1. The api for the countries is http://localhost:5000/api/countries Output of the api is [{"country\_id":"AR","country\_name":"Argentina","region\_id":3},{"country\_id":"AU","country\_name":"Australia","region\_id":5},{"country\_id":"BR","country\_name":"Brazil","region\_id":3},{"country\_id":"CA","country\_name":"Canada","region\_id":2},{"country\_id":"CN","country\_name":"China","region\_id":4},{"country\_id":"DE","country\_name":"Germany","region\_id":1},{"country\_id":"EG","country\_name":"Egypt","region\_id":6},{"country\_id":"ES","country\_name":"Spain","region\_id":1},{"country\_id":"FR","country\_name":"France","region\_id":1},{"country\_id":"IN","country\_name":"India","region\_id":4},{"country\_id":"IT","country\_name":"Italy","region\_id":1},{"country\_id":"JP","country\_name":"Japan","region\_id":4},{"country\_id":"KR","country\_name":"South Korea","region\_id":4},{"country\_id":"MX","country\_name":"Mexico","region\_id":2},{"country\_id":"NL","country\_name":"Netherlands","region\_id":1},{"country\_id":"RU","country\_name":"Russia","region\_id":1},{"country\_id":"SG","country\_name":"Singapore","region\_id":4},{"country\_id":"UK","country\_name":"United Kingdom","region\_id":1},{"country\_id":"US","country\_name":"United States of America","region\_id":2},{"country\_id":"ZA","country\_name":"South Africa","region\_id":6}] 1. create the react component from which we can perform CRUD operation in countries tables. 2. The code should be separate from style code but style and code should be within the component. 3. The should follow the standard project structure for react component. ( routes, context, layouts, pages, features, services, hooks, components, utils ). 4. The table should have pasignation, filter , sorting and export to docx, excel, csv , pdf.

**ChatGPT said:**

Updated saved memory

Here is a complete implementation plan and working example for a **React CRUD component** for the countries table. It includes:

* ✅ Standard project structure
* ✅ Pagination, sorting, filtering
* ✅ Export to CSV, Excel, PDF, DOCX
* ✅ Code and style separated but scoped inside the component

**✅ 1. Project Structure**

css

CopyEdit

src/

├── components/

│ └── CountryTable/

│ ├── CountryTable.jsx

│ └── CountryTable.module.css

├── features/

│ └── countries/

│ ├── CountriesPage.jsx

├── services/

│ └── countryService.js

├── utils/

│ └── exportUtils.js

├── hooks/

│ └── useCountries.js

├── routes/

│ └── AppRoutes.jsx

├── App.jsx

└── main.jsx

**✅ 2. CountryTable.jsx (Component)**

jsx

CopyEdit

// src/components/CountryTable/CountryTable.jsx

import React, { useState, useEffect } from 'react';

import styles from './CountryTable.module.css';

import { fetchCountries, deleteCountry, addCountry, updateCountry } from '../../services/countryService';

import { exportToCSV, exportToPDF, exportToExcel, exportToDOCX } from '../../utils/exportUtils';

const CountryTable = () => {

const [countries, setCountries] = useState([]);

const [filtered, setFiltered] = useState([]);

const [query, setQuery] = useState('');

const [sortKey, setSortKey] = useState('country\_name');

const [sortAsc, setSortAsc] = useState(true);

const [currentPage, setCurrentPage] = useState(1);

const pageSize = 5;

useEffect(() => {

loadCountries();

}, []);

useEffect(() => {

handleFilter(query);

}, [countries]);

const loadCountries = async () => {

const data = await fetchCountries();

setCountries(data);

};

const handleFilter = (text) => {

setQuery(text);

const filteredData = countries.filter(c =>

c.country\_name.toLowerCase().includes(text.toLowerCase())

);

setFiltered(filteredData);

setCurrentPage(1);

};

const handleSort = (key) => {

setSortKey(key);

setSortAsc(!sortAsc);

};

const sortedData = [...filtered].sort((a, b) => {

const aValue = a[sortKey]?.toString().toLowerCase();

const bValue = b[sortKey]?.toString().toLowerCase();

if (aValue < bValue) return sortAsc ? -1 : 1;

if (aValue > bValue) return sortAsc ? 1 : -1;

return 0;

});

const paginated = sortedData.slice((currentPage - 1) \* pageSize, currentPage \* pageSize);

const totalPages = Math.ceil(filtered.length / pageSize);

return (

<div className={styles.container}>

<h2>Countries Management</h2>

<input

type="text"

placeholder="Search by name..."

value={query}

onChange={(e) => handleFilter(e.target.value)}

className={styles.search}

/>

<table className={styles.table}>

<thead>

<tr>

<th onClick={() => handleSort('country\_id')}>ID</th>

<th onClick={() => handleSort('country\_name')}>Name</th>

<th onClick={() => handleSort('region\_id')}>Region</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

{paginated.map((c) => (

<tr key={c.country\_id}>

<td>{c.country\_id}</td>

<td>{c.country\_name}</td>

<td>{c.region\_id}</td>

<td>

<button onClick={() => deleteCountry(c.country\_id).then(loadCountries)}>🗑️</button>

</td>

</tr>

))}

</tbody>

</table>

<div className={styles.pagination}>

{Array.from({ length: totalPages }).map((\_, i) => (

<button key={i} onClick={() => setCurrentPage(i + 1)} className={i + 1 === currentPage ? styles.active : ''}>

{i + 1}

</button>

))}

</div>

<div className={styles.exportButtons}>

<button onClick={() => exportToCSV(countries)}>Export CSV</button>

<button onClick={() => exportToExcel(countries)}>Export Excel</button>

<button onClick={() => exportToPDF(countries)}>Export PDF</button>

<button onClick={() => exportToDOCX(countries)}>Export DOCX</button>

</div>

</div>

);

};

export default CountryTable;

**✅ 3. CountryTable.module.css**

css

CopyEdit

/\* src/components/CountryTable/CountryTable.module.css \*/

.container {

padding: 20px;

max-width: 900px;

margin: auto;

}

.search {

margin-bottom: 10px;

padding: 5px;

width: 300px;

}

.table {

width: 100%;

border-collapse: collapse;

}

.table th, .table td {

border: 1px solid #ccc;

padding: 8px;

}

.table th {

cursor: pointer;

background: #f0f0f0;

}

.pagination button {

margin: 5px;

padding: 5px 10px;

}

.pagination .active {

font-weight: bold;

background-color: #333;

color: white;

}

.exportButtons {

margin-top: 15px;

}

.exportButtons button {

margin-right: 10px;

padding: 6px 12px;

}

**✅ 4. countryService.js**

js

CopyEdit

// src/services/countryService.js

import axios from 'axios';

const API = 'http://localhost:5000/api/countries';

export const fetchCountries = async () => {

const res = await axios.get(API);

return res.data;

};

export const deleteCountry = async (id) => {

await axios.delete(`${API}/${id}`);

};

export const addCountry = async (data) => {

const res = await axios.post(API, data);

return res.data;

};

export const updateCountry = async (id, data) => {

const res = await axios.put(`${API}/${id}`, data);

return res.data;

};

**✅ 5. exportUtils.js**

js

CopyEdit

// src/utils/exportUtils.js

import \* as XLSX from 'xlsx';

import jsPDF from 'jspdf';

import autoTable from 'jspdf-autotable';

import { saveAs } from 'file-saver';

import { Document, Packer, Paragraph, Table, TableCell, TableRow, TextRun } from 'docx';

export const exportToCSV = (data) => {

const ws = XLSX.utils.json\_to\_sheet(data);

const wb = XLSX.utils.book\_new();

XLSX.utils.book\_append\_sheet(wb, ws, 'Countries');

XLSX.writeFile(wb, 'countries.csv');

};

export const exportToExcel = (data) => {

const ws = XLSX.utils.json\_to\_sheet(data);

const wb = XLSX.utils.book\_new();

XLSX.utils.book\_append\_sheet(wb, ws, 'Countries');

XLSX.writeFile(wb, 'countries.xlsx');

};

export const exportToPDF = (data) => {

const doc = new jsPDF();

autoTable(doc, {

head: [['Country ID', 'Country Name', 'Region ID']],

body: data.map(d => [d.country\_id, d.country\_name, d.region\_id])

});

doc.save('countries.pdf');

};

export const exportToDOCX = async (data) => {

const rows = data.map(d => new TableRow({

children: [

new TableCell({ children: [new Paragraph(d.country\_id)] }),

new TableCell({ children: [new Paragraph(d.country\_name)] }),

new TableCell({ children: [new Paragraph(d.region\_id.toString())] })

]

}));

const doc = new Document({

sections: [{ children: [new Table({ rows })] }]

});

const blob = await Packer.toBlob(doc);

saveAs(blob, 'countries.docx');

};

**✅ 6. CountriesPage.jsx**

jsx

CopyEdit

// src/features/countries/CountriesPage.jsx

import React from 'react';

import CountryTable from '../../components/CountryTable/CountryTable';

const CountriesPage = () => {

return (

<div>

<CountryTable />

</div>

);

};

export default CountriesPage;

**✅ 7. AppRoutes.jsx**

jsx

CopyEdit

// src/routes/AppRoutes.jsx

import React from 'react';

import { Routes, Route } from 'react-router-dom';

import CountriesPage from '../features/countries/CountriesPage';

const AppRoutes = () => (

<Routes>

<Route path="/" element={<CountriesPage />} />

</Routes>

);

export default AppRoutes;

**✅ 8. App Setup (App.jsx & main.jsx)**

jsx

CopyEdit

// src/App.jsx

import React from 'react';

import AppRoutes from './routes/AppRoutes';

const App = () => <AppRoutes />;

export default App;

jsx

CopyEdit

// src/main.jsx

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

import { BrowserRouter } from 'react-router-dom';

import './index.css';

ReactDOM.createRoot(document.getElementById('root')).render(

<BrowserRouter>

<App />

</BrowserRouter>

);

**✅ Notes**

* You can add forms for addCountry() and updateCountry() similarly in the CountryTable.
* Required packages:

bash

CopyEdit

npm install axios xlsx jspdf jspdf-autotable file-saver docx