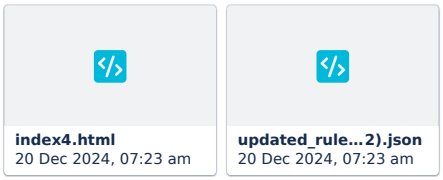
3.HTLM Json Editor



Here is the complete code with all external resources replaced with local files. This includes the required JavaScript and CSS files from the CodeMirror and D3.js libraries, stored locally. You'll need to download these files and place them in the appropriate directories.

Project Directory Structure:

```
/project-directory
  /css
    codemirror.css
    dracula.css
  /js
    codemirror.js
    javascript.js
    matchbrackets.js
    d3.js
  /index.html

```

Full Code:

1. `index.html`

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>JSON Rule Editor with Enhanced Tree Visualization</title>
  <!-- Local Stylesheets -->
  <link rel="stylesheet" href="css/codemirror.css">
  <link rel="stylesheet" href="css/dracula.css"> <!-- Optional theme -->
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
    }
    #jsonEditor, #treeContainer {
      border: 1px solid #ccc;
      padding: 10px;
      border-radius: 5px;
      margin-top: 20px;
    }
    #treeContainer {
      margin-top: 30px;
    }
```

```
    button {
      margin: 10px 0;
      padding: 10px 20px;
      background-color: #007bff;
      color: white;
      border: none;
      cursor: pointer;
    }
    button:hover {
      background-color: #0056b3;
    }
    svg {
      width: 100%;
      height: 600px;
      border: 1px solid #ccc;
    }
    .error {
      color: red;
      font-size: 14px;
      margin-top: 10px;
    }
    #jsonEditorWrapper {
      display: flex;
      margin-top: 10px;
    }
    .CodeMirror {
      border: 1px solid #ccc;
      border-radius: 5px;
      font-family: monospace;
      height: 400px;
      width: 100%;
    }
    .error-line {
      background-color: #ffcccb;
    }
  </style>
</head>
<body>
  <h1>JSON Rule Editor with Enhanced Tree Visualization</h1>
  <input type="file" id="fileInput" />
  <div id="jsonEditor">
    <h2>JSON Editor</h2>
    <div id="jsonEditorWrapper">
      <textarea id="jsonTextArea" rows="15" cols="80"></textarea>
    </div>
    <br />
    <button id="downloadBtn">Download JSON</button>
    <button id="refreshBtn">Refresh Tree</button>
    <button id="formatBtn">Format JSON</button>
    <p id="errorMessage" class="error" style="display: none;"></p>
  </div>
  <div id="treeContainer">
    <h2>Decision Tree Visualization</h2>
    <div id="treeView"></div>
  </div>

  <!-- Local Scripts -->
  <script src="js/codemirror.js"></script>
```

```
  <script src="js/javascript.js"></script>
  <script src="js/matchbrackets.js"></script>
  <script src="js/d3.js"></script>

  <script>
    let jsonData = null;
    let jsonEditor = null;

    // Initialize CodeMirror editor
    function initializeCodeMirror() {
      jsonEditor = CodeMirror.fromTextArea(document.getElementById("jsonTextArea"), {
        mode: "application/json",
        theme: "dracula", // Optional theme
        lineNumbers: true,  // Enable built-in line numbers
        matchBrackets: true,
        lineWrapping: true
      });

      // When the content changes, update the tree and validate JSON
      jsonEditor.on('change', function() {
        const updatedJson = jsonEditor.getValue();
        const result = validateJSONWithLineNumbers(updatedJson);
        if (result.isValid) {
          document.getElementById("errorMessage").style.display = "none"; // Hide error message
        } else {
          document.getElementById("errorMessage").textContent = `Invalid JSON format. Error at line
${result.errorLine}`;
          document.getElementById("errorMessage").style.display = "block";
          highlightErrorLine(updatedJson, result.errorLine); // Highlight error line
        }
      });
    }

    // Handle file upload
    document.getElementById("fileInput").addEventListener("change", function (event) {
      const file = event.target.files[0];
      const reader = new FileReader();
      reader.onload = (e) => {
        try {
          jsonData = JSON.parse(e.target.result);
          jsonEditor.setValue(JSON.stringify(jsonData, null, 2)); // Set the JSON content in CodeMirror
editor
          renderTree(jsonData); // Render initial tree
          document.getElementById("errorMessage").style.display = "none"; // Hide error message
        } catch (error) {
          document.getElementById("errorMessage").textContent = "Invalid JSON file.";
          document.getElementById("errorMessage").style.display = "block";
        }
      };
      reader.readAsText(file);
    });

    // Handle JSON download
    document.getElementById("downloadBtn").addEventListener("click", function () {
      const updatedJson = jsonEditor.getValue();
      if (isValidJSON(updatedJson)) {
        const blob = new Blob([updatedJson], { type: "application/json" });
        const link = document.createElement("a");
```

```
        link.href = URL.createObjectURL(blob);
        link.download = "updated_rules.json";
        link.click();
      }
    });

    // Handle refresh button click to re-render the tree
    document.getElementById("refreshBtn").addEventListener("click", function () {
      const updatedJson = jsonEditor.getValue();
      const result = validateJSONWithLineNumbers(updatedJson);
      if (result.isValid) {
        jsonData = JSON.parse(updatedJson); // Update jsonData with edited JSON
        renderTree(jsonData); // Re-render tree with updated JSON
        document.getElementById("errorMessage").style.display = "none"; // Hide error message
      } else {
        document.getElementById("errorMessage").textContent = `Invalid JSON format. Error at line
${result.errorLine}`;
        document.getElementById("errorMessage").style.display = "block";
        highlightErrorLine(updatedJson, result.errorLine); // Highlight error line
      }
    });

    // Handle format button click
    document.getElementById("formatBtn").addEventListener("click", function () {
      const updatedJson = jsonEditor.getValue();
      const result = validateJSONWithLineNumbers(updatedJson);
      if (result.isValid) {
        const formattedJson = JSON.stringify(JSON.parse(updatedJson), null, 2);
        jsonEditor.setValue(formattedJson); // Set formatted JSON in editor
        document.getElementById("errorMessage").style.display = "none"; // Hide error message
      } else {
        // If invalid JSON, show the error message and highlight the error
        document.getElementById("errorMessage").textContent = `Invalid JSON format. Error at line
${result.errorLine}`;
        document.getElementById("errorMessage").style.display = "block";
        highlightErrorLine(updatedJson, result.errorLine); // Highlight error line
      }
    });

    // JSON Validation Function with Line Number
    function validateJSONWithLineNumbers(jsonString) {
      try {
        JSON.parse(jsonString);
        return { isValid: true };
      } catch (error) {
        const lineNumber = extractLineNumber(error.message);
        return { isValid: false, errorLine: lineNumber };
      }
    }

    // Extract line number from the error message
    function extractLineNumber(errorMessage) {
      const match = errorMessage.match(/in JSON at position (\d+)/);
      if (match) {
        const position = parseInt(match[1]);
        const jsonText = jsonEditor.getValue();
        const lines = jsonText.split('\n');
        let charCount = 0;
```

```
        for (let i = 0; i < lines.length; i++) {
          charCount += lines[i].length + 1; // Account for newline
          if (charCount > position) {
            return i + 1; // Return line number (1-based index)
          }
        }
      }
      return null;
    }

    // Highlight the line with JSON error
    function highlightErrorLine(jsonString, errorLine) {
      const lines = jsonString.split('\n');
      const start = lines.slice(0, errorLine - 1).join('\n').length;
      const end = start + lines[errorLine - 1].length;
      jsonEditor.setSelection({ line: errorLine - 1, ch: start }, { line: errorLine - 1, ch: end });

      // Highlight the corresponding line in CodeMirror
      const lineElems = jsonEditor.getDoc().lineInfo(errorLine - 1).text;
      jsonEditor.addLineClass(errorLine - 1, 'background', 'error-line');
    }

    // Render collapsible tree
    function renderTree(data) {
      const treeData = data.features[0].rule.conditions.map((cond) => generateTree(cond));

      const root = d3.hierarchy({ name: "Root", children: treeData });

      const svg = d3.select("#treeView").html("").append("svg").attr("width", 1000).attr("height", 600);

      const g = svg.append("g").attr("transform", "translate(50,50)");

      const treeLayout = d3.tree().size([900, 500]);
      const treeRoot = treeLayout(root);

      // Links
      g.selectAll(".link")
        .data(treeRoot.links())
        .enter()
        .append("line")
        .attr("class", "link")
        .attr("x1", (d) => d.source.x)
        .attr("y1", (d) => d.source.y)
        .attr("x2", (d) => d.target.x)
        .attr("y2", (d) => d.target.y)
        .attr("stroke", "#aaa")
        .attr("stroke-width", 2);

      // Nodes
      const nodes = g.selectAll(".node")
        .data(treeRoot.descendants())
        .enter()
        .append("g")
        .attr("class", "node")
        .attr("transform", (d) => `translate(${d.x},${d.y})`)
        .on("click", (event, d) => {
          if (d.children) {
            d._children = d.children;
```

```
            d.children = null;
          } else {
            d.children = d._children;
            d._children = null;
          }
          renderTree(data); // Re-render
        });

      nodes.append("circle")
        .attr("r", 10)
        .attr("fill", (d) => (d.children || d._children ? "#007bff" : "#ffa500"))
        .attr("stroke", "#333")
        .attr("stroke-width", 2);

      nodes.append("text")
        .attr("dy", -15)
        .attr("text-anchor", "middle")
        .text((d) => d.data.name)
        .style("font-size", "12px");
    }

    function generateTree(condition) {
      return {
        name: condition.condition || condition.value,
        children: condition.sub_conditions ? condition.sub_conditions.map((c) => generateTree(c)) : [],
      };
    }

    // Initialize CodeMirror editor after the DOM is loaded
    window.onload = function() {
      initializeCodeMirror();
    };
  </script>
</body>
</html>

```

2. Local Files to Download

- `codemirror.css`: You can get this from CodeMirror's GitHub repository.
- `dracula.css`: You can find this theme at CodeMirror's themes.
- `codemirror.js`: Download the codemirror.js file.
- `javascript.js`: Download the JavaScript mode.
- `matchbrackets.js`: Download matchbrackets addon.
- `d3.js`: Download the D3.js library and use the `d3.min.js` file for production.

This structure will ensure that your application works completely offline with local resources.