ASSET MANAGEMENT TOOL USING JS

Undertaken at

SYMBIOSYS TECHNOLOGIES

By

DUBBAKA JHANSI

Under the esteemed guidance of

D.Sudheer Babu

Manager-IT

Symbiosys Technologies

Rushikonda ,Visakhapatnam.

DEPARTMENT OF INFORMATION TECHNOLOGY

MVGR COLLEGE OF ENGINEERING(A)

VIZIANAGARAM-535003,(2021-25)

ACKNOWLEDGEMENT

I wish to express my heartful thanks to all the people who have made it possible for me to do this project and to present this project.

At the outset I would like to express my deepest gratitude to the for Symbiosys Technologies giving me a valuable opportunity to do this project.

I am grateful to **Mr. D.Sudheer Babu** Manager-IT,Symbiosys Technologies, Visakhapatnam who had guided me to undertake the project work and spend his valuable time in completing this project work.

I am thankful to one and all those who have directly and indirectly helped me in finishing this project.

DECLARATION

The project report entitled “Asset Management Tool Using JS” submitted by me(DUBBAKA JHANSI),is a part of the project training done by me at Symbiosys Technologies ,under the esteemed guidance of **Mr. D.Sudheer Babu** Manager-IT

.

(DUBBAKA JHANSI)

CERTIFICATE

This is to certify that the project entitled “Asset Management Tool ” is bonafide record of work done by DUBBAKA JHANSI of Department of information technology ,MVGR COLLEGE OF ENGINEERING(A), VIZIANAGARAM-535003.She did this project under my guidance and supervision.

D.Sudheer Babu

Manager-IT

PROBLEM STATEMENT

- **Extracting and Displaying Information from a Belarc Advisor HTML Report Using JavaScript**

  The problem entails developing a JavaScript application to parse and present detailed system information extracted from an HTML report generated by Belarc Advisor. The goal is to efficiently navigate the nested HTML structure of the report to extract key data points such as hardware specifications, software inventory, network details, and security status. This information will then be formatted into a clear and organized display, likely using an HTML table, to facilitate easy comprehension and analysis by users. The application must handle various complexities of HTML parsing and ensure accurate extraction and presentation of critical system-related information.

REQUIREMENTS

- **Hardware Requirements:**
  - RAM: At least 2GB.
  - Storage: At least 10 GB
- **Software Requirements:**
  - Latest OS
  - Browser(Latest Versions Only)
  - Excel Sheets

Technologies:

Java Script:

JavaScript is a scripting language that enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else. Parsing in JavaScript involves analyzing structured data, often in formats like JSON or XML, to extract meaningful information. The JavaScript code utilizes DOM manipulation to parse HTML content for system details. It locates captions and table headings using querySelectorAll and extracts their text content with textContent. Conditional statements handle data extraction based on specific captions, like system model and processor. Complex parsing involves splitting and filtering HTML elements. Extracted details are then displayed in a table format. This approach efficiently processes HTML content for systematic presentation of system specifications.

Hyper Text Markup Language:

HTML is the markup language that we use to structure and give meaning to our web content, for example defining paragraphs, headings, and data tables, or embedding images and videos in the page.

IMPLEMENTATION

Step-1: Designing a form interface that allows users to upload files

Code:

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Process Files and Export to Excel</title>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.16.9/xlsx.full.min.js"></script>
 <style>
 body::before {
  content: "";
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-image: url("sym.jpg");
  background-size: 1000px 300px;
  background-repeat: no-repeat;
  background-attachment: fixed;
  background-position: center;
  opacity: 0.3; /* Adjust the transparency level */
  z-index: -1;
```

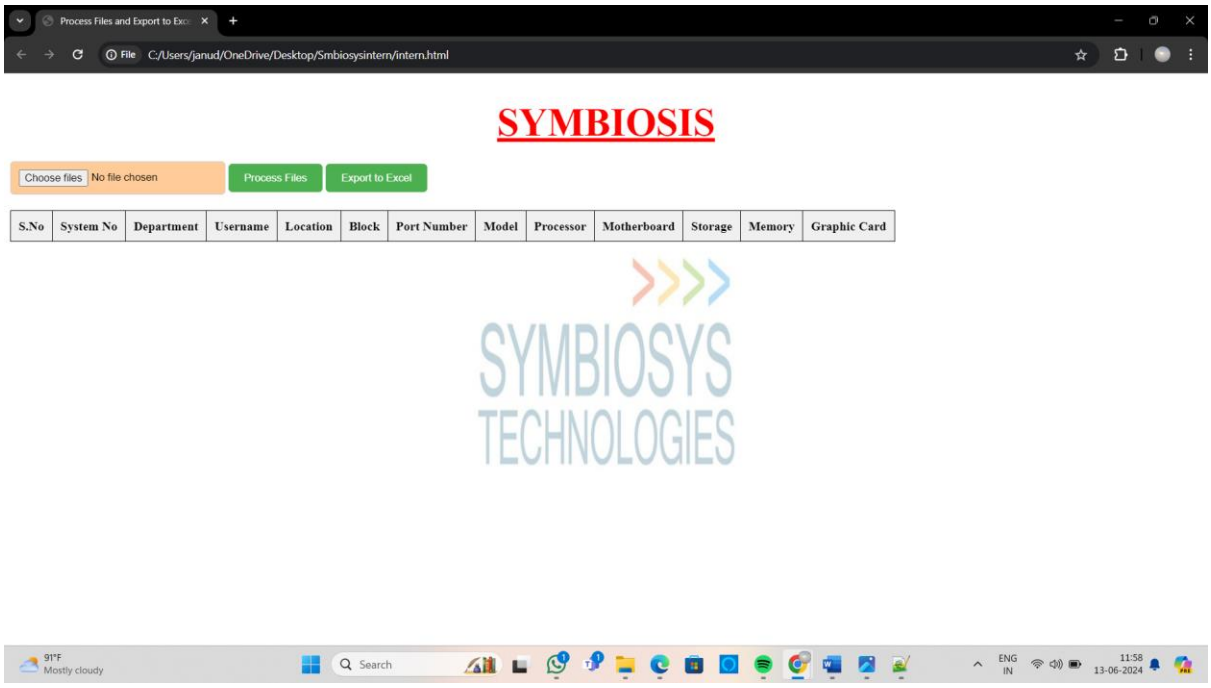
```
}

h1 {
 color: red;
 text-align: center;
 margin-bottom: 20px;
 font-size:50px;
}

table, th, td {
 border: 1px solid black;
 border-collapse: collapse;
 padding: 10px;
  background-color:transparent;
}

th {
 background-color: #f0f0f0;
}

td {
 text-align: center;
}

.spaced {
 margin-bottom: 20px;
}
```

```
.file-input-container {
 margin-bottom: 20px;
}

.file-input-container input[type="file"] {
 padding: 10px;
 border: 1px solid #ccc;
 border-radius: 5px;
 background-color:#ffcc99;
}

.file-input-container button {
 background-color: #4CAF50;
 color: #fff;
 padding: 10px 20px;
 border: none;
 border-radius: 5px;
 cursor: pointer;
}

.file-input-container button:hover {
 background-color: #3e8e41;
}
</style>
</head>
<body>
```

```
  <h1><u>SYMBIOSIS</u></h1>
  <div class="file-input-container spaced">
    <input type="file" id="fileInput" multiple>
    <button onclick="processFiles()">Process Files</button>

  <button onclick="exportTableToExcel()">Export to Excel</button>
</div>
  <table id="dataTable">
    <thead>
      <tr>
        <th>S.No</th>
        <th>System No</th>
        <th>Department</th>
        <th>Username</th>
        <th>Location</th>
        <th>Block</th>
        <th>Port Number</th>
        <th>Model</th>
        <th>Processor</th>
        <th>Motherboard</th>
        <th>Storage</th>
        <th>Memory</th>
        <th>Graphic Card</th>
      </tr>
    </thead>
    <tbody>
      <!-- Data will be inserted here -->
```

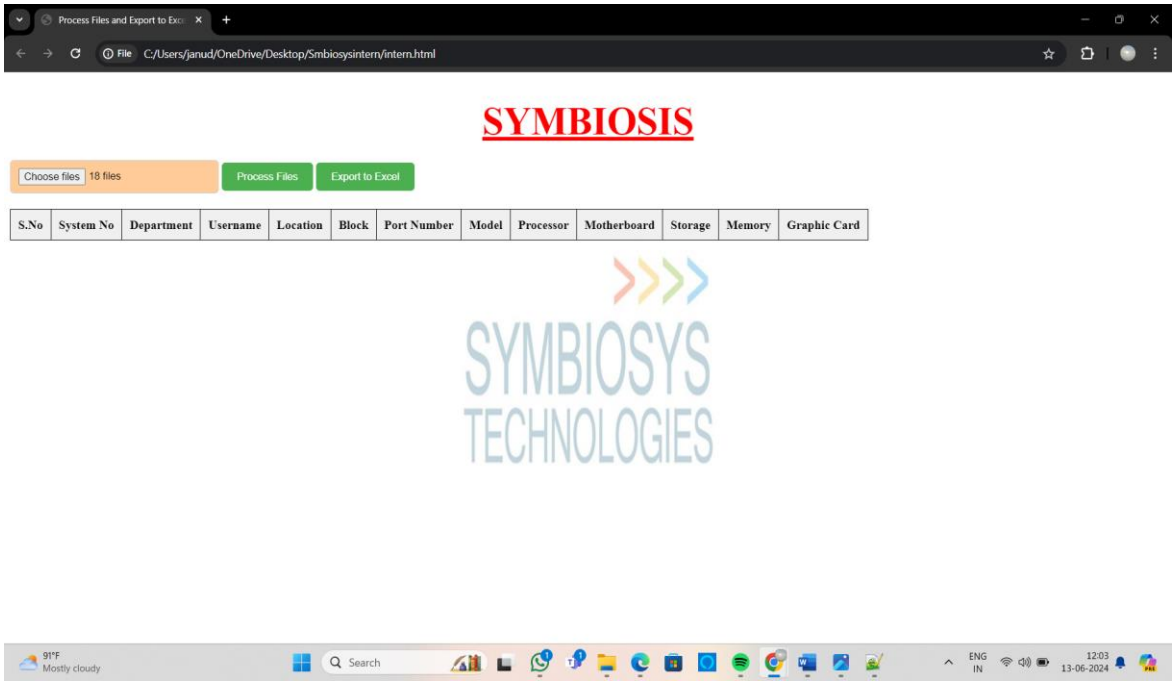
```
        </tbody>
      </table>
    </body>
    </html>
```

Outcome:



Step-2: Allowing users to upload files directly within the form.

Outcome:



Step-3: After Uploading the files we have to click on process files.

Code:

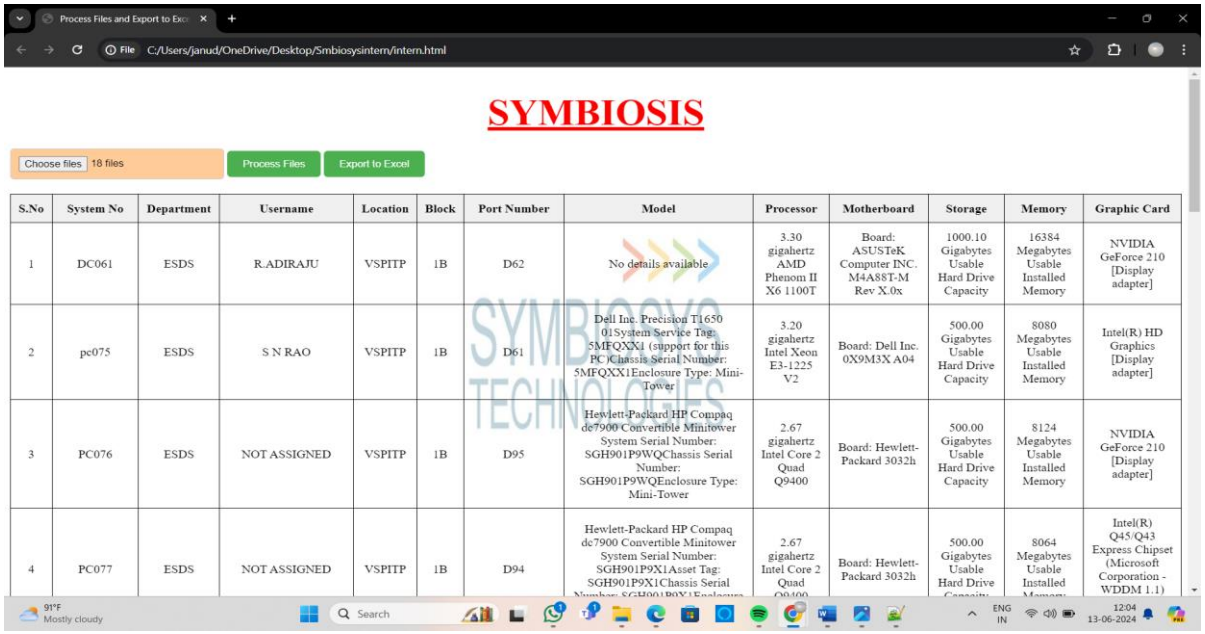
```
<script>
        async function processFiles() {
            const fileInput = document.getElementById('fileInput');
            const files = fileInput.files;
            const dataTable = document.getElementById('dataTable').getElementsByTagName('tbody')[0];

            for (let i = 0; i < files.length; i++) {
                const file = files[i];
                const content = await file.text();
                const parser = new DOMParser();
                const doc = parser.parseFromString(content, 'text/html');
        const extractValue = (captionText) => {
                    const captionElement = Array.from(doc.querySelectorAll('caption')).find(caption => caption.textContent.trim().includes(captionText));
                    if (captionElement) {
                        const td = captionElement.nextElementSibling?.querySelector('td');
                        if (td) {
                            const parts = td.innerHTML.split('<br>');
                            return parts.length > 0 ? parts[0].trim() : 'N/A';
                        }
                    }
                    return 'N/A';
                };
```

```
        const captions = doc.querySelectorAll('caption');
        let model = extractValue(' System Model');
        let processor = extractValue('Processor a');
        let motherboard = extractValue('Main Circuit Board b');
        let storage = extractValue('Drives');
        let memory = extractValue('Memory');
        let graphicCard = extractValue('Dsiplay ');


    captions.forEach(caption => {
            switch (caption.textContent.trim()) {
                case 'System Model':
                    model = caption.nextElementSibling.textContent.trim();
                    break;
                case 'Display':
                    let tdElement = caption.nextElementSibling.querySelector('td');
        let graphicCardContent = "";
        let nodes = tdElement.childNodes;

        for (let node of nodes) {
            if (node.nodeName === "BR") {
                break;
            }
            graphicCardContent += node.textContent || node.nodeValue;
        }

        graphicCard = graphicCardContent.trim();
```

```
        break;
            }
        });
let systemNo = '';
        let username = '';
        let dept = '';
        let location = '';
        let block = '';
        let portNumber = 'Port ';

        const thElements = doc.querySelectorAll('th');
        thElements.forEach(th => {
            switch (th.textContent.trim()) {
                case 'Computer Name:':
                    systemNo = th.nextElementSibling.textContent.trim();
                    break;
                case 'Windows Logon:':
                    username = th.nextElementSibling.textContent.trim();
                    break;
                case 'Dept:':
                    dept = th.nextElementSibling.textContent.trim();
                    break;
                case 'Location:':
                    location = th.nextElementSibling.textContent.trim();
                    break;
                case 'Block:':
                    block = th.nextElementSibling.textContent.trim();
```

```
                    break;
                case 'Port Number:':
                    portNumber = th.nextElementSibling.textContent.trim();
                    break;
            }
        });


        const newRow = dataTable.insertRow();
        newRow.insertCell(0).innerText = i + 1;
        newRow.insertCell(1).innerText = systemNo;
        newRow.insertCell(2).innerText = dept;
        newRow.insertCell(3).innerText = username;
        newRow.insertCell(4).innerText = location;
        newRow.insertCell(5).innerText = block;
        newRow.insertCell(6).innerText = portNumber;
        newRow.insertCell(7).innerText = model;
        newRow.insertCell(8).innerText = processor;
        newRow.insertCell(9).innerText = motherboard;
        newRow.insertCell(10).innerText = storage;
        newRow.insertCell(11).innerText = memory;
        newRow.insertCell(12).innerText = graphicCard;
    }
}
</script>
```

Outcome:



S.No	System No	Department	Username	Location	Block	Port Number	Model	Processor	Motherboard	Storage	Memory	Graphic Card
1	DC061	ESDS	R.ADIRAJU	VSPITP	1B	D62	No details available	3.30 gigahertz AMD Phenom II X6 1100T	Board: ASUSTeK Computer INC. M4A88T-M Rev X.0x	1000.10 Gigabytes Usable Hard Drive Capacity	16384 Megabytes Usable Installed Memory	NVIDIA GeForce 210 [Display adapter]
2	pc075	ESDS	S N RAO	VSPITP	1B	D61	Dell Inc. Precision T1650 01System Service Tag: 5MFQXX1 (support for this PC)Chassis Serial Number: 5MFQXX1Enclosure Type: Mini-Tower	3.20 gigahertz Intel Xeon E3-1225 V2	Board: Dell Inc. 0X9M3X A04	500.00 Gigabytes Usable Hard Drive Capacity	8080 Megabytes Usable Installed Memory	Intel(R) HD Graphics [Display adapter]
3	PC076	ESDS	NOT ASSIGNED	VSPITP	1B	D95	Hewlett-Packard HP Compaq dc7900 Convertible Minitower System Serial Number: SGH901P9WQChassis Serial Number: SGH901P9WQEnclosure Type: Mini-Tower	2.67 gigahertz Intel Core 2 Quad Q9400	Board: Hewlett-Packard 3032h	500.00 Gigabytes Usable Hard Drive Capacity	8124 Megabytes Usable Installed Memory	NVIDIA GeForce 210 [Display adapter]
4	PC077	ESDS	NOT ASSIGNED	VSPITP	1B	D94	Hewlett-Packard HP Compaq dc7900 Convertible Minitower System Serial Number: SGH901P9X1Asset Tag: SGH901P9X1Chassis Serial Number: SGH901P9X1Enclosure	2.67 gigahertz Intel Core 2 Quad Q9400	Board: Hewlett-Packard 3032h	500.00 Gigabytes Usable Hard Drive Capacity	8064 Megabytes Usable Installed Memory	Intel(R) Q45/Q43 Express Chipset (Microsoft Corporation - WDDM 1.1)

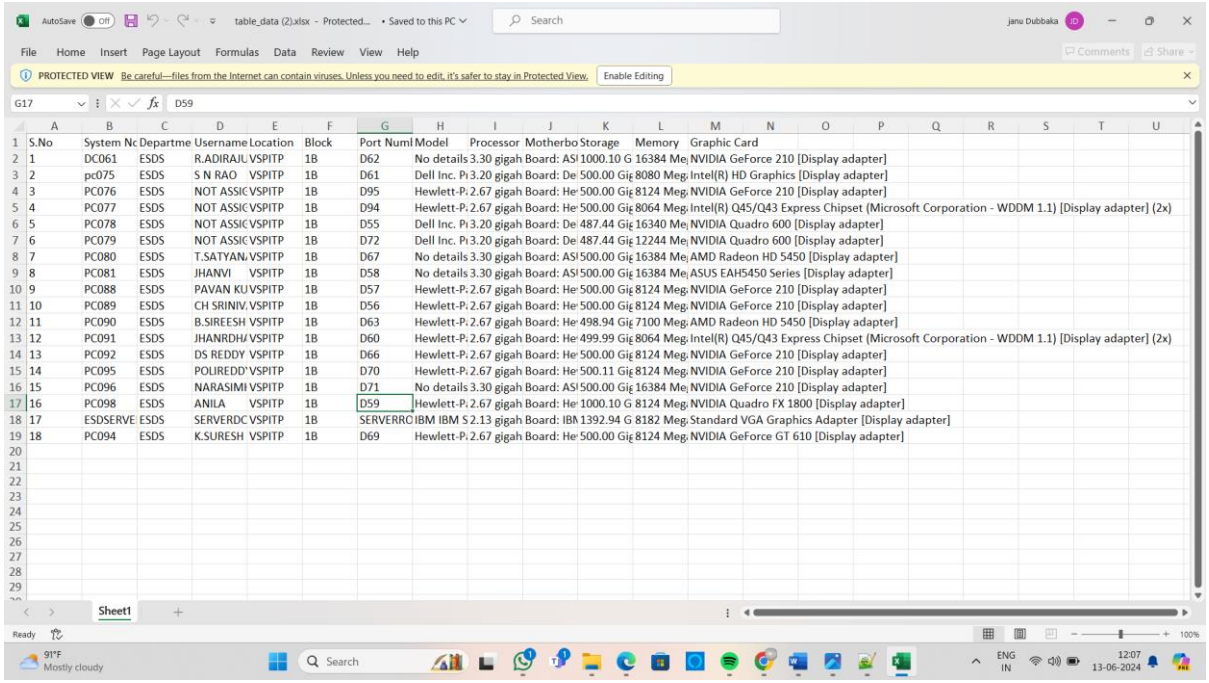
Step-4: If we want to export the table into excel then click on export to excel.

Code:

```
<script>
 function exportTableToExcel() {
        const table = document.getElementById("dataTable");
        const rows = table.querySelectorAll("tr");

        // Extract table data
        const data = [];
        rows.forEach(row => {
            const rowData = [];
            const cells = row.querySelectorAll("th, td");
            cells.forEach(cell => {
                rowData.push(cell.innerText);
            });
            data.push(rowData);
        });

        // Create a new workbook and add the data
        const wb = XLSX.utils.book_new();
        const ws = XLSX.utils.aoa_to_sheet(data);

        // Append the worksheet to the workbook
        XLSX.utils.book_append_sheet(wb, ws, "Sheet1");

        // Generate Excel file and trigger download
        XLSX.writeFile(wb, "table_data.xlsx");
```

```
}
</script>
```

Outcome:



S.No	System No	Departme	Username	Location	Block	Port Numl	Model	Processor	Motherbo	Storage	Memory	Graphic Card
1	DC061	ESDS	R.ADIRAJU	VSPITP	1B	D62	No details	3.30 gigah	Board: ASI	1000.10 G	16384 Meg	NVIDIA GeForce 210 [Display adapter]
2	pc075	ESDS	S N RAO	VSPITP	1B	D61	Dell Inc. P	3.20 gigah	Board: De	500.00 Gig	8080 Meg	Intel(R) HD Graphics [Display adapter]
3	PC076	ESDS	NOT ASSIC	VSPITP	1B	D95	Hewlett-P	2.67 gigah	Board: He	500.00 Gig	8124 Meg	NVIDIA GeForce 210 [Display adapter]
4	PC077	ESDS	NOT ASSIC	VSPITP	1B	D94	Hewlett-P	2.67 gigah	Board: He	500.00 Gig	8064 Meg	Intel(R) Q45/Q43 Express Chipset (Microsoft Corporation - WDDM 1.1) [Display adapter] (2x)
5	PC078	ESDS	NOT ASSIC	VSPITP	1B	D55	Dell Inc. P	3.20 gigah	Board: De	487.44 Gig	16340 Meg	NVIDIA Quadro 600 [Display adapter]
6	PC079	ESDS	NOT ASSIC	VSPITP	1B	D72	Dell Inc. P	3.20 gigah	Board: De	487.44 Gig	12244 Meg	NVIDIA Quadro 600 [Display adapter]
7	PC080	ESDS	T.SATYAN	VSPITP	1B	D67	No details	3.30 gigah	Board: ASI	500.00 Gig	16384 Meg	AMD Radeon HD 5450 [Display adapter]
8	PC081	ESDS	JHANVI	VSPITP	1B	D58	No details	3.30 gigah	Board: ASI	500.00 Gig	16384 Meg	ASUS EAH5450 Series [Display adapter]
9	PC088	ESDS	PAVAN KU	VSPITP	1B	D57	Hewlett-P	2.67 gigah	Board: He	500.00 Gig	8124 Meg	NVIDIA GeForce 210 [Display adapter]
10	PC089	ESDS	CH SRINIV	VSPITP	1B	D56	Hewlett-P	2.67 gigah	Board: He	500.00 Gig	8124 Meg	NVIDIA GeForce 210 [Display adapter]
11	PC090	ESDS	B.SIREESH	VSPITP	1B	D63	Hewlett-P	2.67 gigah	Board: He	498.94 Gig	7100 Meg	AMD Radeon HD 5450 [Display adapter]
12	PC091	ESDS	JHANRDH	VSPITP	1B	D60	Hewlett-P	2.67 gigah	Board: He	499.99 Gig	8064 Meg	Intel(R) Q45/Q43 Express Chipset (Microsoft Corporation - WDDM 1.1) [Display adapter] (2x)
13	PC092	ESDS	DS REDDY	VSPITP	1B	D66	Hewlett-P	2.67 gigah	Board: He	500.00 Gig	8124 Meg	NVIDIA GeForce 210 [Display adapter]
14	PC095	ESDS	POLIREDD	VSPITP	1B	D70	Hewlett-P	2.67 gigah	Board: He	500.11 Gig	8124 Meg	NVIDIA GeForce 210 [Display adapter]
15	PC096	ESDS	NARASIMH	VSPITP	1B	D71	No details	3.30 gigah	Board: ASI	500.00 Gig	16384 Meg	NVIDIA GeForce 210 [Display adapter]
16	PC098	ESDS	ANILA	VSPITP	1B	D59	Hewlett-P	2.67 gigah	Board: He	1000.10 G	8124 Meg	NVIDIA Quadro FX 1800 [Display adapter]
17	ESDSERVE	ESDS	SERVERDC	VSPITP	1B	SERVERRO	IBM IBM S	2.13 gigah	Board: IBI	1392.94 G	8182 Meg	Standard VGA Graphics Adapter [Display adapter]
18	PC094	ESDS	K.SURESH	VSPITP	1B	D69	Hewlett-P	2.67 gigah	Board: He	500.00 Gig	8124 Meg	NVIDIA GeForce GT 610 [Display adapter]

CONCLUSION

In conclusion, the development of a JavaScript application to parse and display detailed system information from a Belarc Advisor HTML report addresses the critical need for efficient data comprehension and analysis. By navigating the intricate HTML structure, the application extracts essential data points including hardware specifications, software inventory, network details, and security status. The formatting of this information into an HTML table ensures clarity and organization, enabling users to easily interpret and utilize the extracted data. This project underscores the importance of robust HTML parsing techniques to handle complexities inherent in diverse system reports, ensuring accuracy in data extraction and presentation. Ultimately, the application enhances user productivity by providing a streamlined approach to accessing and understanding comprehensive system insights, thereby supporting informed decision-making and effective management of IT resources