Material Table in React

Relevel
by Unacademy

WHAT DOES IT TAKE TO BUILD IT

Material Table – Introduction

Key Features of material table

- Actions
- Component Overriding
- Custom Column Rendering
- Detail Panel
- Editable
- Export
- Filtering
- Grouping
- Localization
- Remote Data
- Search
- Selection
- Sorting
- Styling
- Tree Data

Material Table - Introduction

- Material-table includes an inline filter component that allows users to design custom data filters.

- Details and Case Studies. All of the aforementioned functionalities are supported by the material table, which is probably the only table library that does.

- Material UI is the most popular UI library in the React ecosystem, and the library is developed on top of it. If you're already using Material UI, the material table will fit right in

Installation

Install material-table with npm

- npm install material-table @material-ui/core –save

Add Material Icons

- Importing the material icons font via html OR importing material icons and using the material-table icons prop are the two ways to use icons in material-table.

```
<link
  rel="stylesheet"

href="https://fonts.googleapis.com/icon?famil
y=Material+Icons"
/>
```

Import Material Icons

When compared to using a font library, icons may be imported and utilized with a material table, giving you additional options for customizing the appearance and feel of the table.

```
npm install @material-ui/icons --save
```

Key Features of Material Table

1. Actions

Using the actions prop, you may add buttons to rows or the toolbar. The Actions prop accepts a list of actions.

```
<MaterialTable
  // other props
  actions={[
    {
      icon: 'save',
      tooltip: 'Save User',
      onClick: (event, rowData) => {
        // Do save operation
      }
    }
  ]}
/>
```

2. Component Overriding

The material-table is made up of a lot of different parts. You can alter some of them to suit your requirements.

```
import MaterialTable, { MTableToolbar } from
'material-table';
<MaterialTable
    // other props
    components={{
        Toolbar: props => (
            <div style={{ backgroundColor:
'#e8eaf5' }}>
                <MTableToolbar {...props} />
            </div>
        )
    }}
/>
```

3. Custom Column Rendering

You can use a material table to customize the render of any column. You can, for example, render a picture instead of an image URL.

```
import MaterialTable, { MTableToolbar } from
'material-table';
<MaterialTable
    // other props
    columns={[
      {
        field: 'url',
        title: 'Avatar',
        render: rowData => <img src={rowData.url}
style={{width: 50, borderRadius: '50%'}}/>
      }
    ]}
/>
```

4. Detail Panel

Using the detailPanel functionality, you can add a detail panel to each row. As a function, one detail panel. You can just set a function that returns a React element if you only need one type of detail panel.

```
<MaterialTable
  // other props
  detailPanel={rowData => {
    return (
      <iframe
        width="100%"
        height="315"
        src="https://www.youtube.com/embed/C0DPdy98e4c"
        frameborder="0"
        allow="accelerometer; autoplay; encrypted-media;
gyroscope; picture-in-picture"
        allowfullscreen
      />
    )
  }}
/>
```

5. Editable

Inline editing is available in material-table, allowing users to add, remove, and update new rows. You may also utilize the cellEditable feature to allow users to edit only the data in a single cell.

Editing in the Usage Row

To make table rows editable, use the editable prop, which provides functions for onRowAdd, onRowUpdate, and onRowDelete. A promise must be returned by every function. You can also enable bulk updates by enabling editable. onBulkUpdate.

6. Export

Material-table has an export capability built-in. Only the csv format is supported. To enable export, you must first open the exportButton.

```
import MaterialTable from 'material-table';
<MaterialTable
    // other props
    options={{
      exportButton: true
    }}
/>
```

7. Filtering

The inline filtering function in material-table allows users to filter each column separately.

```
import MaterialTable from 'material-table';
<MaterialTable
    // other props
    options={{
      filtering: true
    }}
/>
```

8. Grouping

The grouping feature in the material table allows users to group data by one or more columns.

To make grouping available, go to settings and select grouping.

```
import MaterialTable from 'material-table';
<MaterialTable
    // other props
    options={{
      grouping: true
    }}
/>
```

9. Localization

To localize material-table, you can use your own text.

```
<MaterialTable
    // other props
    localization={{
        pagination: {
            labelDisplayedRows: '{from}-{to} of {count}'
        },
        toolbar: {
            nRowsSelected: '{0} row(s) selected'
        },
        header: {
            actions: 'Actions'
        },
        body: {
            emptyDataSourceMessage: 'No records to display',
            filterRow: {
                filterTooltip: 'Filter'
            }
        }
    }}
/>
```

The default texts of the datatable could be changed or translated using the localization settings like body, body.addTooltip, body.deleteTooltip, body.editTooltip, body.editRow, pagination, toolbar, header, header.actions, etc.

10. Remote Data

The remote data functionality of material-table allows the user to create a custom data fetching function. Searching, filtering, sorting, and paging are all ignored by material-table while using this feature, and must be done manually.
You must supply a function that returns a Promise containing the data, the current page, and the totalCount instead of providing the data array as prop.data.

```
import MaterialTable from 'material-table';
<MaterialTable
    // other props
    data={query =>
        new Promise((resolve, reject) => {
            // prepare your data and then call resolve like this:
            resolve({
                data: // your data array
                page: // current page number
                totalCount: // total row number
            });
        })
    }
/>;
```

11. Search

The inline search feature in material-table allows users to search a text across all columns.

To make search available, go to settings and select filtering.

```
import MaterialTable from 'material-table';
<MaterialTable
    // other props
    options={{
      search: true
    }}
/>
```

12. Selection

Users can pick several rows in the material-table using the selection feature. You must first open selection in choices to make it available.

```
import MaterialTable from 'material-table';
<MaterialTable
    // other props
    options={{
      selection: true
    }}
/>
```

13. Sorting

The sorting function in material-table allows users to sort data by any column.

To use sorting, go to the settings menu and select Sorting.

```
import MaterialTable from 'material-table';
<MaterialTable
    // other props
    options={{
      sorting: true
    }}
/>
```

14. Styling

To add styles to some components, material-table includes a few styling choices.

15. Tree Data

The tree data feature of material-table can be used to store data with nested relationships.

To make tree data visible, you must first set parentChildData, which locates the data's parent.

```
import MaterialTable from 'material-table';
<MaterialTable
    // other props
    parentChildData={(row, rows) => rows.find(a =>
a.id === row.parentId)}
/>
```

Now Lets use this material table in our CRM app to display the list of admin users in admin.js file in the form of table

1. Create admin.js file in your react project and install material table then import it to the admin.js file

```
import React, { useEffect, useState } from
"react";
import MaterialTable from "@material-table/core";
import { ExportCsv, ExportPdf } from
'@material-table/exporters';
import { Modal, Button } from 'react-bootstrap'

import axios from 'axios';
import '../styles/admin.css';

const BASE_URL =process.env.REACT_APP_SERVER_URL
```

2. fetch user data using axios fetch api

```
const fetchUsers = (userId) => {
    axios.get(BASE_URL + '/crm/api/v1/users/' +
userId, {
        headers: {
            'x-access-token':
localStorage.getItem("token")
        }
    }).then(function (response) {
        if (response.status === 200) {
            if (userId) {
                setUserDetail(response.data[0])
                showUserModal()
            }
```

```
else
                setUserList(response.data);
        }
    })
        .catch(function (error) {
            console.log(error);
        });
  }
```

3. show the users data in form of rows and columns and add the features like export, search and filter on the table content using material table

```
<MaterialTable
  onRowClick={(event, rowData) =>
fetchUsers(rowData.userId)}

  data={userList}
  columns={[
      {
          title: "USER ID",
          field: "userId",
      },
      {
          title: "Name",
          field: "name",

      },
```

```
{
        title: "EMAIL",
        field: "email",
        filtering: false
    },
    {
        title: "ROLE",
        field: "userTypes",
        lookup: {
            "ADMIN": "ADMIN",
            "CUSTOMER": "CUSTOMER",
            "ENGINEER": "ENGINEER",
```

```
},
        },
        {
            title: "Status",
            field: "userStatus",
            lookup: {
                "APPROVED": "APPROVED",
                "PENDING": "PENDING",
                "REJECTED": "REJECTED",
```

```
 },
      },
  ]}
  options={{
      filtering: true,
      sorting: true,
      exportMenu: [{
          label: 'Export PDF',
          exportFunc: (cols, datas) =>
ExportPdf(cols, datas, 'userRecords')
      }, {
          label: 'Export CSV',
          exportFunc: (cols, datas) =>
ExportCsv(cols, datas, 'userRecords')
```

```
 }],
      headerStyle: {
          backgroundColor: 'darkblue',
          color: '#FFF'
        },
        rowStyle: {
          backgroundColor: '#EEE',
        }
  }}
  title="USER RECORDS"
/>
```

Thank You!