With the database layer and Sparkplug engine done, now it’s time to build a slick web UI to allow users to browse the Devices, their Metrics and the metric value history for a given metric.

At a high level, the web app should:

* Provide a universal search bar, allowing them to type a partial name of a device or device metric with dynamic autosuggest
* Offer a three-pane explorer interface where the first pane shows all devices, the send pane shows metrics for the device selected in the first pane and a third pane that shows the current value for the metric along with a table of value history and a graph allowing quick visualization of the value over the past week.

# Detailed Requirements

## Authentication

No auth for now

## Search

The search box should be prominent at the top of the UI. As the user types, a list of possible matches should hover below the search box. Each search suggestion should include context about the type of search result (device or metric). In the case of metrics, a breadcrumb-like interface should show the parent device as well. An icon next to each result should also indicate the type of result: device or metric. SVG would be preferred for fast loading and caching.

## Devices Pane

Should list all devices sorted by device name along with other relevant details including the device’s birth date. The UI should be extensible to accommodate additional metadata in the future. The UI should scroll vertically and be very responsive and efficient. On initial load, the first item in the list should be automatically selected. The user should be able to use arrow keys to navigate the list. The user’s selection should survive a browser refresh.

## Metrics Pane

The metrics pane, appearing as the second column, should dynamically update to display all available metrics associated with the selected device. Each metric entry should include the metric’s name, most recent value and timestamp.

## Metric Detail Pane

The third pane should serve as a focused detail view. Here, users can see the real-time value of the selected metric, a compact data table displaying recent values over time, and an embedded graph visualizing the metric’s trend for the past seven days. Users should be able to toggle between different time ranges (such as last 24 hours, last week, or custom range) for both the table and graph.

Both the metrics and value-history panes must be optimized for performance, even when handling a large number of devices or high-frequency data streams. Keyboard navigation should allow smooth transition between panes, supporting power users.

# Technical Details

## RESOURCE APIS

We will need an API exposes the required information to the UI. A RESTful API is preferred. As for the content type, JSON is fine but if there is a more efficient way to encode lots of data that does not add too much complexity on the UI feel free to do that.

The API should support pagination assuming alphanumeric sorting for now but should be flexible enough to support other sorting strategies in the future.

We also need to support very large responses, so consider having the API stream results to the client.

## SEARCH API

We also need a universal search API that is quick enough to support dynamic autocomplete / suggestions. For now, search strings are only expected to match device names and metric names. This may change in the future.

## UI

Open to any web framework as long as it’s popular, has strong tooling support and can handle a large amount of data. I’m most familiar with React, Vite, MUI DataGrid but open to anything else if it’s a better fit.

The UI should have a toggle for dark / light theme.

We should assume the backend is fast so no need for a data loading indicator / spinner.

## Non-Functional Requirements

* Use local storage for user personalization (remembering their selection)
* No need to cache data locally in the browser. Assume the backend is fast.

If any of these requirements are not clear please stop and ask me questions. I’m not looking for a “one shot” implementation. Would prefer to refine the requirements before wasting tokens generating undesired code.