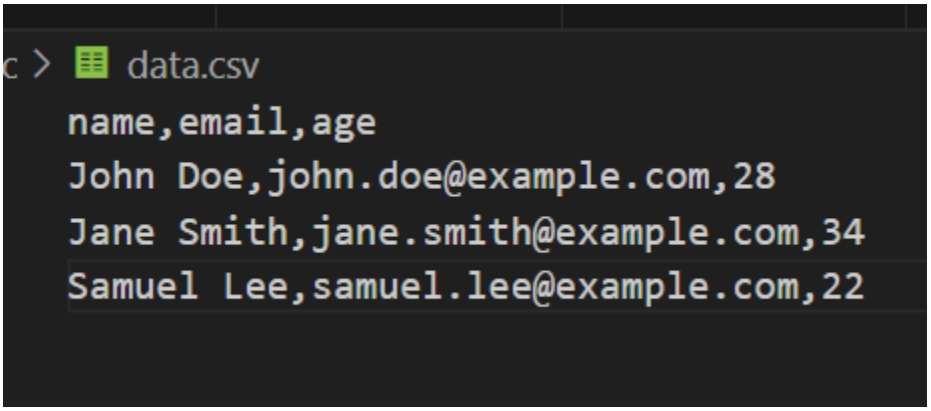
How To Read Csv File In React Js :-

CSV Data

name	email	age
John Doe	john.doe@example.com	28
Jane Smith	jane.smith@example.com	34
Samuel Lee	samuel.lee@example.com	22



To fetch data from a CSV file in a React application, you can use libraries like `papaparse` for parsing the CSV file into JavaScript objects. Here's how you can implement data fetching from a CSV file in React:

Step 1: Install `papaparse`

First, install `papaparse` to parse CSV files:

`cmd (terminal):-`

```
npm install papaparse
```

Step 2: Create a React Component to Fetch and Parse CSV Data

Here is an example of how you can use React to fetch a CSV file, parse it using `papaparse`, and display the data in a table.

Write code for file CsvDataFetcher.js

```
import React, { useState, useEffect } from 'react';
import Papa from 'papaparse';

function CsvDataFetcher() {
  const [csvData, setCsvData] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  useEffect(() => {
    // Fetch the CSV file and parse it using papaparse
    fetch('/data.csv') // Specify the path to your CSV file
      .then((response) => response.text())
      .then((data) => {
        Papa.parse(data, {
          header: true, // Treat the first row as headers
          dynamicTyping: true, // Automatically cast values like numbers,
booleans
          complete: (result) => {
            setCsvData(result.data); // Set the parsed data in the state
            setLoading(false); // Set loading to false once data is fetched
          },
          error: (error) => {
            setError('Error parsing CSV: ' + error.message);
            setLoading(false); // Set loading to false if there's an error
          },
        });
      })
      .catch((error) => {
        setError('Error fetching CSV: ' + error.message);
        setLoading(false); // Set loading to false if the fetch fails
      });
  }, []); // Empty dependency array means this runs only once, when the component
mounts

  if (loading) {
    return <p>Loading...</p>;
  }
```

```
  if (error) {
    return <p>{error}</p>;
  }

  return (
    <div>
      <h1>CSV Data</h1>
      <table border="1" style={{ width: '100%', textAlign: 'left' }}>
        <thead>
          <tr>
            {/* Assuming CSV file has headers */}
            {csvData[0] && Object.keys(csvData[0]).map((key) => <th
key={key}>{key}</th>)}
          </tr>
        </thead>
        <tbody>
          {csvData.map((row, index) => (
            <tr key={index}>
              {Object.values(row).map((value, i) => (
                <td key={i}>{value}</td>
              ))}
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
}

export default CsvDataFetcher;
```

Step 3: Use the Component in Your App

Now, you can use the `CsvDataFetcher` component in your `App.js`:

`jsx code:-`

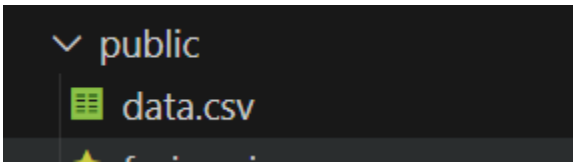
```
import React from 'react';
import CsvDataFetcher from './CsvDataFetcher';

function App() {
  return (
    <div className="App">
      <CsvDataFetcher />
    </div>
  );
}

export default App;
```

Step 4: Add the CSV File to Public Directory

Ensure that your CSV file is accessible to the app by placing it in the `public` folder of your React app. The `public` folder is where assets like images, styles, and CSV files are stored. For example, you can place your file in `public/data.csv`



Step 5: Run the Application

Now you can run your React app:

cmd terminal:-

```
npm start
```

Explanation:

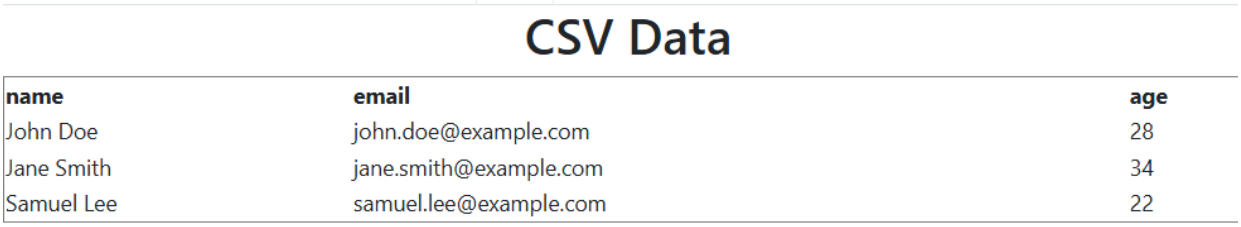
1. **`Papa.parse()`**:
   - `Papa.parse()` is used to parse the CSV content into a JavaScript object.
   - `header: true` indicates that the first row contains headers, so the keys of the resulting objects will be taken from the header row.
   - `dynamicTyping: true` automatically converts values to their appropriate types (e.g., numbers, booleans).
2. **Error and Loading State**:
   - While the CSV file is being fetched and parsed, the `loading` state is `true`, and a loading message is shown.
   - If there is an error (either in fetching or parsing), the `error` state will display the error message.
3. **Displaying the Data**:
   - After successfully parsing the CSV, the data is displayed in a table.
   - The `Object.keys(csvData[0])` dynamically creates the table headers from the CSV header row.
   - The table body is generated by mapping through `csvData` and displaying the values of each row.

CSV File Example (`data.csv`):

`csv file code:-`

```
name,email,age
John Doe,john.doe@example.com,28
Jane Smith,jane.smith@example.com,34
Samuel Lee,samuel.lee@example.com,22
```

When the CSV file is successfully parsed and displayed, the output will look like this:



CSV Data

name	email	age
John Doe	john.doe@example.com	28
Jane Smith	jane.smith@example.com	34
Samuel Lee	samuel.lee@example.com	22