React Excel File Editing

how to change a excel file with react only

To manipulate an Excel file directly in React (using only the front end), you can use a JavaScript library like `xlsx`. This library allows you to read, modify, and write Excel files in a browser environment, which is ideal for React.

Here's how you can set it up and manipulate an Excel file:

1. Install the `xlsx` library

Run the following command to add the `xlsx` library to your project:

```bash
npm install xlsx
```

2. Create a File Input and Read the Excel File

You can use a file input to allow the user to upload an Excel file, then read it with `xlsx`.

```jsx
import React, { useState } from 'react';
import * as XLSX from 'xlsx';

function ExcelEditor() {
  const [data, setData] = useState([]);

  // Function to handle file upload
  const handleFileUpload = (e) => {
    const file = e.target.files[0];
    const reader = new FileReader();

    reader.onload = (event) => {
      const binaryStr = event.target.result;
      const workbook = XLSX.read(binaryStr, { type: 'binary' });
      const firstSheetName = workbook.SheetNames[0];
      const worksheet = workbook.Sheets[firstSheetName];
      const jsonData = XLSX.utils.sheet_to_json(worksheet);
      setData(jsonData);
    };
```

```
    reader.readAsBinaryString(file);
  };

  return (
    <div>
      <input type="file" onChange={handleFileUpload} />
      <pre>{JSON.stringify(data, null, 2)}</pre>
    </div>
  );
}

export default ExcelEditor;
```

3. Modify the Data and Export the Excel File

Once the data is loaded into a JSON format, you can modify it as needed, and then export it back to an Excel file.

```jsx
// Function to modify data and export as Excel file
const exportToExcel = () => {
  // Make some changes to data
  const updatedData = data.map(row => ({
    ...row,
    NewColumn: 'New Data', // Add a new column with example data
  }));

  // Convert JSON data to worksheet
  const worksheet = XLSX.utils.json_to_sheet(updatedData);
  const workbook = XLSX.utils.book_new();
  XLSX.utils.book_append_sheet(workbook, worksheet, 'Sheet1');

  // Export the file
  XLSX.writeFile(workbook, 'UpdatedFile.xlsx');
};
```

4. Add the Export Button to the Component

Update the component to include the export button:

```jsx
```

```
return (
  <div>
    <input type="file" onChange={handleFileUpload} />
    <button onClick={exportToExcel}>Export Modified File</button>
    <pre>{JSON.stringify(data, null, 2)}</pre>
  </div>
);
```

Full Component Code

Here's the complete example with both file upload and export functionality:

```jsx
import React, { useState } from 'react';
import * as XLSX from 'xlsx';

function ExcelEditor() {
  const [data, setData] = useState([]);

  const handleFileUpload = (e) => {
    const file = e.target.files[0];
    const reader = new FileReader();

    reader.onload = (event) => {
      const binaryStr = event.target.result;
      const workbook = XLSX.read(binaryStr, { type: 'binary' });
      const firstSheetName = workbook.SheetNames[0];
      const worksheet = workbook.Sheets[firstSheetName];
      const jsonData = XLSX.utils.sheet_to_json(worksheet);
      setData(jsonData);
    };

    reader.readAsBinaryString(file);
  };

  const exportToExcel = () => {
    const updatedData = data.map(row => ({
      ...row,
      NewColumn: 'New Data',
    }));
```

```
    const worksheet = XLSX.utils.json_to_sheet(updatedData);
    const workbook = XLSX.utils.book_new();
    XLSX.utils.book_append_sheet(workbook, worksheet, 'Sheet1');
    XLSX.writeFile(workbook, 'UpdatedFile.xlsx');
  };

  return (
    <div>
      <input type="file" onChange={handleFileUpload} />
      <button onClick={exportToExcel}>Export Modified File</button>
      <pre>{JSON.stringify(data, null, 2)}</pre>
    </div>
  );
}

export default ExcelEditor;
```

Explanation

1. **File Input**: Allows users to upload an Excel file.
2. **File Reader**: Reads the file content and converts it to a workbook using `xlsx`.
3. **Data Manipulation**: Modify the JSON data as needed.
4. **Export Functionality**: Converts the modified JSON back to Excel format and downloads it.

This approach works directly in the React front end without needing any back-end processing.