

Web Technology

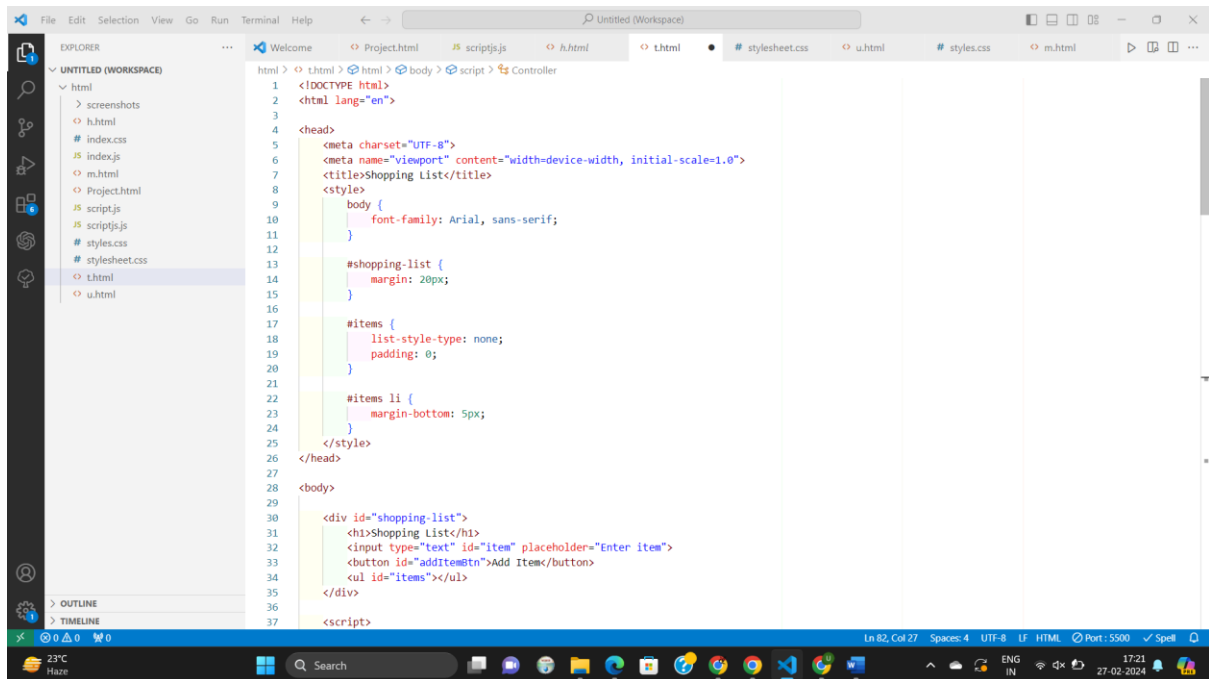
Lab Assignment –

Name :- Utsav Shingala

Roll No. :- 22CS3065

Branch :- CSE

Task 1)



The screenshot displays a web development IDE with a file explorer on the left, a central code editor, and a terminal at the bottom. The file explorer shows a project structure with files like index.css, index.js, Project.html, script.js, styles.css, t.html, and u.html. The code editor is open to a file named 't.html', which contains the following code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     <meta charset="UTF-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Shopping List</title>
8     <style>
9         body {
10             font-family: Arial, sans-serif;
11         }
12
13         #shopping-list {
14             margin: 20px;
15         }
16
17         #items {
18             list-style-type: none;
19             padding: 0;
20         }
21
22         #items li {
23             margin-bottom: 5px;
24         }
25     </style>
26 </head>
27
28 <body>
29
30     <div id="shopping-list">
31         <h1>Shopping List</h1>
32         <input type="text" id="item" placeholder="Enter item">
33         <button id="addItemBtn">Add Item</button>
34         <ul id="items"></ul>
35     </div>
36
37 </body>
38 </html>
```

The code defines a simple shopping list interface with a text input, an 'Add Item' button, and an unordered list. The CSS styles the list items and the container. The terminal at the bottom shows the status bar with 'Ln 82, Col 27', 'Spaces: 4', 'UTF-8', 'LF', 'HTML', 'Port: 5500', and 'Spell'.

This screenshot shows the VS Code editor with a workspace named 'untitled (workspace)'. The Explorer sidebar on the left shows a file structure with folders 'screenshots' and 'html', and files 'h.html', 'index.css', 'index.js', 'm.html', 'Project.html', 'script.js', 'stylesheet.css', 't.html', and 'u.html'. The main editor area displays the 'Controller' file, which contains two classes: 'Model' and 'View'. The 'Model' class has methods for initializing items from localStorage, adding items, saving to localStorage, and retrieving items. The 'View' class handles DOM interactions, including getting the items list, item input, and add button, and methods for rendering the list and clearing the input.

```
37 <script>
38 class Model {
39   constructor() {
40     this.items = JSON.parse(localStorage.getItem('shoppingList')) || [];
41   }
42
43   addItem(item) {
44     this.items.push(item);
45     this.save();
46   }
47
48   getItems() {
49     return this.items;
50   }
51
52   save() {
53     localStorage.setItem('shoppingList', JSON.stringify(this.items));
54   }
55 }
56
57 class View {
58   constructor() {
59     this.itemsList = document.getElementById('items');
60     this.itemInput = document.getElementById('item');
61     this.addItemBtn = document.getElementById('addItemBtn');
62   }
63
64   getNewItem() {
65     return this.itemInput.value.trim();
66   }
67
68   clearInput() {
69     this.itemInput.value = '';
70   }
71
72   renderItems(items) {
73     this.itemsList.innerHTML = '';
```

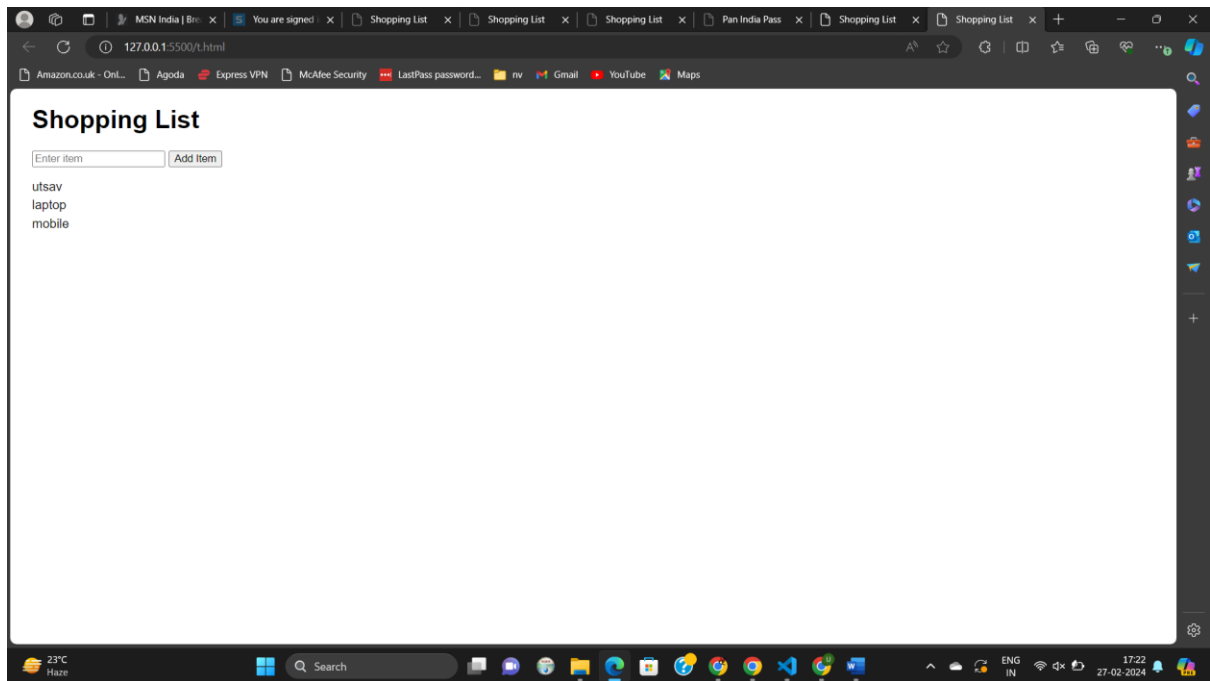
This screenshot shows the continuation of the 'Controller' file in the VS Code editor. It includes the 'renderItems' method of the 'View' class, which uses 'items.forEach' to create and append list items to the DOM. Below this, the 'Controller' class is defined, which takes 'model' and 'view' as arguments in its constructor. It sets up an event listener for the 'addItem' button and calls 'initialRender'. The 'initialRender' method calls 'renderItems' with the current model's items. The 'addItem' method gets a new item from the view, adds it to the model, renders the updated list, and clears the input.

```
70   }
71
72   renderItems(items) {
73     this.itemsList.innerHTML = '';
74     items.forEach(item => {
75       const li = document.createElement('li');
76       li.textContent = item;
77       this.itemsList.appendChild(li);
78     });
79   }
80 }
81
82 class Controller {
83   constructor(model, view) {
84     this.model = model;
85     this.view = view;
86
87     this.view.addItemBtn.addEventListener('click', () => this.addItem());
88     this.initialRender();
89   }
90
91   initialRender() {
92     this.view.renderItems(this.model.getItems());
93   }
94
95   addItem() {
96     const newItem = this.view.getNewItem();
97     if (newItem !== '') {
98       this.model.addItem(newItem);
99       this.view.renderItems(this.model.getItems());
100       this.view.clearInput();
101     }
102   }
103 }
104
105
```

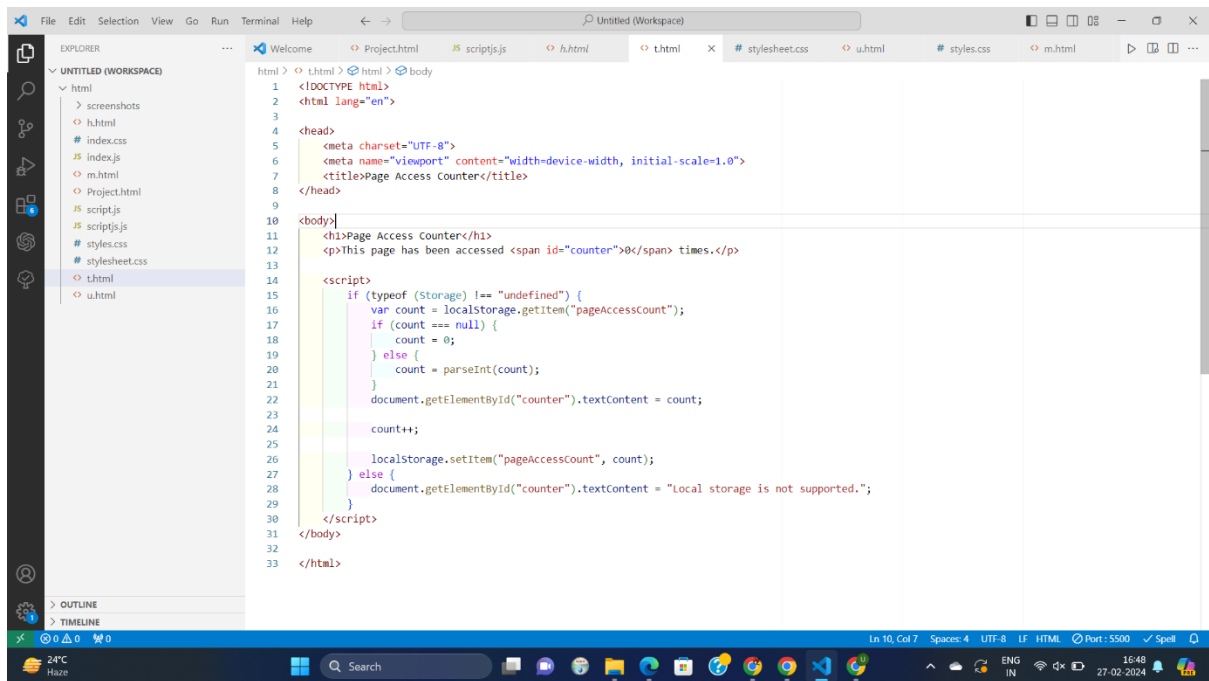
The screenshot shows the Visual Studio Code editor with a workspace named 'untitled (workspace)'. The Explorer sidebar on the left shows a file structure with folders 'html' and 'scriptjs', and files 'index.js', 'Project.html', 'scriptjs.js', 'styles.css', 'stylesheet.css', 't.html', and 'u.html'. The main editor area displays the content of 't.html', which includes a script tag for a Controller. The code is as follows:

```
87 this.view.addItemBtn.addEventListener('click', () => this.addItem());
88
89 this.initialRender();
90
91
92 initialRender() {
93   this.view.renderItems(this.model.getItems());
94 }
95
96 addItem() {
97   const newItem = this.view.getNewItem();
98   if (newItem !== '') {
99     this.model.addItem(newItem);
100    this.view.renderItems(this.model.getItems());
101    this.view.clearInput();
102  }
103 }
104
105
106 const model = new Model();
107 const view = new View();
108 const controller = new Controller(model, view);
109 </script>
110
111 </body>
112
113 </html>
```

The status bar at the bottom indicates the current position is Line 82, Column 27, with 4 spaces, UTF-8 encoding, LF line endings, and a port of 5500. The system tray shows a temperature of 23°C, a 'Haze' weather condition, and the date 27-02-2024.



Task 2)



The screenshot shows the Visual Studio Code editor with a workspace named 'Untitled (Workspace)'. The Explorer panel on the left shows a file structure with 'html' as the selected folder. The main editor area displays the code for 't.html'. The code is as follows:

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Page Access Counter</title>
8 </head>
9
10 <body>
11   <h2>Page Access Counter</h2>
12   <p>This page has been accessed <span id="counter">0</span> times.</p>
13
14   <script>
15     if (typeof (Storage) !== "undefined") {
16       var count = localStorage.getItem("pageAccessCount");
17       if (count === null) {
18         count = 0;
19       } else {
20         count = parseInt(count);
21       }
22       document.getElementById("counter").textContent = count;
23
24       count++;
25
26       localStorage.setItem("pageAccessCount", count);
27     } else {
28       document.getElementById("counter").textContent = "Local storage is not supported.";
29     }
30   }
31 </script>
32 </body>
33 </html>
```

