IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 2, December 2022
Impact Factor: 6.252

Modularization of React Data-Tables

Pranav Jadhav[1], Pratapsingh Garud[2], Aditya Kulkarni[3], Vaibhav Kuwar[4], Sanket Salunke[5], Yogesh Sharma[6]
Students, Department of Computer Engineering[1,2,3,4,5,6]
Professor, Department of Computer Engineering[6]
Vishwakarma Institute of Information Technology, VIIT, Pune, Maharashtra, India

Abstract: *React data table is one of the great table provision libraries out there. With more than fifty thousand downloads every week, react data table has proven it provides better functionality than the other available react table libraries. Its highly customisable features help us to bring out the most of it. It offers quick access to features like sorting, pagination, and themes. This paper will demonstrate how we can optimise and modularize this component for a custom use case. This modularized component will provide us with a setup that will be reusable over all our react pages or components. Before all that we will be going through understanding of the react-data-table functionalities. In the later parts of this paper, we will also go through props passing techniques used in the optimisations.*

Keywords: *React JS,* Modularity, Data Tables, Props, Sorting, Pagination, Themes, Optimisations.

I. INTRODUCTION

React-Data-Tables component is available as a package on the npmjs - node package manager (JavaScript). It can be installed in a react using 'npm i react-data-table-component' in power shell.

Consider a case, we are building a react app for a business case where a seller needs information of the customers. We have the data in the database. We need the data to be shown in tables with proper sorting according to the column titles. This sorting should be available seamlessly in ascending as well decreasing order on a single click of the column title.

Let's see how this can implemented using react-datable-component through the following code snippet:

```
import DataTable from "react-data-table-component";
const columns = [
  {
        name: "Name",
        selector: (row) => row.name, sortable: true,
  },
  {
    name: "Email",
    selector: (row) =>
    row.email,
    sortable: true,
  },
];
const data = [
  {
    id: 1,
    name: "Ipsum Lorem",
    email: "lorem47@gmail.com",
  },
  {
    id: 2,
    name: "Amet Elit",
    email: "amet70@gmail.com",
```

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 2, December 2022
Impact Factor: 6.252

```
  },
];

function MyComponent() {
  return <DataTable columns={columns} data={data} />;
}
```

For now, we will be using hard-coded data as declared above
Note: it is important that id of each row to be unique to avoid re-rendering of the elements.

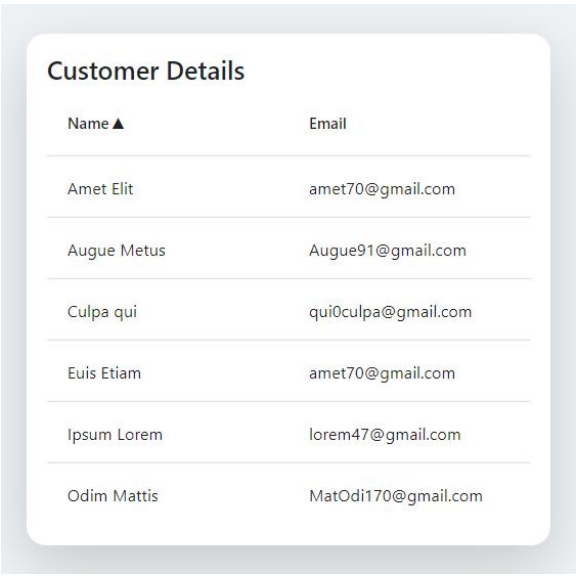


Fig. 1. Features of react-data-table component

II. MODULARIZATION:

Now, consider a case where you must use data tables for multiple components, say for customer list and for a ledger of each customer. Appearance of this table should be similar across all the components hence we need this instance of data table to be reusable. The table would be properly paginated and sortable on click of the column name. Table should be exportable as csv file. All of this can be implemented as following code snippet.

```
import React from "react";
import DataTable from "react-data-table-component";
import { Button } from "@mui/material";
 const DashTable = ({ isLoading, columns, data, paginationPerPage }) => { const Export = ({ onExport }) => (
    <Button onClick={(e) => onExport(e.target.value)}>Export</Button>
  );
  const actionsMemo = React.useMemo(
    () => <Export onExport={() => downloadCSV(data)} />, []
  );
  const [pagesOptions, setPagesOptions] = React.useState([]);
  const convertArrayOfObjectsToCSV = (array) => { let result;
    const columnDelimiter = ","; const lineDelimiter = "\n";
    const keys = Object.keys(data[0]);
    result = "";
    result += keys.join(columnDelimiter); result += lineDelimiter;
    array.forEach((item) => { let ctr = 0; keys.forEach((key) => {
```

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 2, December 2022
Impact Factor: 6.252

```
            if (ctr > 0) result += columnDelimiter; result += item[key];
            ctr++;
          });
          result += lineDelimiter;
});
      return result;
    };
    const downloadCSV = (array) => {
      const link = document.createElement("a");
      let csv = convertArrayOfObjectsToCSV(array); if (csv == null) return;
      const filename = "export.csv";
      if  (!csv.match(/^data:text\/csv/i))  {
        csv  =  `data:text/csv;charset=utf-8,${csv}`;
      }

      link.setAttribute("href", encodeURI(csv)); link.setAttribute("download", filename); link.click();
    };

    React.useEffect(() => { const options = [];
      for (var i = 1; i <= 6; i++) { options.push(paginationPerPage * i);
      }
      setPagesOptions(options);
    }, []);
    return (
      <DataTable  columns={columns} data={data} progressPending={isLoading} progressComponent={
          <div
            className="spinner-border"
            style={{ width: "3rem", height: "3rem", color: "#5e72e4" }} role="status"
          ></div>
        }
        pagination paginationPerPage={paginationPerPage}
          //[3] paginationRowsPerPageOptions={pagesOptions}   actions={actionsMemo}
highlightOnHover

      />
    );
  };
  export default DashTable;
```

We use the following snippet to implement the dashtable in a component.

```
  <DashTable    isLoading={isCustomerDataLoading}    columns={customerDataHeaders}    data={filteredCustomerData}
    paginationPerPage={10}
  />;
```

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 2, December 2022
Impact Factor: 6.252

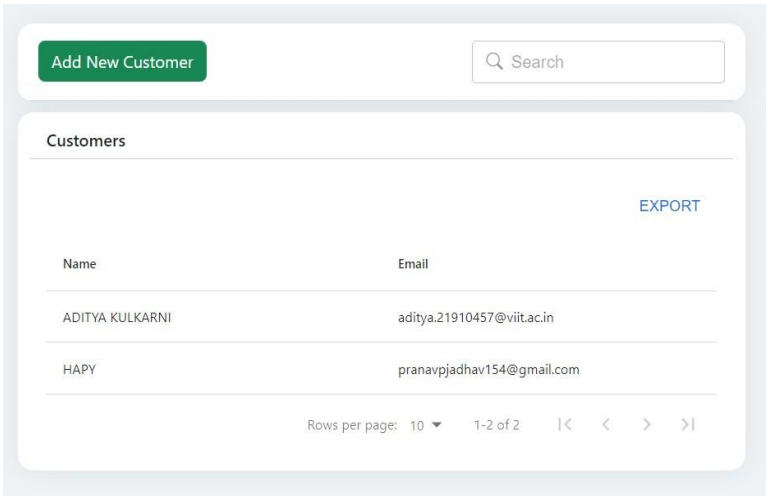


Fig. 2. Visualisation of the rendered customer table

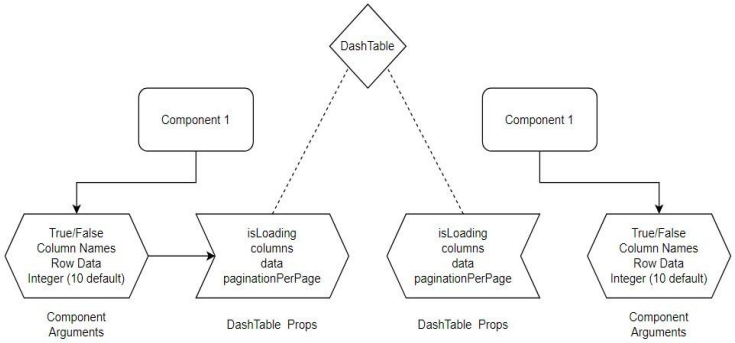
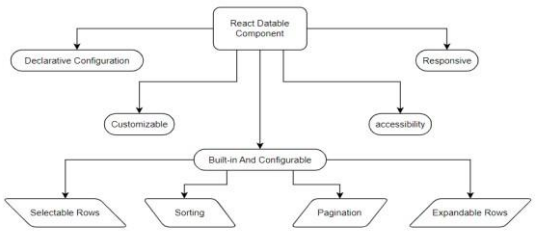


Fig. 3. Props passing mechanism in DashTable

III. CONCLUSION



. Fig. 4. Features of react-data-table component

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 2, December 2022
Impact Factor: 6.252

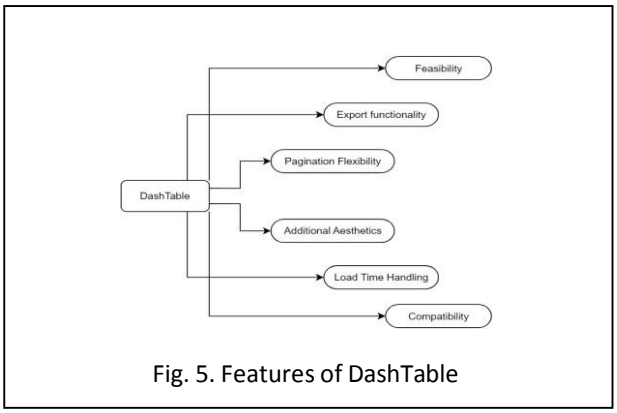


Fig. 5. Features of DashTable

Thus, approach will increase modularity of a react-webpage. It offers several additional features to the general data tables. Above diagram represent the features of the react-data-table component and a diagram of features of dash table. Dashtable offers extensive features added and modularised onto the datatable. This modularised version of datatable ensures provision of features efficiently. With props passing mechanism it provides all the functionality in minimum number of lines, which otherwise would have caused repetition. Dashtable also provides a boost to performance due to reduction in the number of state and prop variables [4] and since Dashtable is a independent variable. Dashtable can be called by other components to avail its functionality. [4]

REFERENCES

[1] "Key Features and Installation", npmjs-software registry, https://www.npmjs.com/package/react-data-table-component

[2] "Basic Examples", react-data-table-component docs, https://react-data-table-component.netlify.app/?path=/docs/getting-started-examples--page

[3] "Pagination implementation", react-data-table-component docs, https://react-data-table-component.netlify.app/?path=/docs/pagination-remote--remote

[4] Arshad Javeed, "Performance Optimization Techniques for ReactJS", IEEE, 2019