Project Title – CSV READER

Project Description -

CSV Reader is a user-friendly desktop application designed to simplify the process of exploring and visualizing CSV (Comma-Separated Values) files. Built on the React frameworks, this application provides an intuitive interface for users to effortlessly select, load, and view CSV files from their local filesystem.
For the front end, React JS Framework was used. This project has the following ReactJS components -

App Component: This single ReactJS component allows users to upload their CSV files. Based on the given CSV File, it will create a table out of it. The component uses ***papaparse*** library for parsing CSV data. Pagination is done in the component to seamlessly access CSV data.

At the start, the Choose File option is available through the input method of HTML. If no file is selected, an alert is generated that no file is selected. After the successful upload of the file, a table with records per page table is generated. Users can navigate to different records using the Previous and Next Page buttons.

How to run the Project?

Use *npm install -I <dependency_name>* to install the following dependencies used in the project -

Client Side -

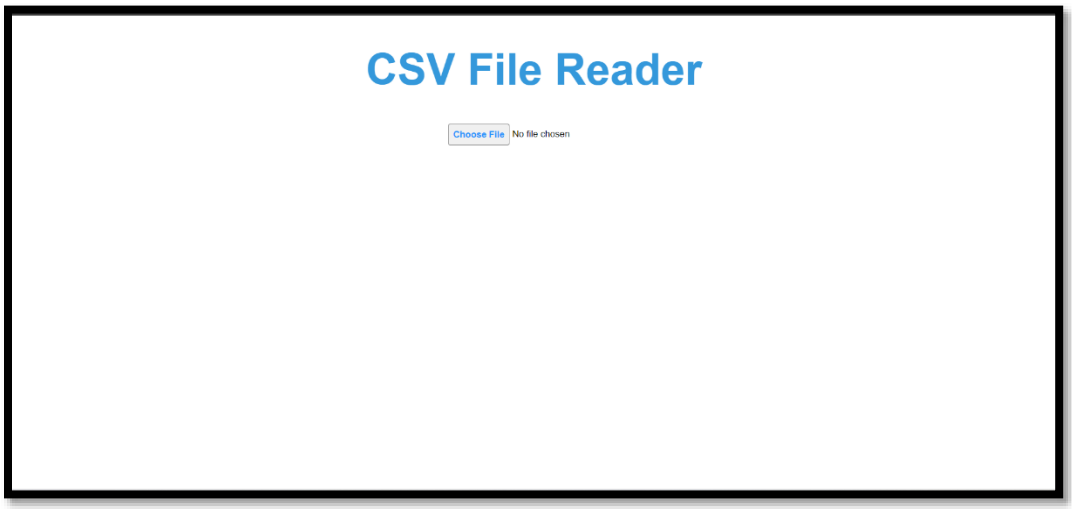
1. **React -** The react package contains only the functionality necessary to define React components.
2. **React-dom -** This package serves as the entry point to the DOM and server renderers for React.
3. **Papaparse -** Fast and powerful CSV parser for the browser that supports web workers and streaming large files. Converts CSV to JSON and JSON to CSV.
4. **React-toastify:** The React-Toastify package enables developers to add toast notifications to their applications and also can set notifications and warnings. It also

allows us to display toasts to users that contain messages and information for a set amount of time.

To Run the Project –

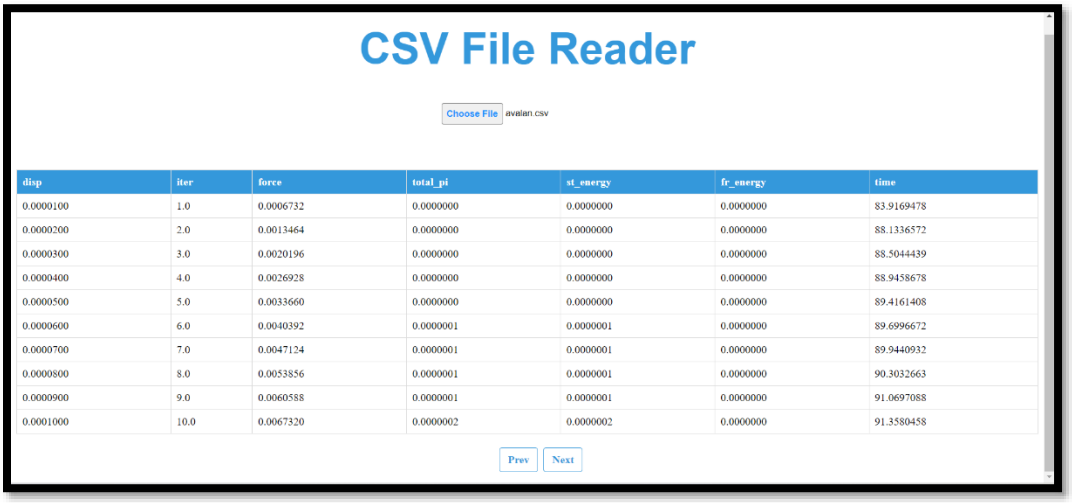
- Open the terminal and write npm run start to start the app .
- App is running on "http://localhost:3000" .
- The page must display CSV Reader Heading followed by a Choose File Button.
- Click Choose File to select the CSV File to open and read its contents on the app.

Landing Page -



- After Clicking on Choose File button, a dialog box will appear asking to select the file to open.

- The dialog box asks you to upload CSV file to access the data.
- After selecting the file, the information will be displayed as follows –



disp	iter	force	total_pi	st_energy	fr_energy	time
0.0000100	1.0	0.0006732	0.0000000	0.0000000	0.0000000	83.9169478
0.0000200	2.0	0.0013464	0.0000000	0.0000000	0.0000000	88.1336572
0.0000300	3.0	0.0020196	0.0000000	0.0000000	0.0000000	88.5044439
0.0000400	4.0	0.0026928	0.0000000	0.0000000	0.0000000	88.9458678
0.0000500	5.0	0.0033660	0.0000000	0.0000000	0.0000000	89.4161408
0.0000600	6.0	0.0040392	0.0000001	0.0000001	0.0000000	89.6996672
0.0000700	7.0	0.0047124	0.0000001	0.0000001	0.0000000	89.9440932
0.0000800	8.0	0.0053856	0.0000001	0.0000001	0.0000000	90.3032663
0.0000900	9.0	0.0060588	0.0000001	0.0000001	0.0000000	91.0697088
0.0001000	10.0	0.0067320	0.0000002	0.0000002	0.0000000	91.3580458

- A table displaying all the CSV data will be displayed.
- Clicking through Next and Previous Page, user can easily navigate to different section of records.

Code Breakdown (App.js file) –

Import Statements

```
import "./App.css";
import { useState } from "react";
import Papa from "papaparse";
import { ToastContainer, toast } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
```

Imports necessary for the component include the styling file (App.css), React's useState hook, Papa from the papaparse library for CSV parsing, and ToastContainer and toast from react-toastify for displaying error messages.

React Functional Component

```
function App() {
  // State to store parsed data
  const [parsedData, setParsedData] = useState([]);

  // State to store table Column name
  const [tableRows, setTableRows] = useState([]);

  // State to store the values
  const [values, setValues] = useState([]);
```

Three state variables (parsedData, tableRows, values) are declared using the useState hook to manage the state of parsed CSV data, table column names, and values, respectively.

File Upload and Error Handling Functions

```
// Error message when no file selected
const fileError = () => {
  toast.error('No file chosen', {
    // ...toast configuration options
  });
};

// File selection function handler
const changeHandler = (event) => {
  // ... (File parsing logic)
};
```

fileError: Displays an error toast message when no file is selected.

changeHandler: Handles the file selection event, parses the selected CSV file using papaparse, and updates the state variables with parsed data, table column names, and values.

Error Handling Toast Functions

```
const prevPageError = () => {
  toast.error('You are already on Page 1', {
    // ...toast configuration options
  });
};

const nextPageError = () => {
  toast.error('No More Records to show!', {
    // ...toast configuration options
  });
};
```

prevPageError: Displays an error toast message when attempting to go to a previous page when already on Page 1.

nextPageError: Displays an error toast message when attempting to go to the next page when there are no more records to show.

Pagination Variables and Functions

Pagination Variables

```
const [currentPage, setCurrentPage] = useState(1);
const recordsPerPage = 10;
const lastIndex = currentPage * recordsPerPage;
const firstIndex = lastIndex - recordsPerPage;
const records = values.slice(firstIndex, lastIndex);

function prePage() {
  // ... (Function to navigate to the previous page with error handling)
}

function nextPage() {
  // ... (Function to navigate to the next page with error handling)
}
```

Pagination-related state variables and functions to navigate to the previous and next pages are defined.

JSX - UI Rendering

```
return (
  <div>
    {/* Page Head */}
    <h1> CSV File Reader </h1>
```

```
    {/* File Uploader */}
    <input
      type="file"
      name="file"
      id="fileInput"
      onChange={changeHandler}
      accept=".csv"
      style={{ display: "block", margin: "10px auto" }}
      data-file-name={values.length > 0 ? values[0].name : ""}
    />

    {/* Table */}
    <table>
      {/* ... (Table header and body rendering) */}
    </table>

    {values.length > 0 && (
      <nav>
        <ul className="pagination">
          {/* ... (Pagination controls) */}
        </ul>
      </nav>
    )}
    <ToastContainer/>
  </div>
 );
}
```

The main component function returns JSX for rendering the UI, including a file input, table for displaying CSV data, pagination controls, and error toast notifications.

Conclusion

The provided code defines a React component for a CSV Reader App, incorporating functionality for file uploading, parsing, error handling, pagination, and UI rendering. It utilizes the papaparse library for CSV parsing and react-toastify for displaying error messages.