import React, { useState, useEffect, useRef } from 'react';

import { PDFDocument } from 'pdf-lib';

import { DragDropContext, Droppable, Draggable } from 'react-beautiful-dnd';

import \* as pdfjsLib from 'pdfjs-dist';

import 'pdfjs-dist/build/pdf.worker.entry';

const PdfRearranger = () => {

const [pdfDoc, setPdfDoc] = useState(null);

const [pages, setPages] = useState([]);

const [pageImages, setPageImages] = useState([]);

const handleFileUpload = async (event) => {

const file = event.target.files[0];

const arrayBuffer = await file.arrayBuffer();

const pdfDoc = await PDFDocument.load(arrayBuffer);

setPdfDoc(pdfDoc);

setPages(Array.from({ length: pdfDoc.getPageCount() }, (\_, i) => i));

await renderPdfPages(arrayBuffer, pdfDoc.getPageCount());

};

const renderPdfPages = async (arrayBuffer, pageCount) => {

const loadingTask = pdfjsLib.getDocument({ data: arrayBuffer });

const pdf = await loadingTask.promise;

const images = [];

for (let i = 1; i <= pageCount; i++) {

const page = await pdf.getPage(i);

const viewport = page.getViewport({ scale: 1.5 });

const canvas = document.createElement('canvas');

const context = canvas.getContext('2d');

canvas.height = viewport.height;

canvas.width = viewport.width;

await page.render({ canvasContext: context, viewport }).promise;

images.push(canvas.toDataURL());

}

setPageImages(images);

};

const handleRearrange = async () => {

if (!pdfDoc) return;

const newPdfDoc = await PDFDocument.create();

for (let i = 0; i < pages.length; i++) {

const pageIndex = pages[i];

const [copiedPage] = await newPdfDoc.copyPages(pdfDoc, [pageIndex]);

newPdfDoc.addPage(copiedPage);

}

const pdfBytes = await newPdfDoc.save();

const blob = new Blob([pdfBytes], { type: 'application/pdf' });

const url = URL.createObjectURL(blob);

const link = document.createElement('a');

link.href = url;

link.download = 'rearranged.pdf';

document.body.appendChild(link);

link.click();

document.body.removeChild(link);

};

const onDragEnd = (result) => {

if (!result.destination) return;

const reorderedPages = Array.from(pages);

const [removed] = reorderedPages.splice(result.source.index, 1);

reorderedPages.splice(result.destination.index, 0, removed);

setPages(reorderedPages);

};

return (

<div>

<input type="file" accept="application/pdf" onChange={handleFileUpload} />

<button onClick={handleRearrange} disabled={!pdfDoc}>Rearrange PDF</button>

<DragDropContext onDragEnd={onDragEnd}>

<Droppable droppableId="pages">

{(provided) => (

<div {...provided.droppableProps} ref={provided.innerRef} style={{ display: 'flex', flexDirection: 'column', gap: '10px' }}>

{pages.map((pageIndex, index) => (

<Draggable key={pageIndex} draggableId={pageIndex.toString()} index={index}>

{(provided) => (

<div

ref={provided.innerRef}

{...provided.draggableProps}

{...provided.dragHandleProps}

style={{

userSelect: 'none',

padding: '8px',

margin: '0 0 8px 0',

minHeight: '50px',

backgroundColor: '#fff',

color: '#333',

border: '1px solid #ddd',

...provided.draggableProps.style,

}}

>

<img src={pageImages[pageIndex]} alt={`Page ${pageIndex + 1}`} style={{ width: '100%' }} />

</div>

)}

</Draggable>

))}

{provided.placeholder}

</div>

)}

</Droppable>

</DragDropContext>

</div>

);

};

export default PdfRearranger;