Yamm task

Your task is to create a dashboard that has a navigation sidebar + a table of list of refund orders. Single order record should be consisting of the following:

```
Id => string;
reason: => string;
store_name => string;
store_logo => sting;
store_url => sting;
amount => number;
active:  boolean;
decision: null;
Items => {
            name: string;
            id: string;
            price: number;
            quantity: number;
    }[]
```

Each row should display all the above information except for the items, display item length only along with a decision that should be not yet + actions.
Table row has three actions:

1. Dropdown menu to change order decision [“reject”, “accept”, “escalate”]
2. A switch that toggles record status whether it should be active or inactive.
3. Last one is an IconButton that upon clicking it, I should be navigated to an order page that has the order items in detail.

- Design and implement a reusable table component that can be utilized across different parts of the application.
- The component should be highly configurable to support varying numbers of columns, different data types, and custom actions.
- Ensure that all props and state are typed appropriately for the table component.
- Any action taken should reflect on the table immediately without needing to reload the page.
- A notification toaster should be shown upon any action taken.
  Table data must be paginated max per_page = 15

While you are developing the task consider the following

Data Fetching

The data for the tables should be fetched from a RESTful API. Implement appropriate loading and error handling states while the data is being fetched.
The APIs can be mocked using a tool like JSON Server or a similar solution.

Git Commit History

Your commit history should clearly reflect the development process, including meaningful commit messages that describe the changes made.

Code Quality

Focus on structure, readability, and adherence to best practices. Component Reusability: Ensure your table component can be reused across different parts of the application.

Separation of Concerns

Keep your logic, state management, and UI components well-separated.

User Experience

Follow a design system and consider responsiveness, accessibility, and overall usability in your UI design.

Evaluation Criteria

Component Design:

The ability to create a generic, reusable table component.

State Management:

Effective use of state management tools and techniques.

API Integration:

Proper handling of API calls and state updates.

Code Quality:

Readability, maintainability, and adherence to best practices.

User Experience:

The applications usability, responsiveness, and accessibility.