05.05.2025

Senior Frontend Developer Coding Challenge 1

Hierarchical Asset Management in a Unified Namespace

Context

Cybus Connectware enables powerful industrial data integration using the **Unified Namespace (UNS)** pattern. A UNS allows the modeling of complex asset hierarchies and semantic context in a flexible, scalable way.

In this challenge, you'll build a simplified front end to manage an industrial asset hierarchy with labels and contextual data, inspired by Connectware's principles.

Goals

Build a React-based frontend that allows users to:

- Define a hierarchical structure for a multi-site industrial organization.

  Example of a *minimal tree*:

Unset
```
Cybus
├── Germany
│   ├── Hamburg
│   │   └── Hall A
│   │         └── Line 1
│   │         ├── DEHHWelding_03 [Asset]
│   │                │      ├── current [Datapoint]
│   │                │      ├── pressure [Datapoint]
│   │                │      └── state [Datapoint]
│   │         ├── DEHHWelding_04 [Asset]
│   │         └── AOI
│   │                       └── DEHHCam_1 [Asset]
│   │
│   └── Berlin
│      ├── Line 1
│      │   └── DEBWelding_03 [Asset]
│      └── CMS [Asset]
└── Monitoring [Asset]
```

- **Modify the hierarchy dynamically**.
- **Assign assets** to any node in the hierarchy.

- **Assign labeled metadata** to:
  - **Nodes** (e.g. `Project: CybusPowerCell`)
  - **Assets** (e.g. `Asset: DEHHWelding_03`, `Type: IS-Q6000A`)
  - **Datapoints**, which are part of an asset (e.g. `current`, `pressure`, `state`)
- Retrieve and display:
  - **Full hierarchy path** of any asset or datapoint
    Example: `Cybus > Germany > Hamburg > Hall A > Line 1 > DEHHWelding_03 > pressure`
  - **All labels** of any asset or datapoint
  - **All labels** of parent nodes

Requirements

Functional

Users can:

- Add/edit/delete nodes in the hierarchy
- Assign assets to hierarchy nodes
- Add/edit/delete labels for nodes, assets and datapoints
- Retrieve the full path of a selected asset or datapoint
- View all labels for an asset or datapoint including the labels of their parent nodes

Technical

- Use **React** with **TypeScript**
- Use `localStorage` with asynchronous access actions
- No UI framework required, but you may use Material UI as we do

- Provide **clear documentation** on how to run and use the application
- Structure code for **scalability and testability**

Bonus

Extra Credit

- **Visualization** of the hierarchy structure
- **Drag-and-drop** support to reorder or move nodes in the hierarchy

Nice to Have

- **Search/filtering** by node, asset name, or label

Submission

Please provide a GitHub repo or archive that includes:

- Source code
- README with:
    - Setup instructions
    - A screenshot or short Loom-style video walkthrough
- A short file describing:
    - Your approach
    - Data model
    - Key decisions and trade-offs

Use Your Judgment

This challenge intentionally leaves room for interpretation. Feel free to be creative with the UI/UX, component structure, and features. We'd love to see how you think and what you prioritize when given the freedom to design your own solution.