Development

Created by Oliver Paszkiewicz

Table of Contents

Typed Setup JSON

The entire setup JSON underwent TypeScript typing to enforce strict typing, enhancing code robustness, and ensuring safe typing for simplified error handling during development. This phase of the development journey commenced with an initial challenge—a limited understanding of TypeScript and its mechanisms. Bridging this knowledge gap required delving into the fundamentals of this typed programming language. As the development progressed, adjustments to the typed settings object became necessary to address emerging errors within the code.

Reusable elements for rendering settings

Dynamic rendering for various settings was achieved through the creation of reusable React components. The implementation involved sophisticated logic capable of rendering all settings options based on the currently selected setting. This dynamic rendering adds flexibility and adaptability to changes in the data structure. Notably, the logic was intricately designed to automatically render any newly added option to the database, eliminating the need for manual adjustments to the codebase. The challenge in creating these reusable components lay in their integration with Material UI, a React component library that implements Google's Material Design. Consequently, adjustments were required in the data used for rendering to align seamlessly with Material UI's specifications.

Redux for global state management

Redux was employed to manage three distinct global states: the Settings Data State, responsible for controlling the data and handling changes; the Dialog Visibility State, which governs the visibility of all modals and dialogs for consistent user interface interactions; and the JSON Fallback State, managing fallback mechanisms in case the user would like to use the previous interface for managing the settings. This three-fold approach enhances scalability and maintainability by isolating concerns, providing a clear structure for handling different aspects of the application's state. The challenging part was during the setup of these redux states, and I needed some help from my company’s supervisor to do it properly.

Implementation of the logic for saving the settings

The logic for saving the entire JSON settings object from the interface to the backend server has been implemented, utilizing the POST method for sending it to the backend.

Conclusion

My journey commenced with the creation of type-safe code and the implementation of TypeScript types for the entire JSON setup. I successfully accomplished dynamic rendering of settings through reusable React components, incorporating intricate logic for the automatic rendering of new settings options. Despite encountering challenges in integrating with Material UI, necessitating adjustments for seamless alignment, I overcame these hurdles. Leveraging Redux, I managed three global states: Settings Data, Dialog Visibility, and JSON Fallback, ensuring scalability and maintainability. The challenges in setting up Redux states were effectively addressed with guidance from the company's supervisor, ensuring the successful implementation and integration with components.