Excel to JSON Convert Documentation

1. At first library package should be installed :

```
“npm install xlsx”,
“npm install  export-from-json”,
“npm install react-icons”
“npm install react-toastify”
“npm install antd
```

2. Import Library component

```
import React, { useEffect, useState } from "react";

import * as XLSX from "xlsx";

import exportFromJSON from "export-from-json";

import { FaArrowAltCircleDown } from "react-icons/fa";

import { GrFormView } from "react-icons/gr";

import { ToastContainer, toast } from "react-toastify";

import "react-toastify/dist/ReactToastify.css";

import { Button, Modal } from "antd";
```

3. Create State :

```
const [file, setFile] = useState("");
const [items, setItems] = useState([]);
const [editingData, setEditingData] = useState(null);
  const [isEditing, setIsEditing] = useState(false);
  const [isModalOpen, setIsModalOpen] = useState(false);
```

Uploaded xlsx file will be in ‘file’ state &
Converted Json file will be in ‘items’ state &
After Editing json data will be stored in editingData

4.Create input for excel file upload:

```
<label className="convert_input_label">
        {file ? file.name.substring(0, 19) : "Upload xl file"}
        <input
          type="file"
          name="file"
        onChange={(e) => {
          const files = e.target.files[0];
          readExcel(files);
          setFile(e.target.files[0]);
        }}
        accept=".xls, .xlsx"
        className="convert_xl_btn "
        hidden
      />
    </label>
```

5. Convert xl to json file by ‘readExcel’ function & set in ‘items’ state

```
  const readExcel = (file) => {
    const promise = new Promise((resolve, reject) => {
      const fileReader = new FileReader();
      fileReader.readAsArrayBuffer(file);

      fileReader.onload = (e) => {
        const bufferArray = e.target.result;
const wb = XLSX.read(bufferArray, { type: "buffer" });
        const wsname = wb.SheetNames[0];

        const ws = wb.Sheets[wsname];

        const data = XLSX.utils.sheet_to_json(ws);

        resolve(data);
      };

      fileReader.onerror = (error) => {
        reject(error);
      };
    });

    promise.then((d) => {
      setItems(d);
    });
  };
```

6. Display This converted items in a table:

```
      const isURL = (str) => {
        try {
          new URL(str);
          return true;
        } catch (error) {
            return false;
          }
        };
<div className="convert_table_container">
          {items && items.length > 0 ? (
            <div className="convert_table_padding_container">
              <div className="convert_table_div ">
                <table className="convert_table">
                  <thead className="coverted_table_head">
                    <tr>
                      {Object.keys(items[0]).map((header) => (
                        <th key={header}>{header}</th>
                      ))}
                    </tr>
                  </thead>
                  <tbody>
                    {items.map((row, index) => (
                      <tr key={index}>
                        {Object.values(row).map((cell, cellIndex) => (
                          <td key={cellIndex}>
                            {isURL(cell) ? (
                              <img
                                src={cell}
                                alt="Image"
                                style={{
                                  maxWidth: "15vw",
                                  maxHeight: "11vh",
                                }}
                              />
                            ) : (
                              cell
                            )}
                          </td>
                        ))}
                      </tr>
                    ))}
                  </tbody>
                </table>
              </div>
            </div>
          ) : (
            <p>No data available</p>
          )}
        </div>
```

7. Make two Button for Download Converted JSON & CSV File

```
<div className="convert_download_div">
 {" "}
 {items.length > 0 && (
  <>
   <button
    onClick={() => setIsModalOpen(true)}
    className="convert_download_btn"
   >
    <label htmlFor="json" className="json_label">
     VIEW JSON
    </label>
    <GrFormView />
   </button>
   <button
    onClick={handleJSONDownload}
    className="convert_download_btn"
   >
    <label htmlFor="json" className="json_label">
     JSON
    </label>
    <FaArrowAltCircleDown />
   </button>
   <button
    onClick={handleCsvDownload}
    className="convert_download_btn"
   >
    <label htmlFor="json" className="json_label">
     CSV
    </label>
    <FaArrowAltCircleDown />
   </button>
  </>
 )}
</div>
```

8. Create & Download Json File by 'handleJSONDownload' function :

```
const handleJSONDownload = () => {
 const jsonContent = JSON.stringify(items, null, 2);
 const blob = new Blob([jsonContent], { type: "application/json" });
 const url = URL.createObjectURL(blob);
 const a = document.createElement("a");
 a.href = url;
 a.download = "excel_data.json";
```

```
  document.body.appendChild(a);
  a.click();
  document.body.removeChild(a);
  URL.revokeObjectURL(url);
};
```

9. Create & Download CSV File by ‘handleCsvDownload’ function :

```
const handleCsvDownload = () => {
  const data = items;
  const fileName = "excel_data";
  const exportType = exportFromJSON.types.csv;

  exportFromJSON({ data, fileName, exportType });
};
```

10. Json data will be shown in the modal :

```
 <div className="modal_div">
<Modal
  title="JSON DATA"
  open={isModalOpen}
  onOk={handleOk}
  onCancel={handleCancel}
  style={{
    // maxWidth: '100%',
    // margin: 0,
    // marginLeft: 0,
    // overflow: 'hidden',
    top: 120,
    bottom: 0,
    left: 0,
    right: 0,
    // width:'100%',
    // height: '100%',
    height: "80vh",
    width: "70vw",
    display: "flex",
    // position: "fixed",
    // alignItems: "center",
    justifyContent: "center",
  }}
  footer={null}
>
  <div className="json_container">
    <div className="json_data">
      {/* <h4 className="json_text">JSON DATA</h4> */}
      <div className="json_button">
        {isEditing ? (
                    <button onClick={handleSaveClick} className="save_btn">
                      Save Changes
                    </button>
                  ) : (
                    <>
                      <button onClick={handleEditClick} className="edit_btn">
                        EDIT DATA
                      </button>
                      <button onClick={handleCopyClick} className="copy_btn">
                        COPY DATA
                      </button>
                    </>
                  )}
                </div>
                <div className="json_field">
                  {isEditing ? (
                    <textarea
                      value={editingData}
                      onChange={handleInputChange}
                      rows={14}
                      className="editing_json_text"
                    />
                  ) : (
                    <pre>{JSON.stringify(items, null, 2)}</pre>
                  )}
                </div>
              </div>
            </div>
          </Modal>
        </div>
```

11.Function for handling modal:

```
const handleOk = () => {
  setIsModalOpen(false);
};
const handleCancel = () => {
  setIsModalOpen(false);
};
```

12.Function for handling copy edit & save data:

```
const handleEditClick = () => {

  // Make a copy of currentData to editingData for editing

  setEditingData(JSON.stringify(items, null, 2));

  setIsEditing(true);
```

```
};

const handleSaveClick = () => {
  try {
    // Save the edited data to currentData after parsing the JSON
    const editedData = JSON.parse(editingData);
    // Note: You might want to perform additional validation before saving
    setItems(editedData);
    setEditingData(null);
    setIsEditing(false);
    toast("Changes saved!", { autoClose: 1500 });
    setIsModalOpen(false);
  } catch (error) {
    toast("Error parsing JSON. Please make sure the input is valid.", {
      autoClose: 1500,
    });
  }
};

const handleInputChange = (e) => {
  // Update the editingData based on user input
  setEditingData(e.target.value);
};

const handleCopyClick = () => {
  // Convert the currentData to a JSON-formatted string
  const jsonString = JSON.stringify(items, null, 2);

  // Copy the JSON string to the clipboard
  navigator.clipboard.writeText(jsonString).then(() => {
    toast(" Data copied to clipboard!", { autoClose: 1500 });
```

```
   });
   setIsModalOpen(false);
  };
```

13. Finally use the ToastContainer inside the main div to get testify notification:

```
<ToastContainer />
```