React.js Storage.Timeline Widgets

Introduction

`Storage.Timeline` is a database for handling time series (timeline) data. More documentation on the database can be found here: https://github.com/vitche/documentation-storage-timeline

Author #1:	Andrew Mikhailov andrew@vitche.com
Author #2:	Herman Stepanov germantv663@gmail.com
Version:	1.0
Date:	2024.03.02

Visualizing Time Series Data with UI Components

In the realm of data analysis and monitoring, the ability to visualize and interact with time series data is paramount. `Storage.Timeline` is a robust database designed specifically for handling time series (timeline) data, offering a solid foundation for storing and retrieving temporal data points efficiently. To leverage the full potential of `Storage.Timeline`, developers and users alike require a comprehensive and convenient User Interface (UI) that allows for intuitive navigation and visualization of time series data.

The Importance of UI in Time Series Analysis

Time series data, by its nature, is vast and complex, often encompassing countless data points across a continuous time spectrum. The sheer volume of data can be overwhelming, making it challenging to discern patterns, trends, and anomalies. A well-designed UI bridges this gap by providing tools and visualizations that transform raw data into understandable and actionable insights.

Features of an Effective Time Series Data UI

An effective UI for navigating `Storage.Timeline` data should include the following features:

- **Easily navigating the storage's schema:** The `Storage.Timeline` database is designed to contain multiple schema, which model the target domain from the time series perspective. So, navigating multiple schema must be done conveniently.
- **Interactive Time Range Selection:** Users should be able to select and modify the time range of the data they wish to view, enabling them to focus on specific intervals of interest.

- **Zoom and Pan Capabilities:** To examine data at various levels of granularity, the UI should offer zoom and pan functionalities, allowing users to drill down into detailed views or out to broader overviews with ease.
- **Real-time Data Streaming:** For applications requiring up-to-the-minute data, the UI should support real-time data streaming, updating visualizations dynamically as new data arrives.
- **Customizable Data Plots:** Different use cases call for different types of visualizations (e.g., line charts, bar graphs, heat maps). A versatile UI should provide a variety of customizable data plots to suit various analytical needs.
- **Annotations and Event Markers:** Users often need to note significant events or anomalies within the data. The UI should allow for the placement of annotations and event markers within the visualizations for easy reference and analysis.
- **Responsive Design:** With the increasing use of mobile devices for data analysis, the UI must be responsive, ensuring a seamless experience across desktops, tablets, and smartphones.

UI use-cases

Integrating the `Storage.Timeline` widget to any Website

The UI widgets need to be designed in a way that it should be possible to:

- Place widgets on any third-party Website, which supports `React.js`.
- Display the `Unconnected Widget`, which connects to the database, specified by the user.
- Display the `Connected Store Widget`, which immediately displays all schema in the specified store.
- Display the `Connected Schema Widget`, which immediately displays all time-lines in the specified store and schema.
- Display the `Connected Timeline Widget`, which immediately displays all time-line values in the specified store, schema and timeline.
- Navigate from the unconnected widget to the store widget and then to the schema and timeline widgets.

Connecting to the database

To display content for a particular `Storage.Timeline` store:

1. The user specifies the URL to the database either for version #1 or for version #2 APIs. E.g. https://europe-west1-hype-dev.cloudfunctions.net/storage-timeline.
2. The user clicks `Connect`.
3. The list of schema in this `Storage.Timeline` store is displayed.

Navigating the `Store`

The `Connected Store Widget` displays the list of all schema in the store. Each schema is clickable.

Navigating the `Schema`

The `Connected Schema Widget` displays the list of all timelines in the schema. Each timeline is clickable.

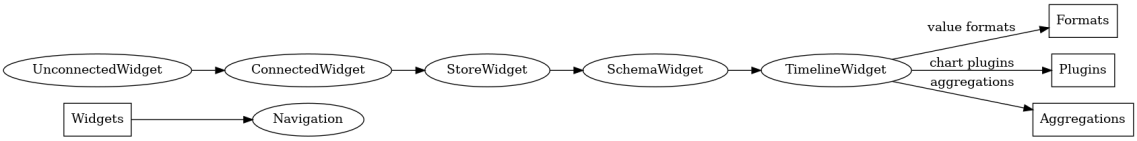
Navigating the `Timeline`

- The `Connected Timeline Widget` displays the list of all timeline data.
- It is possible to choose the timeline value format:
  - The string.

    - The number.
    - The JSON document.
- It is possible to copy the value into clipboard.
- It is possible to download the given value as a file.
- **IN-FUTURE**. It is possible to display additional chart plugins, which display the time series data. One plugin is already available, which displays biological activity data.
- **IN-FUTURE**. It is possible to apply the `In-Browser Aggregate Computation` to the time series data.

The components diagram

Below is the project's components diagram:



Conclusion

The development of a comprehensive and convenient UI for `Storage.Timeline` is not just a luxury; it is a necessity for anyone looking to harness the power of time series data. Such an interface will empower developers, analysts, and business users to make informed decisions based on clear, interactive visualizations of temporal data. As `Storage.Timeline` continues to evolve, so too should the tools we use to interact with it, ensuring that we can continue to unlock the full potential of time series analysis.