index_html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Smart Search</title>
   <link rel="stylesheet" href="styles.css">
</head>
<body>
   <div class="layout">
      <aside class="toc">
         <h2>A-Z Knowledgebase</h2>
         <ul id="tocList"></ul>
      </aside>
      <main class="content">
         <h1>Search The Knowledgebase</h1>
         <input type="text" id="searchBox" placeholder="Search..." oninput="filterResults()">
         <div id="contentDisplay">
            <p>Select a topic from the A-Z list or use the search above.</p>
         </div>

<section id="editorSection" style="margin-top:40px;">
  <h2>Edit Knowledgebase</h2>
  <form id="editorForm">
    <label for="parentSelect">Parent Category:</label>
    <select id="parentSelect"></select><br><br>

    <label for="itemTitle">Title:</label><br>
    <input type="text" id="itemTitle" style="width:100%;" /><br><br>

    <label for="itemContent">Content (HTML allowed):</label><br>
    <textarea id="itemContent" rows="10" style="width:100%;"></textarea><br><br>

    <button type="button" id="saveItem">Save</button>
    <button type="button" id="deleteItem">Delete</button>
<br>
    <button id="exportJsButton" onclick="exportDataToFile()">💾 Save to File</button>
<button id="exportTxtButton" onclick="exportDataAsTxt()">📄 Save as Text</button> <input
type="file" id="fileInput" onchange="importDataFromFile(this)"
accept=".js,.json,.txt"><br><br>
<p id="editorStatus" style="margin-top:10px; color:green;"></p>
  </form>
</section>

      </main>
   </div>
   <script src="data.js"></script>
   <script src="app.js"></script>
   <script src="editor.js"></script>
</body>
</html>
```

———-

app_js

```
document.addEventListener("DOMContentLoaded", function () {
    const tocRoot = document.getElementById('tocList');
    const contentDisplay = document.getElementById('contentDisplay');
    const searchBox = document.getElementById('searchBox');

    const dataMap = new Map();
    const childMap = new Map();

    data.forEach(item => {
        dataMap.set(item.id, item);
        if (item.parent) {
            if (!childMap.has(item.parent)) {
                childMap.set(item.parent, []);
            }
            childMap.get(item.parent).push(item);
        } else {
            if (!childMap.has(null)) childMap.set(null, []);
            childMap.get(null).push(item);
        }
    });

    function buildTOC(parentId, container) {
        const children = childMap.get(parentId) || [];
        children.forEach(item => {
            const li = document.createElement('li');
            const titleSpan = document.createElement('span');
            titleSpan.textContent = item.title;
            titleSpan.classList.add('toc-item');
            li.appendChild(titleSpan);
            container.appendChild(li);

            if (item.content && !item.parent) {
                titleSpan.onclick = () => {
                    document.querySelectorAll('.collapsible').forEach(el => {
                        if (el !== li.querySelector('ul')) {
                            el.classList.remove('visible');
                        }
                    });
                    const subList = li.querySelector('ul');
                    if (subList) subList.classList.toggle('visible');
                };
            } else if (item.content) {
                titleSpan.onclick = () => displayContent(item);
            }

            if (childMap.has(item.id)) {
                const ul = document.createElement('ul');
                ul.classList.add('sub-toc');
```

```
            ul.classList.add('collapsible');
            buildTOC(item.id, ul);
            li.appendChild(ul);
        }
    });
}

function resetTOC() {
    tocRoot.innerHTML = '';
    buildTOC(null, tocRoot);
    contentDisplay.innerHTML = "<p>Select a topic from the TOC or use the search above.</
p>";
}

resetTOC();

window.filterResults = function () {
    const query = searchBox.value.toLowerCase().trim();

    if (query === "") {
        resetTOC();
        return;
    }

    const matches = [];

    data.forEach(item => {
        if (
            item.title.toLowerCase().includes(query) ||
            (item.content && item.content.toLowerCase().includes(query))
        ) {
            matches.push(item);
        }
    });

    tocRoot.innerHTML = '';

    matches.forEach(match => {
        const li = document.createElement('li');
        li.classList.add('toc-match');
        li.textContent = `${match.title}`;
        li.onclick = () => displayContent(match);
        tocRoot.appendChild(li);
    });

    if (matches.length > 0) {
        renderSummary(matches);
    } else {
        contentDisplay.innerHTML = "<p>No results found.</p>";
    }
};

function renderSummary(items) {
    contentDisplay.innerHTML = "<h2>Search Results</h2>";
```

```
        const ul = document.createElement('ul');
        items.forEach(item => {
            const li = document.createElement('li');
            const a = document.createElement('a');
            a.href = "#";
            a.textContent = item.title;
            a.onclick = (e) => {
                e.preventDefault();
                displayContent(item);
            };
            const snippet = document.createElement('p');
            snippet.textContent = item.content.replace(/<[^>]+>/g, '').slice(0, 100) + "...";
            li.appendChild(a);
            li.appendChild(snippet);
            ul.appendChild(li);
        });
        contentDisplay.appendChild(ul);
    }


    function displayContent(item) {
        contentDisplay.innerHTML = `<h2>${item.title}</h2>${item.content}
        <br><button onclick='editItem(${item.id})'>✏️ Edit This</button>`;
    }

});
```

———

editor_js

```
document.addEventListener("DOMContentLoaded", function () {
    const editorForm = document.getElementById("editorForm");
    const parentSelect = document.getElementById("parentSelect");
    const titleInput = document.getElementById("itemTitle");
    const contentInput = document.getElementById("itemContent");
    const saveBtn = document.getElementById("saveItem");
    const deleteBtn = document.getElementById("deleteItem");
    const statusMsg = document.getElementById("editorStatus");

    let editingId = null;

    window.editItem = function(itemId) {
        const item = data.find(i => i.id === itemId);
        if (item) {
            editingId = itemId;
            titleInput.value = item.title;
            contentInput.value = item.content;
            if (item.parent) parentSelect.value = item.parent;
            statusMsg.textContent = `Editing item ${editingId}`;
        }
```

```
};

function populateParentOptions() {
    parentSelect.innerHTML = '';
    data.forEach(item => {
        if (!item.parent) {
            const option = document.createElement("option");
            option.value = item.id;
            option.textContent = item.title;
            parentSelect.appendChild(option);
        }
    });
}

function resetEditor() {
    editingId = null;
    titleInput.value = '';
    contentInput.value = '';
    statusMsg.textContent = 'Ready to add new item.';
}

function saveToLocal() {
    localStorage.setItem("knowledgebaseData", JSON.stringify(data));
}

function loadFromLocal() {
    const stored = localStorage.getItem("knowledgebaseData");
    if (stored) {
        try {
            const parsed = JSON.parse(stored);
            if (Array.isArray(parsed)) {
                data.length = 0;
                data.push(...parsed);
            }
        } catch (e) {
            console.error("Failed to parse stored data");
        }
    }
}

saveBtn.onclick = () => {
    const title = titleInput.value.trim();
    const content = contentInput.value.trim();
    const parent = parseInt(parentSelect.value);

    if (!title || !content) {
        alert("Title and content are required.");
        return;
    }

    if (editingId !== null) {
        const item = data.find(i => i.id === editingId);
        if (item) {
            item.title = title;
```

```
            item.content = content;
            item.parent = parent;
            statusMsg.textContent = `Updated item ${editingId}`;
        saveToLocal();
        populateParentOptions();
        if (typeof filterResults === 'function') filterResults();
            }
        } else {
            const newId = data.reduce((max, i) => Math.max(max, i.id), 0) + 1;
            data.push({
                id: newId,
                parent: parent,
                title: title,
                content: content
            });
            statusMsg.textContent = `Added item ${newId}`;
        saveToLocal();
        populateParentOptions();
        if (typeof filterResults === 'function') filterResults();
        }

        saveToLocal();
        resetEditor();
        if (typeof filterResults === 'function') filterResults();
};

deleteBtn.onclick = () => {
    if (editingId !== null) {
        const index = data.findIndex(i => i.id === editingId);
        if (index > -1) {
            data.splice(index, 1);
            statusMsg.textContent = `Deleted item ${editingId}`;
            resetEditor();
            saveToLocal();
            if (typeof filterResults === 'function') filterResults();
        }
    }
};

window.exportDataToFile = function () {
    const blobContent = `const data = ${JSON.stringify(data, null, 2)};`;
    const blob = new Blob([blobContent], { type: "application/javascript" });
    const url = URL.createObjectURL(blob);

    const a = document.createElement("a");
    a.href = url;
    a.download = "data.js";
    a.click();
    URL.revokeObjectURL(url);
};

loadFromLocal();
populateParentOptions();
if (typeof filterResults === 'function') filterResults();
```

```
    resetEditor();
    if (typeof filterResults === 'function') filterResults();
});

// Save data.js formatted
window.exportDataToFile = function () {
    const blobContent = `const data = ${JSON.stringify(data, null, 2)};`;
    const blob = new Blob([blobContent], { type: "application/javascript" });
    const url = URL.createObjectURL(blob);
    const a = document.createElement("a");
    a.href = url;
    a.download = "data.js";
    a.click();
    URL.revokeObjectURL(url);
};

// Save raw JSON as .txt
window.exportDataAsTxt = function () {
    const blob = new Blob([JSON.stringify(data, null, 2)], { type: "text/plain" });
    const url = URL.createObjectURL(blob);
    const a = document.createElement("a");
    a.href = url;
    a.download = "knowledgebase_data.txt";
    a.click();
    URL.revokeObjectURL(url);
};

// Load imported file (json or js format with array or assignment)
window.importDataFromFile = function (input) {
    const file = input.files[0];
    if (!file) return;

    const reader = new FileReader();
    reader.onload = function (e) {
        const text = e.target.result;
        try {
            let parsed;
            if (text.trim().startsWith("const data")) {
                parsed = eval(text.replace("const data", "parsed ="));
            } else {
                parsed = JSON.parse(text);
            }
            if (Array.isArray(parsed)) {
                data.length = 0;
                data.push(...parsed);
                localStorage.setItem("knowledgebaseData", JSON.stringify(data));
                if (typeof filterResults === 'function') filterResults();
                populateParentOptions();
    if (typeof filterResults === 'function') filterResults();
                alert("Data imported successfully.");
            } else {
                alert("Invalid data format.");
            }
        } catch (err) {
```

```
            console.error("Import failed:", err);
            alert("Failed to import data.");
        }
    };
    reader.readAsText(file);
};


// Save as JS
window.exportDataToFile = function () {
    const blobContent = `const data = ${JSON.stringify(data, null, 2)};`;
    const blob = new Blob([blobContent], { type: "application/javascript" });
    const url = URL.createObjectURL(blob);
    const a = document.createElement("a");
    a.href = url;
    a.download = "data.js";
    a.click();
    URL.revokeObjectURL(url);
};

// Save as text
window.exportDataAsTxt = function () {
    const blob = new Blob([JSON.stringify(data, null, 2)], { type: "text/plain" });
    const url = URL.createObjectURL(blob);
    const a = document.createElement("a");
    a.href = url;
    a.download = "knowledgebase_data.txt";
    a.click();
    URL.revokeObjectURL(url);
};

// Import from file
window.importDataFromFile = function (input) {
    const file = input.files[0];
    if (!file) return;

    const reader = new FileReader();
    reader.onload = function (e) {
        let text = e.target.result.trim();
        try {
            if (text.startsWith("const data")) {
                text = text.replace(/const data\s*=\s*/, "");
                text = text.replace(/;\s*$/, "");
            }
            const parsed = JSON.parse(text);
            if (Array.isArray(parsed)) {
                data.length = 0;
                data.push(...parsed);
                localStorage.setItem("knowledgebaseData", JSON.stringify(data));
                if (typeof populateParentOptions === 'function') populateParentOptions();
                if (typeof filterResults === 'function') filterResults();
                alert("Imported data successfully.");
            } else {
                alert("Invalid format.");
```

```
            }
        } catch (err) {
            console.error("Import error:", err);
            alert("Could not import data.");
        }
    };
    reader.readAsText(file);
};
```

———————-

style_css

```
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    margin: 0;
    background: #f4f6f8;
    color: #333;
}

.layout {
    display: flex;
    height: 100vh;
    overflow: hidden;
}

aside.toc {
    width: 280px;
    background: #1e1e2f;
    color: white;
    padding: 20px;
    overflow-y: auto;
    box-shadow: 2px 0 5px rgba(0,0,0,0.1);
}

aside.toc h2 {
    font-size: 20px;
    margin-bottom: 15px;
    color: #FCCC44;
    border-bottom: 1px solid #FCCC44;
    padding-bottom: 5px;
}

#tocList {
    list-style: none;
    padding: 0;
}

#tocList li {
    padding: 10px;
    cursor: pointer;
    color: #333;
```

```
    background: #fff;
    border-radius: 4px;
    border: 1px solid #ccc; /* added border */
    margin-bottom: 5px;
    transition: background 0.3s ease, color 0.3s ease;
}

#tocList li:hover {
    background-color: #D41C2C;
    color: #FCCC44;
}

.toc-match {
    background: #fff;
    color: #D41C2C;
    border-left: 4px solid #FCCC44;
    margin-bottom: 10px;
    padding: 10px;
    font-weight: bold;
}

.sub-toc {
    list-style-type: circle;
    padding-left: 20px;
}

.sub-toc li {
    font-size: 90%;
    margin: 5px 0;
}

main.content {
    flex: 1;
    padding: 40px;
    overflow-y: auto;
    background: #ffffff;
    box-shadow: inset 1px 0 5px rgba(0,0,0,0.05);
}

h1 {
    font-size: 24px;
    color: #D41C2C;
}

input#searchBox {
    width: 100%;
    padding: 12px;
    font-size: 16px;
    margin-bottom: 30px;
    border: 1px solid #ccc;
    border-radius: 6px;
}

#contentDisplay {
```

```
    background: #fafafa;
    padding: 20px;
    border: 1px solid #ddd;
    border-radius: 8px;
    box-shadow: 0 1px 3px rgba(0,0,0,0.1);
}

#contentDisplay ul {
    list-style: none;
    padding: 0;
}

#contentDisplay li {
    margin-bottom: 15px;
    padding-bottom: 10px;
    border-bottom: 1px solid #eee;
}

#contentDisplay a {
    font-weight: bold;
    color: #D41C2C;
    text-decoration: none;
}

#contentDisplay a:hover {
    text-decoration: underline;
}

.toc-item {
    display: block;
    font-weight: bold;
    cursor: pointer;
}

.collapsible {
    display: none;
}

.collapsible.visible {
    display: block;
}


/* Enhance the appearance of expanded sub-toc lists */
.sub-toc.collapsible.visible {
    background: #f0f2f5;
    margin-top: 6px;
    padding: 10px 15px;
    border-left: 3px solid #D41C2C;
    border-radius: 6px;
}

.sub-toc li {
    color: #333;
```

```
    font-weight: normal;
    padding: 5px 0;
}


/* Unified width for search bar and content area */
main.content {
    flex: 1;
    padding: 40px;
    overflow-y: auto;
    background: #ffffff;
    box-shadow: inset 1px 0 5px rgba(0,0,0,0.05);
    display: flex;
    flex-direction: column;
}

input#searchBox {
    width: 100%;
    padding: 12px 15px;
    font-size: 16px;
    margin-bottom: 20px;
    border: 1px solid #ccc;
    border-radius: 6px;
    box-sizing: border-box;
}

/* Ensure content display aligns with search box width */
#contentDisplay {
    width: 100%;
    background: #fafafa;
    padding: 20px 25px;
    border: 1px solid #ddd;
    border-radius: 8px;
    box-shadow: 0 1px 3px rgba(0,0,0,0.1);
    box-sizing: border-box;
}

/* Modern card-style list */
#contentDisplay ul {
    list-style: none;
    padding: 0;
    margin: 0;
}

#contentDisplay li {
    background: #fff;
    border: 1px solid #eee;
    border-radius: 6px;
    margin-bottom: 15px;
    padding: 15px 20px;
    transition: box-shadow 0.2s ease;
}

#contentDisplay li:hover {
```

```
    box-shadow: 0 2px 8px rgba(0,0,0,0.08);
}

#contentDisplay a {
    font-weight: bold;
    font-size: 18px;
    color: #D41C2C;
    text-decoration: none;
}

#contentDisplay a:hover {
    text-decoration: underline;
}

#contentDisplay p {
    margin-top: 8px;
    font-size: 14px;
    color: #555;
}

/* Sub-TOC polished */
.sub-toc.collapsible.visible {
    background: #f7f8fa;
    margin-top: 6px;
    padding: 12px 18px;
    border-left: 4px solid #D41C2C;
    border-radius: 6px;
}

.sub-toc li {
    color: #333;
    font-weight: normal;
    padding: 6px 0;
}

/* Modern styled table */
.styled-table {
    width: 100%;
    border-collapse: collapse;
    margin: 20px 0;
    font-size: 14px;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.05);
    background-color: #fff;
}

.styled-table thead tr {
    background-color: #D41C2C;
    color: #ffffff;
    text-align: left;
}

.styled-table th,
.styled-table td {
    padding: 12px 15px;
```

```
    border: 1px solid #ddd;
}

.styled-table tbody tr:nth-child(even) {
    background-color: #f9f9f9;
}


/* Modern button styling */
#editorForm button,
#editorForm input[type="file"] {
    background-color: #D41C2C;
    color: white;
    border: none;
    border-radius: 6px;
    padding: 10px 18px;
    margin: 6px 6px 6px 0;
    font-size: 14px;
    cursor: pointer;
    transition: background-color 0.2s ease-in-out;
}

#editorForm button:hover,
#editorForm input[type="file"]:hover {
    background-color: #a91421;
}

/* File input visual enhancements */
#editorForm input[type="file"] {
    background-color: #FCCC44;
    color: #333;
    font-weight: bold;
}


/* Exact button color rules */
#saveItem {
    background-color: #28a745;
}
#saveItem:hover {
    background-color: #218838;
}

#deleteItem {
    background-color: #dc3545;
}
#deleteItem:hover {
    background-color: #c82333;
}

#exportJsButton {
    background-color: #007bff;
}
#exportJsButton:hover {
```

```
    background-color: #0069d9;
}

#exportTxtButton {
    background-color: #6f42c1;
}
#exportTxtButton:hover {
    background-color: #5936a2;
}

#fileInput {
    background-color: #FCCC44;
    color: #333;
    font-weight: bold;
}
#fileInput:hover {
    background-color: #e0b800;
}


/* Forced button color rules with !important */
#saveItem {
    background-color: #28a745 !important;
    color: white !important;
}
#saveItem:hover {
    background-color: #218838 !important;
}

#deleteItem {
    background-color: #dc3545 !important;
    color: white !important;
}
#deleteItem:hover {
    background-color: #c82333 !important;
}

#exportJsButton {
    background-color: #007bff !important;
    color: white !important;
}
#exportJsButton:hover {
    background-color: #0069d9 !important;
}

#exportTxtButton {
    background-color: #6f42c1 !important;
    color: white !important;
}
#exportTxtButton:hover {
    background-color: #5936a2 !important;
}

#fileInput {
```

```
    background-color: #FCCC44 !important;
    color: #333 !important;
    font-weight: bold !important;
}
#fileInput:hover {
    background-color: #e0b800 !important;
}


/* Modern editor section styling */
#editorSection {
    background-color: #fff;
    padding: 30px;
    margin-top: 40px;
    border: 1px solid #ddd;
    border-radius: 8px;
    box-shadow: 0 2px 6px rgba(0,0,0,0.05);
    max-width: 800px;
    margin-left: auto;
    margin-right: auto;
}

#editorSection h2 {
    margin-bottom: 20px;
    color: #0c2340;
}

#editorForm label {
    font-weight: bold;
    display: block;
    margin-top: 16px;
    margin-bottom: 6px;
}

#editorForm input[type="text"],
#editorForm select,
#editorForm textarea {
    width: 100%;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 6px;
    font-size: 14px;
    box-sizing: border-box;
    margin-bottom: 10px;
}

#editorForm button {
    margin-right: 10px;
    margin-top: 10px;
}


/* Match editor width to content area */
#editorSection {
```

```
    width: 100%;
    max-width: 1000px;
}
```

data_js

```
const data = [
  {
    "id": 1,
    "title": "A",
    "content": "<p>Topics starting with A.</p>"
  },
  {
    "id": 2,
    "title": "B",
    "content": "<p>Topics starting with B.</p>"
  },
  {
    "id": 3,
    "title": "C",
    "content": "<p>Topics starting with C.</p>"
  },
  {
    "id": 4,
    "title": "D",
    "content": "<p>Topics starting with D.</p>"
  },
  {
    "id": 5,
    "title": "E",
    "content": "<p>Topics starting with E.</p>"
  },
  {
    "id": 6,
    "title": "F",
    "content": "<p>Topics starting with F.</p>"
  },
  {
    "id": 7,
    "title": "G",
    "content": "<p>Topics starting with G.</p>"
  },
  {
    "id": 8,
    "title": "H",
    "content": "<p>Topics starting with H.</p>"
  },
  {
    "id": 9,
    "title": "I",
    "content": "<p>Topics starting with I.</p>"
  },
```

```
{
  "id": 10,
  "title": "J",
  "content": "<p>Topics starting with J.</p>"
},
{
  "id": 11,
  "title": "K",
  "content": "<p>Topics starting with K.</p>"
},
{
  "id": 12,
  "title": "L",
  "content": "<p>Topics starting with L.</p>"
},
{
  "id": 13,
  "title": "M",
  "content": "<p>Topics starting with M.</p>"
},
{
  "id": 14,
  "title": "N",
  "content": "<p>Topics starting with N.</p>"
},
{
  "id": 15,
  "title": "O",
  "content": "<p>Topics starting with O.</p>"
},
{
  "id": 16,
  "title": "P",
  "content": "<p>Topics starting with P.</p>"
},
{
  "id": 17,
  "title": "Q",
  "content": "<p>Topics starting with Q.</p>"
},
{
  "id": 18,
  "title": "R",
  "content": "<p>Topics starting with R.</p>"
},
{
  "id": 19,
  "title": "S",
  "content": "<p>Topics starting with S.</p>"
},
{
  "id": 20,
  "title": "T",
  "content": "<p>Topics starting with T.</p>"
```

```
  },
  {
    "id": 21,
    "title": "U",
    "content": "<p>Topics starting with U.</p>"
  },
  {
    "id": 22,
    "title": "V",
    "content": "<p>Topics starting with V.</p>"
  },
  {
    "id": 23,
    "title": "W",
    "content": "<p>Topics starting with W.</p>"
  },
  {
    "id": 24,
    "title": "X",
    "content": "<p>Topics starting with X.</p>"
  },
  {
    "id": 25,
    "title": "Y",
    "content": "<p>Topics starting with Y.</p>"
  },
  {
    "id": 26,
    "title": "Z",
    "content": "<p>Topics starting with Z.</p>"
  }
];
```