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

import { PDFDocument } from 'pdf-lib';

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

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

import Cropper from 'react-cropper';

import 'cropperjs/dist/cropper.css';

import './PdfRearranger.css';

const PdfRearranger = () => {

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

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

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

const [selectedPage, setSelectedPage] = useState(null);

const [rotationAngles, setRotationAngles] = useState([]);

const [croppedImage, setCroppedImage] = useState(null);

const cropperRef = useRef(null);

const handleFileUpload = async (event) => {

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

if (!file) return;

try {

const arrayBuffer = await file.arrayBuffer();

const pdfDoc = await PDFDocument.load(arrayBuffer);

setPdfDoc(pdfDoc);

const pageCount = pdfDoc.getPageCount();

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

setRotationAngles(Array(pageCount).fill(0)); // Initialize rotation angles

await renderPdfPages(arrayBuffer, pageCount);

} catch (error) {

console.error('Error loading PDF:', error);

}

};

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

try {

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);

} catch (error) {

console.error('Error rendering PDF pages:', error);

}

};

const handleRearrange = async () => {

if (!pdfDoc) return;

try {

const newPdfDoc = await PDFDocument.create();

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

const pageIndex = pages[i];

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

copiedPage.setRotation(rotationAngles[pageIndex] % 360);

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);

} catch (error) {

console.error('Error creating PDF:', error);

}

};

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);

};

const rotateSelectedPage = (angle) => {

if (selectedPage === null) return;

const newRotationAngles = [...rotationAngles];

newRotationAngles[selectedPage] = (newRotationAngles[selectedPage] + angle) % 360;

setRotationAngles(newRotationAngles);

};

const handleCrop = () => {

const cropper = cropperRef.current.cropper;

const croppedDataUrl = cropper.getCroppedCanvas().toDataURL();

setCroppedImage(croppedDataUrl);

// Update the pageImages with the cropped image

const newPageImages = [...pageImages];

newPageImages[selectedPage] = croppedDataUrl;

setPageImages(newPageImages);

};

return (

<div className="pdf-rearranger">

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

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

<div className="grid-container">

<div className="column">

<DragDropContext onDragEnd={onDragEnd}>

<Droppable droppableId="pages">

{(provided) => (

<div {...provided.droppableProps} ref={provided.innerRef}>

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

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

{(provided) => (

<div

ref={provided.innerRef}

{...provided.draggableProps}

{...provided.dragHandleProps}

onClick={() => setSelectedPage(pageIndex)}

style={{

userSelect: 'none',

padding: '8px',

margin: '0 0 8px 0',

backgroundColor: selectedPage === pageIndex ? '#d3d3d3' : '#fff',

color: '#333',

border: '1px solid #ddd',

cursor: 'pointer',

...provided.draggableProps.style,

}}

>

Page {pageIndex + 1}

</div>

)}

</Draggable>

))}

{provided.placeholder}

</div>

)}

</Droppable>

</DragDropContext>

</div>

<div className="column">

{selectedPage !== null && (

<div>

<Cropper

src={pageImages[selectedPage]}

style={{ height: 400, width: '100%' }}

initialAspectRatio={1}

guides={false}

ref={cropperRef}

/>

<div style={{ display: 'flex', gap: '10px', marginTop: '10px' }}>

<button onClick={() => rotateSelectedPage(90)}>Rotate 90°</button>

<button onClick={() => rotateSelectedPage(-90)}>Rotate -90°</button>

<button onClick={handleCrop}>Crop</button>

</div>

{croppedImage && <img src={croppedImage} alt="Cropped" style={{ width: '100%', marginTop: '10px' }} />}

</div>

)}

</div>

<div className="column">

{pageImages.map((image, index) => (

<div key={index} style={{ marginBottom: '10px' }}>

<img

src={image}

alt={`Page ${index + 1}`}

style={{

width: '100%',

transform: `rotate(${rotationAngles[index]}deg)`

}}

/>

</div>

))}

</div>

</div>

</div>

);

};

export default PdfRearranger;