# Large List with React

full big project for large list from api with react ts vite react window with efficent scallblity and preformence example

2/2

Creating a large-scale React TypeScript project that efficiently handles large lists from an API using Vite and `react-window` can be broken down into several steps. Below is an example structure and implementation guide that demonstrates how to achieve this, focusing on performance and scalability.

## Project Setup

1. **Initialize a Vite Project:**

   Run the following command to create a new Vite project with React and TypeScript:

   ```bash
   npm create vite@latest my-large-list-app --template react-ts
   cd my-large-list-app
   npm install
   ```

2. **Install Required Dependencies:**

   You will need to install `react-window` for rendering large lists efficiently.

   ```bash
   npm install react-window axios
   ```

## Project Structure

Here's a recommended project structure:

```css
my-large-list-app/
│
├── src/
│   ├── components/
│   │   ├── List.tsx
```

```
│   │   ├── ListItem.tsx
│   │
│   ├── hooks/
│   │   ├── useFetchData.ts
│   │
│   ├── models/
│   │   ├── Item.ts
│   │
│   ├── App.tsx
│   ├── main.tsx
│   ├── styles.css
│
└── package.json
```

## Implementing the Components

1. **Define the Data Model:**

   Create a TypeScript model for the items you expect from your API. For example, `src/models/Item.ts`:

   ```typescript
   export interface Item {
       id: number;
       title: string;
       // Add other fields as needed
   }
   ```

2. **Create a Custom Hook to Fetch Data:**

   Create a hook for fetching data from the API, `src/hooks/useFetchData.ts`:

   ```typescript
   import { useEffect, useState } from 'react';
   import axios from 'axios';
   import { Item } from '../models/Item';

   const useFetchData = (url: string) => {
       const [data, setData] = useState<Item[]>([]);
       const [loading, setLoading] = useState<boolean>(true);
       const [error, setError] = useState<string | null>(null);
   ```

```
    useEffect(() => {
        const fetchData = async () => {
            try {
                const response = await axios.get<Item[]>(url);
                setData(response.data);
            } catch (error) {
                setError('Error fetching data');
            } finally {
                setLoading(false);
            }
        };

        fetchData();
    }, [url]);

    return { data, loading, error };
};

export default useFetchData;
```

3. **Create the List Component:**

   This component will render the list using `react-window`, `src/components/List.tsx`:

```typescript
import React from 'react';
import { FixedSizeList as List } from 'react-window';
import ListItem from './ListItem';
import { Item } from '../models/Item';

interface Props {
    items: Item[];
}

const ItemList: React.FC<Props> = ({ items }) => {
    const itemSize = 50; // Height of each item in pixels

    return (
        <List
            height={500} // Height of the list
            itemCount={items.length}
```

```
            itemSize={itemSize}
            width="100%"
        >
            {({ index, style }) => (
                <div style={style}>
                    <ListItem item={items[index]} />
                </div>
            )}
        </List>
    );
};

export default ItemList;
```

4. **Create a List Item Component:**

   This component will render individual items in the list, `src/components/ListItem.tsx`:

```typescript
import React from 'react';
import { Item } from '../models/Item';

interface Props {
    item: Item;
}

const ListItem: React.FC<Props> = ({ item }) => {
    return (
        <div className="list-item">
            <h3>{item.title}</h3>
        </div>
    );
};

export default ListItem;
```

5. **Update the App Component:**

   Use the hook and components in your `App.tsx`:

```typescript
```

```
import React from 'react';
import useFetchData from './hooks/useFetchData';
import ItemList from './components/List';
import './styles.css';

const App: React.FC = () => {
    const { data, loading, error } =
useFetchData('https://api.example.com/items');

    if (loading) return <div>Loading...</div>;
    if (error) return <div>{error}</div>;

    return (
        <div className="App">
            <h1>Large List Example</h1>
            <ItemList items={data} />
        </div>
    );
};

export default App;
```

## Styling

Add some basic styles in `src/styles.css`:

```
.App {
    text-align: center;
    padding: 20px;
}

.list-item {
    padding: 10px;
    border: 1px solid #ccc;
    margin: 5px 0;
    border-radius: 4px;
}
```

## Running the Project

Run the following command to start your development server:

```bash
npm run dev
```

## Conclusion

This setup provides a solid foundation for building a large-scale application that handles large lists efficiently with React, TypeScript, and Vite.

- `react-window` enables efficient rendering of large lists by only rendering items in the viewport, improving performance.
- The use of TypeScript enhances type safety and developer experience.
- The project is structured for scalability, making it easy to add features and components as needed.

You can extend this project by adding features like pagination, sorting, filtering, and more, depending on your application requirements.