WEB TECHNOLOGY LAB

Assignment- 7

Name- Parminder Singh Roll No- 22CS2030 Branch- IDD

T1. Develop prototype 3 continuing with the last lab. Confirm that the app now remembers your list even after a page refresh.

```
<!-- index.html -->
<!DOCTYPE html>
<html lang="en">

<head>
    <script src="data.js"></script>
    <script src="view.js"></script>
    <script src="controller.js"></script>

    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Shopping List MVC</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: black;
            text-align: center;
        }

        h1 {
            color: gold;
        }

        #itemInput {
            color: black;
            background-color: aqua;
        }

        button {
            border-radius: 50%;
        }

        button:hover {
            cursor: grab;
        }

        #shoppingList {
            color: blue;
            font-style: italic;
```

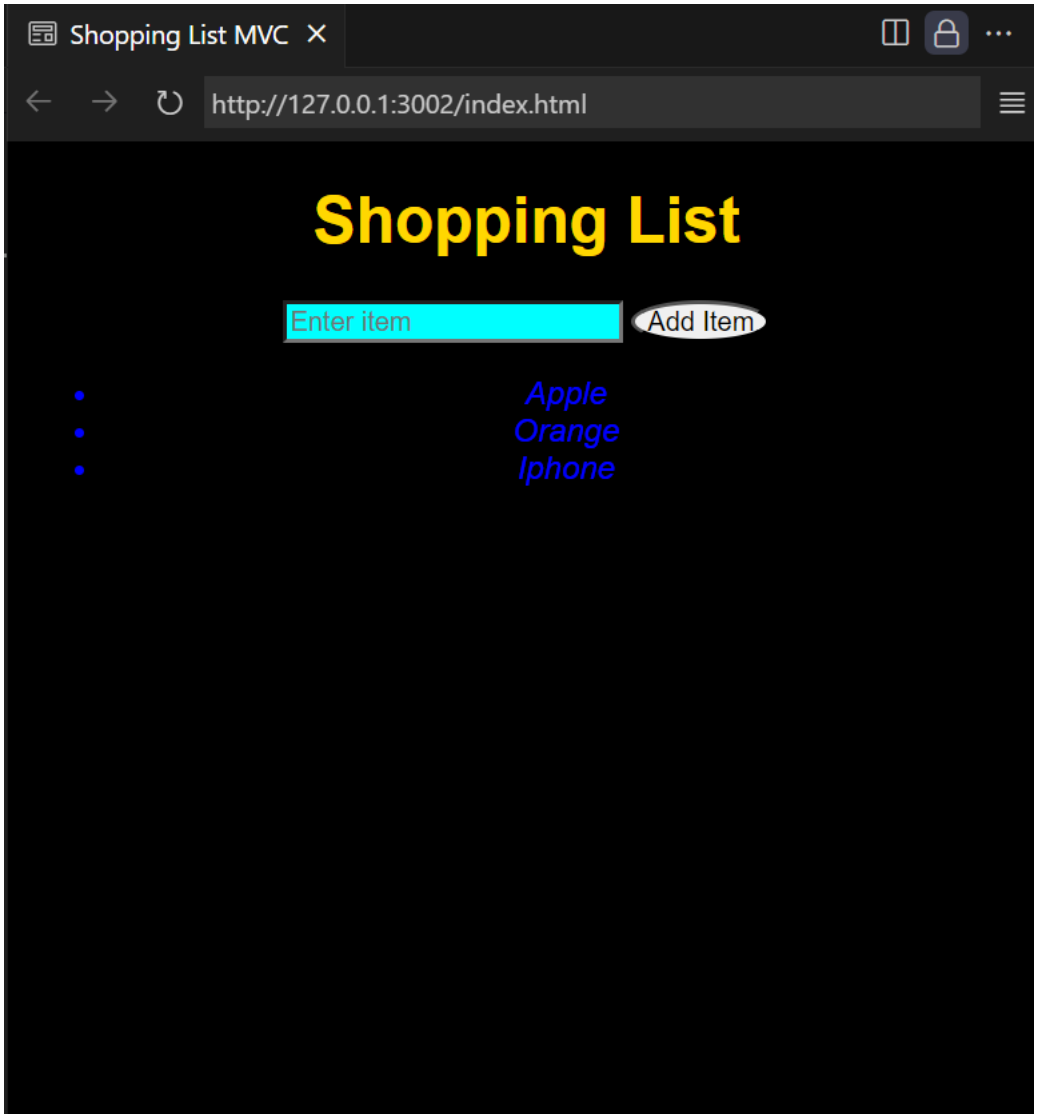
```
        }
    </style>
</head>

<body>
    <h1>Shopping List</h1>

    <div>
        <input type="text" id="itemInput" placeholder="Enter item">
        <button>Add Item</button>
    </div>

    <ul id="shoppingList"></ul>

    <script>
        const model = new ShoppingListModel();
        const view = new ShoppingListView();
        const controller = new ShoppingListController(model, view);

        model.loadItemsFromStorage();
        controller.updateView();
    </script>
</body>

</html>
```

```
// controller.js
class ShoppingListController {
    constructor(model, view) {
        this.model = model;
        this.view = view;
        this.updateView();

        this.view.addItemButton.addEventListener('click', () =>
this.addItem());
    }

    addItem() {
        const itemName = this.view.getItemInputValue();

        if (itemName !== '') {
            this.model.addItem(itemName);
            this.updateView();
            this.view.clearItemInput();
        }
```

```
    }

    updateView() {
        const items = this.model.getItems();
        this.view.updateItemList(items);
    }

    bindRemoveItem() {
        this.view.bindRemoveItem(index => {
            this.model.removeItem(index);
            this.updateView();
        });
    }
}
```

```
// data.js
class ShoppingListModel {
    constructor() {
        this.items = [];
    }

    addItem(itemName) {
        this.items.push(itemName);
        this.saveItemsToStorage();
    }

    removeItem(index) {
        this.items.splice(index, 1);
        this.saveItemsToStorage();
    }

    getItems() {
        return this.items;
    }

    loadItemsFromStorage() {
        const storedItems = JSON.parse(localStorage.getItem("shoppingList"))
|| [];
        this.items = storedItems;
    }

    saveItemsToStorage() {
        localStorage.setItem("shoppingList", JSON.stringify(this.items));
    }
}
```

```
// view.js
class ShoppingListView {
    constructor() {
        this.itemInput = document.getElementById('itemInput');
        this.shoppingList = document.getElementById('shoppingList');
        this.addItemButton = document.querySelector('button');

        this.addItemButton.addEventListener('click', () =>
this.controller.addItem());
    }

    bindRemoveItem(handler) {
        this.shoppingList.addEventListener('click', (event) => {
            if (event.target.tagName === 'LI') {
                const index =
Array.from(this.shoppingList.children).indexOf(event.target);
                handler(index);
            }
        });
    }

    updateItemList(items) {
        this.shoppingList.innerHTML = "";
        items.forEach(item => {
            const listItem = document.createElement("li");
            listItem.textContent = item;
            this.shoppingList.appendChild(listItem);
        });
    }

    getItemInputValue() {
        return this.itemInput.value.trim();
    }

    clearItemInput() {
        this.itemInput.value = '';
    }
}
```



T2. Create a local storage that saves the number of times you have accessed the page and displays it.

```
<!-- index.html -->
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Shopping List MVC</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: black;
```

```
            text-align: center;
        }

        h1 {
            color: gold;
        }

        #itemInput {
            color: black;
            background-color: aqua;
        }

        button {
            border-radius: 50%;
        }

        button:hover {
            cursor: grab;
        }

        #shoppingList {
            color: blue;
            font-style: italic;
        }
    </style>
</head>

<body>
    <h1>Shopping List</h1>

    <div>
        <input type="text" id="itemInput" placeholder="Enter item">
        <button>Add Item</button>
    </div>

    <ul id="shoppingList"></ul>

    <p id="accessCount">Page accessed: <span id="count"></span> times</p>

    <script src="data.js"></script>
    <script src="view.js"></script>
    <script src="controller.js"></script>

    <script>
        const model = new ShoppingListModel();
        const view = new ShoppingListView();
        const controller = new ShoppingListController(model, view);
```

```
        // Load items and access count from localStorage on page load
        model.loadItemsFromStorage();
        model.loadAccessCountFromStorage();
        controller.updateView();

        // Update access count and display
        model.incrementAccessCount();
        controller.updateAccessCount();
    </script>
</body>

</html>
```

```
// controller.js
class ShoppingListController {
    constructor(model, view) {
        this.model = model;
        this.view = view;
        this.view.setController(this);
        this.updateView();
        this.updateAccessCount(); // Add this line to update access count on
initialization

        this.view.addItemButton.addEventListener('click', () =>
this.addItem());
    }

    addItem() {
        const itemName = this.view.getItemInputValue();

        if (itemName !== '') {
            this.model.addItem(itemName);
            this.updateView();
            this.view.clearItemInput();
        }
    }

    updateView() {
        const items = this.model.getItems();
        this.view.updateItemList(items);
    }

    bindRemoveItem() {
        this.view.bindRemoveItem(index => {
            this.model.removeItem(index);
```

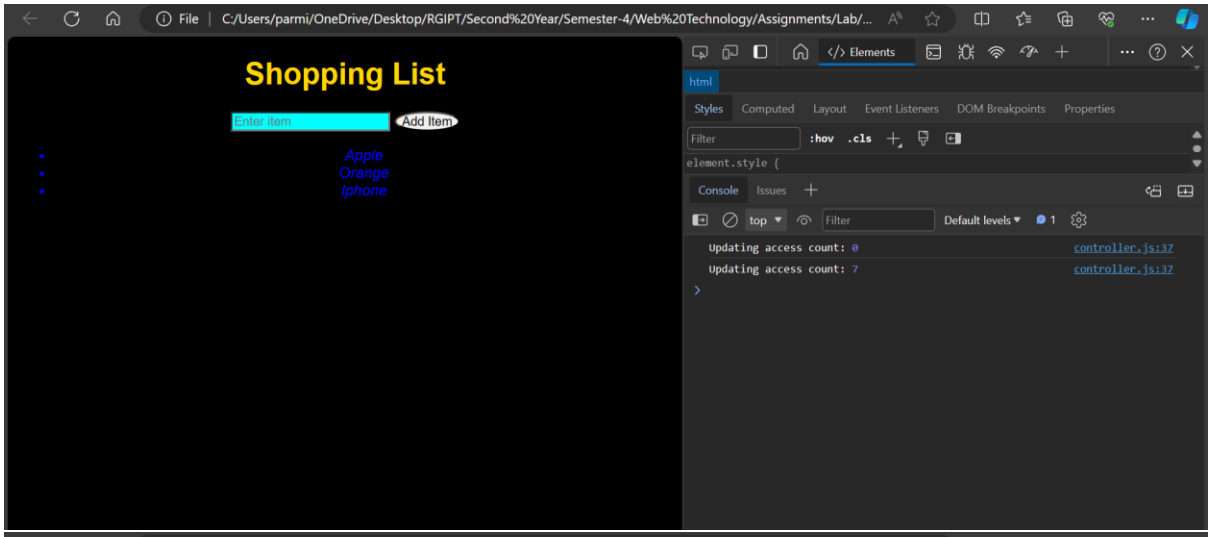
```
            this.updateView();
        });
    }

    updateAccessCount() {
        const accessCount = this.model.getAccessCount();
        console.log('Updating access count:', accessCount);
        this.view.updateAccessCount(accessCount);
    }

}
```

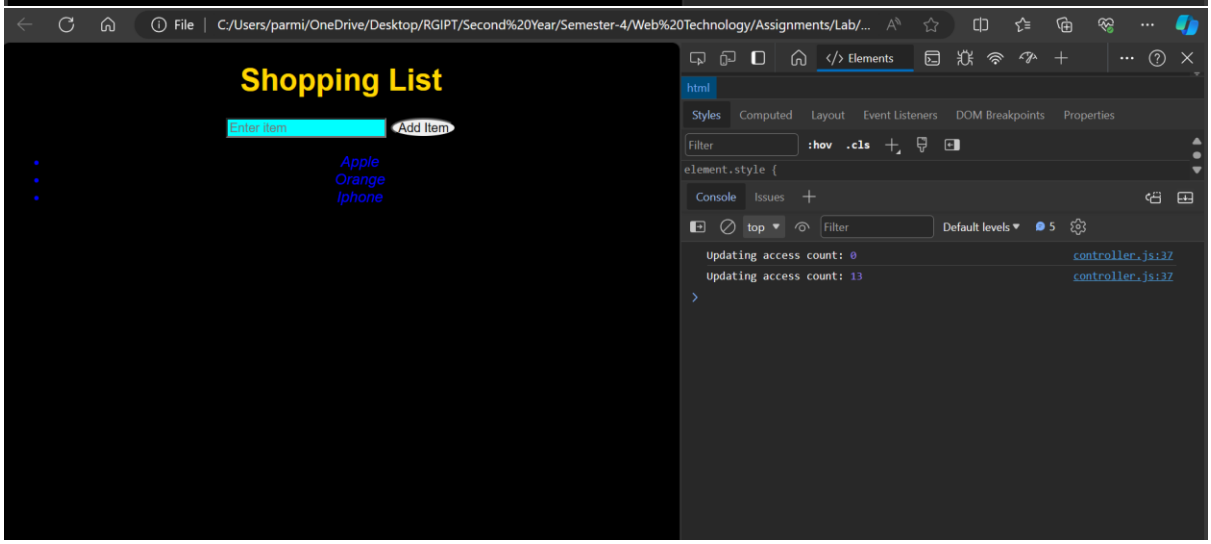
```
// data.js
class ShoppingListModel {
    constructor() {
        this.items = [];
        this.accessCount = 0;
    }

    addItem(itemName) {
        this.items.push(itemName);
        this.saveItemsToStorage();
    }

    removeItem(index) {
        this.items.splice(index, 1);
        this.saveItemsToStorage();
    }

    getItems() {
        return this.items;
    }

    loadItemsFromStorage() {
        const storedItems = JSON.parse(localStorage.getItem("shoppingList"))
|| [];
        this.items = storedItems;
    }

    saveItemsToStorage() {
        localStorage.setItem("shoppingList", JSON.stringify(this.items));
    }

    loadAccessCountFromStorage() {
        this.accessCount = parseInt(localStorage.getItem("accessCount")) || 0;
```

```
    }

    incrementAccessCount() {
        this.accessCount++;
        this.saveAccessCountToStorage();
    }

    getAccessCount() {
        return this.accessCount;
    }

    saveAccessCountToStorage() {
        localStorage.setItem("accessCount", this.accessCount.toString());
    }
}
```

```
// view.js
class ShoppingListView {
    constructor() {
        this.itemInput = document.getElementById('itemInput');
        this.shoppingList = document.getElementById('shoppingList');
        this.addItemButton = document.querySelector('button');
        this.accessCountDisplay = document.getElementById('count');
        this.controller = null;

        this.addItemButton.addEventListener('click', () =>
this.controller.addItem());
    }

    bindRemoveItem(handler) {
        this.shoppingList.addEventListener('click', (event) => {
            if (event.target.tagName === 'LI') {
                const index =
Array.from(this.shoppingList.children).indexOf(event.target);
                handler(index);
            }
        });
    }

    updateItemList(items) {
        this.shoppingList.innerHTML = "";
        items.forEach(item => {
            const listItem = document.createElement("li");
            listItem.textContent = item;
            this.shoppingList.appendChild(listItem);
```

```
        });
    }

    getItemInputValue() {
        return this.itemInput.value.trim();
    }

    clearItemInput() {
        this.itemInput.value = '';
    }

    updateAccessCount(count) {
        this.accessCountDisplay.textContent = count;
    }

    setController(controller) {
        this.controller = controller;
    }
}
```


File | C:/Users/parmi/OneDrive/Desktop/RGIPT/Second%20Year/Semester-4/Web%20Technology/Assignments/Lab/...
Shopping List
Enter item
Add Item
Apple
Orange
Iphone
Elements
html
Styles
Computed
Layout
Event Listeners
DOM Breakpoints
Properties
Filter
:hov
.cls
element.style {
Console
Issues
top
Filter
Default levels
1
Updating access count: 0
controller.js:37
Updating access count: 7
controller.js:37


File | C:/Users/parmi/OneDrive/Desktop/RGIPT/Second%20Year/Semester-4/Web%20Technology/Assignments/Lab/...
Shopping List
Enter item
Add Item
Apple
Orange
Iphone
Elements
html
Styles
Computed
Layout
Event Listeners
DOM Breakpoints
Properties
Filter
:hov
.cls
element.style {
Console
Issues
top
Filter
Default levels
5
Updating access count: 0
controller.js:37
Updating access count: 13
controller.js:37