import React, { useState } from 'react';

import Papa  from 'papaparse';

import TransactionDataService from "../services/transaction.service";

import Mapping from "./Mapping";

export default function Upload(props) {

    const [csvfile, setCsvFile] = useState(undefined);

    const [showUploadButton, setShowUploadButton] = useState();

    const [showMappingDoneButton, setShowMappingDoneButton] = useState();

    const [message, setMessage] = useState("");

    const [fileKeys, setFileKeys] = useState([]);

    const account\_id = props.match.params.id;

    const nick\_name = props.match.params.nick\_name;

    const transForDB = [];

    console.log("--- In Upload");

    function handleFileChange(event) {

        console.log("--- In handleFileChange");

        var csvfile = getFileName(event);

        console.log("csvfile in handleFileChange:");

        console.log(csvfile);

        var rawTransactions = readFile(csvfile);

        console.log("rawTransactions:");

        console.log(rawTransactions);

    };

    function getFileName(event) {

        console.log("--- In getFileName");

        var filename = "(None chosen)";

        var fileextension = '';

        if (event.target.files !== undefined && event.target.files.length > 0) {

            filename = event.target.files[0].name;

            fileextension = filename.split('.').pop();

        };

        if (fileextension !== 'csv') {

            alert("Must be a CSV file.  " + filename + " is not.");

            setCsvFile(undefined);

            setShowUploadButton(false);

            setMessage("Must be a CSV file.  " + filename + " is not.  Please choose another file.");

        } else {

            setCsvFile(event.target.files[0]);

            setShowUploadButton(true);

            setMessage("Choose columns from file for each database field.  When done, click 'Upload Transactions'.");

        };

        return event.target.files[0];

    };

    function readFile(csvfile) {

        console.log("--- In readFile");

        var results = Papa.parse(csvfile, {

            complete: readData,

            header: true

        });

        console.log("in readFile, results (rawTransactions):");

        console.log(results);

        return results;

    };

    function importCSV() {

        console.log("--- In importCSV");

        console.log("csvfile in importCSV:");

        console.log(csvfile);

        var results = Papa.parse(csvfile, {

            complete: updateData,

            header: true

        });

    };

    function readData(result) {

        console.log("--- In readData");

        // go through array in reverse order, so index in result.errors is correct.

        // remove invalid transaction records

        for (var i = result.errors.length-1; i >= 0; i--) {

            result.data.splice(result.errors[i].row, 1);

        };

        const transForDB = result.data.filter(trans => trans.Amount !== "");

        setShowMappingDoneButton(true);

        // var stmt\_date;

        // var formatForDB = [];

        // transForDB.forEach(trans => {

        //     stmt\_date = new Date(trans["Stmt Date"]);

        //     var DBtrans = {

        //         account\_id: account\_id,

        //         trans\_date: new Date(trans.Date),

        //         post\_date: (trans.Date !== "") ? new Date(trans.Date) : null,

        //         verified: (trans["Stmt Date"] !== null) ? "Yes" : "",

        //         amount: (trans["Transaction Type"] === 'debit') ? 0-trans.Amount : trans.Amount,

        //         to\_from: trans.Description,

        //         description: trans.Notes,

        //         category: trans.Category,

        //         stmt\_date: new Date(stmt\_date.getFullYear(), stmt\_date.getMonth(), 1),

        //     }

        //     formatForDB.push(DBtrans);

        // });

        // TransactionDataService.postTransactions(formatForDB)

        // .then(response => {

        //     setMessage(`${formatForDB.length} transactions uploaded to the database.`);

        // })

        // .catch(e => {

        //     console.log(e);

        // });

    };

    function updateData(result) {

        console.log("--- In updateData");

        // go through array in reverse order, so index in result.errors is correct.

        // remove invalid transaction records

        for (var i = result.errors.length-1; i >= 0; i--) {

            result.data.splice(result.errors[i].row, 1);

        };

        const transForDB = result.data.filter(trans => trans.Amount !== "");

        var stmt\_date;

        var formatForDB = [];

        transForDB.forEach(trans => {

            stmt\_date = new Date(trans["Stmt Date"]);

            var DBtrans = {

                account\_id: account\_id,

                trans\_date: new Date(trans.Date),

                post\_date: (trans.Date !== "") ? new Date(trans.Date) : null,

                verified: (trans["Stmt Date"] !== null) ? "Yes" : "",

                amount: (trans["Transaction Type"] === 'debit') ? 0-trans.Amount : trans.Amount,

                to\_from: trans.Description,

                description: trans.Notes,

                category: trans.Category,

                stmt\_date: new Date(stmt\_date.getFullYear(), stmt\_date.getMonth(), 1),

            }

            formatForDB.push(DBtrans);

        });

        TransactionDataService.postTransactions(formatForDB)

        .then(response => {

            setMessage(`${formatForDB.length} transactions uploaded to the database.`);

        })

        .catch(e => {

            console.log(e);

        });

    };

    return (

        <div>

            <h1 className="text-center">

                Upload transactions for <u>{`${nick\_name}`}</u>

            </h1>

            <h3 className = "text-center">Click "Choose file" to choose a CSV file to upload transactions from.

                After a CSV file has been chosen, an 'Upload files' button will appear to allow you to upload transactions.</h3>

            <br />

            <br />

            <p className="text-danger bg-warning text-center"><strong>{message}</strong></p>

            <div className="row">

                <div className="col-2 offset-4">

                    <input

                        className="csv-input ml-5"

                        type="file"

                        name = "file"

                        placeholder = {null}

                        onChange = {handleFileChange}

                    />

                    <br />

                    { showUploadButton ?

                        <div>

                            <Mapping

                                fileKeys={fileKeys}

                                transForDB={transForDB}

                            />

                            <button className="m-5" onClick={importCSV}>Upload transactions</button>

                        </div>

                        : null

                    }

                </div>

            </div>

        </div>

    );

}