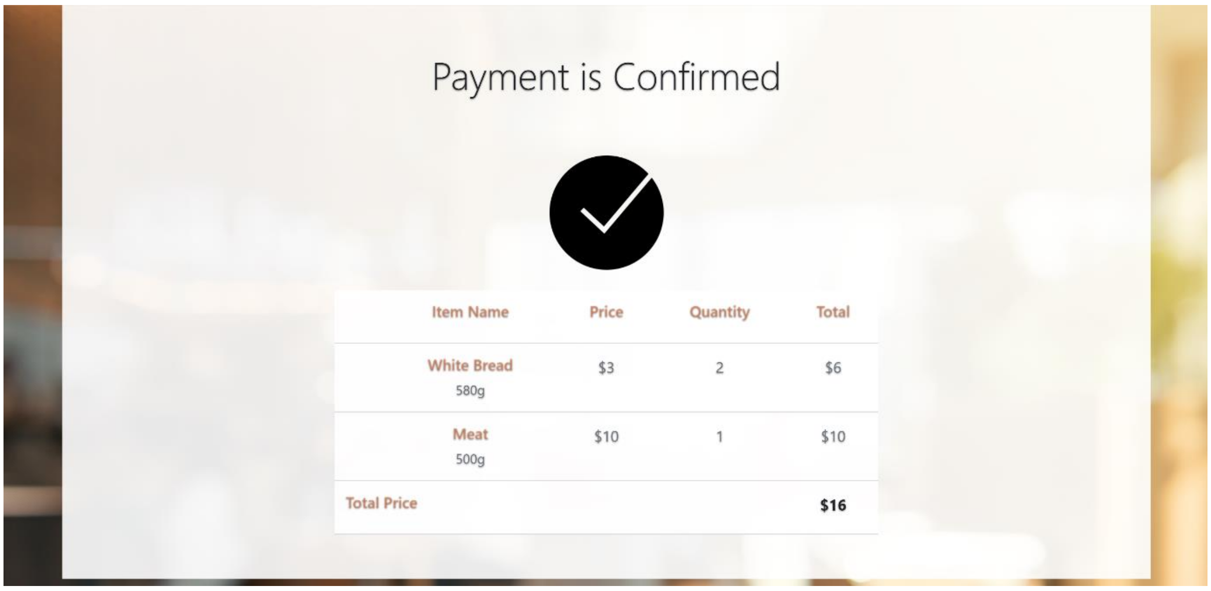
<u>ShowOrder() function</u>

It displays the contents of the TheOrderItems key on the HTML confirmation page.

```
//-------------- Confirmation Page --------------
function ShowOrder() {
    let TheOrderItems = localStorage.getItem("TheOrderItems");
    TheOrderItems = JSON.parse(TheOrderItems);
    let OrderDiv = document.querySelector(".OrderDiv");
    let PaidtotalPrice = localStorage.getItem('PaidtotalPrice');
    //check if the Confirmation Page is currently used, not other
    if (TheOrderItems && OrderDiv) {
        OrderDiv.innerHTML = '';
        Object.values(TheOrderItems).map(item => {
            OrderDiv.innerHTML += `
            <div class="rowClass row mb-5 text-center">
            <div class="col-1 themed-grid-col text-muted">
            </div>
             <div class="col-4 themed-grid-col text-muted"> <h6 class="my-
0">${item.itemName}</h6>
             <small class="text-muted">${item.ItemDetails}</small>
             </div>
             <div class="col-2 themed-grid-col text-muted"><span class="text-
muted price">$${item.price}</span></div>
             <div class="col-3 themed-grid-col text-muted">
                  <span class="text-
muted Quantity">${item.ItemQuantity}</span>
             </div>
             <div class="col-2 themed-grid-col text-muted"><span class="text-
muted Total">$${item.ItemQuantity * item.price}</span></div>
           </div>
            `
        });

         OrderDiv.innerHTML += `

        <div class="rowClass row mb-2">
        <div class="col-10 themed-grid-col text-muted text-
start"><h6>Total Price</h6></div>
        <div class="col-2 themed-grid-
col"><strong>  $${PaidtotalPrice}</strong></div>
        </div>
          `;
    }
}
//call the function when the page is loaded
ShowOrder();
```

Outputs:

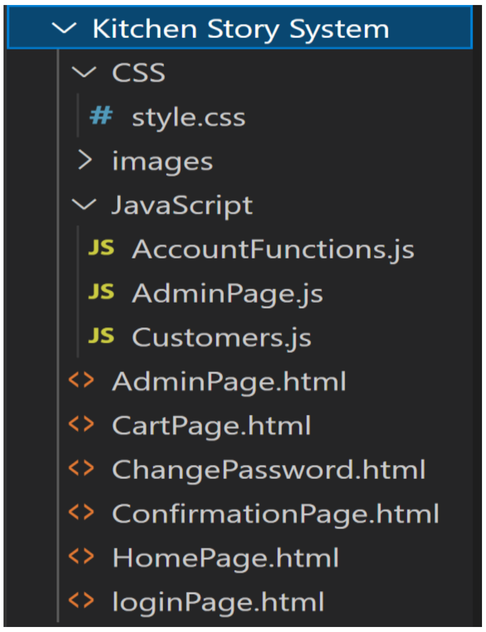
- Displaying the results of order in confirmation page.



Payment is Confirmed

Item Name	Price	Quantity	Total
White Bread 580g	$3	2	$6
Meat 500g	$10	1	$10
Total Price			**$16**

In the end, the codes of the HTML and CSS pages were written. in addition, Bootstrap was used from the getbootstrap.com website.



```
Kitchen Story System
  CSS
    style.css
  images
  JavaScript
    AccountFunctions.js
    AdminPage.js
    Customers.js
  AdminPage.html
  CartPage.html
  ChangePassword.html
  ConfirmationPage.html
  HomePage.html
  loginPage.html
```