import React, { useState } from "react";

import { Document } from "@/entities/all";

import { UploadFile, ExtractDataFromUploadedFile } from "@/integrations/Core";

import { Dialog, DialogContent, DialogHeader, DialogTitle } from "@/components/ui/dialog";

import { Button } from "@/components/ui/button";

import { Input } from "@/components/ui/input";

import { Label } from "@/components/ui/label";

import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select";

import { Upload, FileText, Image, Loader2, CheckCircle } from "lucide-react";

import { Alert, AlertDescription } from "@/components/ui/alert";

const DOCUMENT\_CATEGORIES = [

{ value: "contract", label: "Contract" },

{ value: "correspondence", label: "Correspondence" },

{ value: "evidence", label: "Evidence" },

{ value: "court\_filing", label: "Court Filing" },

{ value: "research", label: "Research" },

{ value: "form", label: "Form" },

{ value: "other", label: "Other" }

];

export default function DocumentUpload({ open, onOpenChange, caseId, onDocumentUploaded }) {

const [file, setFile] = useState(null);

const [title, setTitle] = useState("");

const [category, setCategory] = useState("");

const [isUploading, setIsUploading] = useState(false);

const [uploadStep, setUploadStep] = useState("select"); // select, uploading, processing, complete

const [error, setError] = useState("");

const MAX\_FILE\_SIZE = 10 \* 1024 \* 1024; // 10MB in bytes

const handleFileSelect = (e) => {

const selectedFile = e.target.files[0];

if (selectedFile) {

// Validate file size

if (selectedFile.size > MAX\_FILE\_SIZE) {

setError(`File size (${(selectedFile.size / 1024 / 1024).toFixed(1)}MB) exceeds the 10MB limit. Please choose a smaller file.`);

setFile(null); // Clear selected file if too large

setTitle(""); // Clear title

return;

}

// Validate file type

const allowedTypes = ['pdf', 'png', 'jpg', 'jpeg', 'docx', 'txt'];

const fileExtension = selectedFile.name.split('.').pop().toLowerCase();

if (!allowedTypes.includes(fileExtension)) {

setError("File type not supported. Please use PDF, DOCX, TXT, or image files.");

setFile(null); // Clear selected file if unsupported type

setTitle(""); // Clear title

return;

}

setFile(selectedFile);

setTitle(selectedFile.name.replace(/\.[^/.]+$/, ""));

setError(""); // Clear any previous errors if selection is valid

} else {

// If no file is selected (e.g., user cancels file dialog), clear current selection and error

setFile(null);

setTitle("");

setError("");

}

};

const getFileType = (file) => {

const extension = file.name.split('.').pop().toLowerCase();

if (extension === 'pdf') return 'pdf';

if (['jpg', 'jpeg', 'png'].includes(extension)) return 'image';

if (extension === 'docx') return 'docx';

if (extension === 'txt') return 'txt';

return 'other';

};