import React, { useState } from "react";

import { InvokeLLM } from "@/integrations/Core";

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

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

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

import { Textarea } from "@/components/ui/textarea";

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

import { Badge } from "@/components/ui/badge";

import { FileText, Download, Loader2, PenTool } from "lucide-react";

const DOCUMENT\_TYPES = [

{ value: "demand\_letter", label: "Demand Letter", description: "Formal request for payment or action" },

{ value: "contract", label: "Contract", description: "Legal agreement between parties" },

{ value: "motion", label: "Motion", description: "Court filing requesting specific action" },

{ value: "affidavit", label: "Affidavit", description: "Sworn statement of facts" },

{ value: "cease\_desist", label: "Cease and Desist", description: "Letter demanding cessation of activity" },

{ value: "settlement\_agreement", label: "Settlement Agreement", description: "Agreement to resolve dispute" },

{ value: "privacy\_policy", label: "Privacy Policy", description: "Website/app privacy policy" },

{ value: "terms\_of\_service", label: "Terms of Service", description: "User agreement for services" },

{ value: "employment\_contract", label: "Employment Contract", description: "Agreement between employer and employee" },

{ value: "nda", label: "Non-Disclosure Agreement", description: "Confidentiality agreement" }

];

export default function GeneratorPage() {

const [documentType, setDocumentType] = useState("");

const [formData, setFormData] = useState({

parties: "",

details: "",

jurisdiction: "",

special\_requirements: ""

});

const [generatedDocument, setGeneratedDocument] = useState(null);

const [isGenerating, setIsGenerating] = useState(false);

const handleInputChange = (field, value) => {

setFormData(prev => ({

...prev,

[field]: value

}));

};

const handleGenerate = async () => {

if (!documentType || !formData.details) return;

setIsGenerating(true);

try {

const selectedType = DOCUMENT\_TYPES.find(t => t.value === documentType);

const prompt = `You are a professional legal document drafting AI. Please draft a comprehensive ${selectedType.label} with the following specifications:

Document Type: ${selectedType.label}

Description: ${selectedType.description}

Parties Involved: ${formData.parties || "Not specified"}

Jurisdiction: ${formData.jurisdiction || "Not specified"}

Details and Requirements: ${formData.details}

Special Requirements: ${formData.special\_requirements || "None"}

Please create a professional, legally sound document that includes:

1. Proper legal formatting and structure

2. All necessary clauses and provisions

3. Clear and precise legal language

4. Placeholder sections for signatures and dates

5. Standard legal disclaimers where appropriate

Format the document with proper headings, numbered sections, and professional legal styling. Include [BRACKETED PLACEHOLDERS] for information that needs to be customized.

IMPORTANT: Include a disclaimer that this document should be reviewed by a qualified attorney before use.`;

const response = await InvokeLLM({

prompt,

add\_context\_from\_internet: false

});

setGeneratedDocument({

type: selectedType.label,

content: response,

timestamp: new Date(),

formData: { ...formData }

});

} catch (error) {

console.error("Error generating document:", error);

}

setIsGenerating(false);

};

const handleDownload = () => {

if (!generatedDocument) return;

const element = document.createElement("a");

const file = new Blob([generatedDocument.content], { type: 'text/plain' });

element.href = URL.createObjectURL(file);

element.download = `${generatedDocument.type.replace(/\s+/g, '\_')}\_${Date.now()}.txt`;

document.body.appendChild(element);

element.click();

document.body.removeChild(element);

};

return (

<div className="min-h-screen bg-gradient-to-br from-slate-50 to-blue-50 p-6">

<div className="max-w-6xl mx-auto">

<div className="mb-8">

<h1 className="text-3xl font-bold text-[var(--navy)] mb-2">Document Generator</h1>

<p className="text-slate-600">Generate professional legal documents with AI assistance</p>

</div>

<div className="grid lg:grid-cols-2 gap-8">

{/\* Document Generation Form \*/}

<Card className="bg-white/80 backdrop-blur-sm border-0 shadow-lg">

<CardHeader>

<CardTitle className="flex items-center gap-2 text-xl text-[var(--navy)]">

<PenTool className="w-6 h-6" />

Document Builder

</CardTitle>

</CardHeader>

<CardContent className="space-y-6">

<div className="space-y-2">

<label className="text-sm font-medium text-slate-700">Document Type</label>

<Select value={documentType} onValueChange={setDocumentType}>

<SelectTrigger className="border-slate-200 focus:border-[var(--gold)] focus:ring-[var(--gold)]">

<SelectValue placeholder="Select document type" />

</SelectTrigger>

<SelectContent>

{DOCUMENT\_TYPES.map((type) => (

<SelectItem key={type.value} value={type.value}>

<div>

<div className="font-medium">{type.label}</div>

<div className="text-xs text-slate-500">{type.description}</div>

</div>

</SelectItem>

))}

</SelectContent>

</Select>

</div>

<div className="space-y-2">

<label className="text-sm font-medium text-slate-700">Parties Involved</label>

<Input

value={formData.parties}

onChange={(e) => handleInputChange("parties", e.target.value)}

placeholder="e.g., John Smith and ABC Company"

className="border-slate-200 focus:border-[var(--gold)] focus:ring-[var(--gold)]"

/>

</div>

<div className="space-y-2">

<label className="text-sm font-medium text-slate-700">Jurisdiction</label>

<Input

value={formData.jurisdiction}

onChange={(e) => handleInputChange("jurisdiction", e.target.value)}

placeholder="e.g., California, Federal"

className="border-slate-200 focus:border-[var(--gold)] focus:ring-[var(--gold)]"

/>

</div>

<div className="space-y-2">

<label className="text-sm font-medium text-slate-700">Details and Requirements \*</label>

<Textarea

value={formData.details}

onChange={(e) => handleInputChange("details", e.target.value)}

placeholder="Describe the specific details, terms, conditions, and requirements for this document..."

rows={6}

className="border-slate-200 focus:border-[var(--gold)] focus:ring-[var(--gold)]"

/>

</div>

<div className="space-y-2">

<label className="text-sm font-medium text-slate-700">Special Requirements</label>

<Textarea

value={formData.special\_requirements}

onChange={(e) => handleInputChange("special\_requirements", e.target.value)}

placeholder="Any special clauses, conditions, or requirements..."

rows={3}

className="border-slate-200 focus:border-[var(--gold)] focus:ring-[var(--gold)]"

/>

</div>

<Button

onClick={handleGenerate}

disabled={!documentType || !formData.details || isGenerating}

className="w-full bg-gradient-to-r from-[var(--navy)] to-[var(--navy-light)] hover:shadow-lg transition-all duration-300"

size="lg"

>

{isGenerating ? (

<>

<Loader2 className="w-5 h-5 mr-2 animate-spin" />

Generating Document...

</>

) : (

<>

<FileText className="w-5 h-5 mr-2" />

Generate Document

</>

)}

</Button>

</CardContent>

</Card>

{/\* Generated Document Preview \*/}

<Card className="bg-white/80 backdrop-blur-sm border-0 shadow-lg">

<CardHeader>

<div className="flex items-center justify-between">

<CardTitle className="flex items-center gap-2 text-xl text-[var(--navy)]">

<FileText className="w-6 h-6" />

Generated Document

</CardTitle>

{generatedDocument && (

<Button

onClick={handleDownload}

variant="outline"

size="sm"

className="flex items-center gap-2"

>

<Download className="w-4 h-4" />

Download

</Button>

)}

</div>

{generatedDocument && (

<div className="flex flex-wrap gap-2">

<Badge variant="outline">{generatedDocument.type}</Badge>

<Badge variant="outline" className="text-xs">

{generatedDocument.timestamp.toLocaleString()}

</Badge>

</div>

)}

</CardHeader>

<CardContent>

{generatedDocument ? (

<div className="space-y-4">

<div className="bg-amber-50 border border-amber-200 rounded-lg p-3">

<p className="text-sm text-amber-800">

<strong>Important:</strong> This generated document should be reviewed by a qualified attorney before use.

Legal requirements vary by jurisdiction and situation.

</p>

</div>

<div className="bg-slate-50 rounded-lg p-4 max-h-96 overflow-auto">

<pre className="whitespace-pre-wrap text-sm text-slate-700 font-mono leading-relaxed">

{generatedDocument.content}

</pre>

</div>

</div>

) : (

<div className="text-center py-16">

<div className="w-16 h-16 mx-auto mb-6 bg-gradient-to-br from-[var(--navy)] to-[var(--navy-light)] rounded-2xl flex items-center justify-center">

<FileText className="w-8 h-8 text-[var(--gold)]" />

</div>

<h3 className="text-xl font-semibold text-[var(--navy)] mb-2">Document Preview</h3>

<p className="text-slate-600 max-w-md mx-auto">

Select a document type and fill in the required details to generate your legal document.

The AI will create a professional, customizable template.

</p>

</div>

)}

</CardContent>

</Card>

</div>

{/\* Document Types Grid \*/}

<Card className="mt-8 bg-white/80 backdrop-blur-sm border-0 shadow-lg">

<CardHeader>

<CardTitle className="text-xl text-[var(--navy)]">Available Document Types</CardTitle>

</CardHeader>

<CardContent>

<div className="grid md:grid-cols-2 lg:grid-cols-3 gap-4">

{DOCUMENT\_TYPES.map((type) => (

<div

key={type.value}

className={`p-4 rounded-lg border-2 cursor-pointer transition-all duration-200 ${

documentType === type.value

? 'border-[var(--gold)] bg-[var(--gold)]/10'

: 'border-slate-200 hover:border-slate-300'

}`}

onClick={() => setDocumentType(type.value)}

>

<h4 className="font-medium text-[var(--navy)] mb-1">{type.label}</h4>

<p className="text-sm text-slate-600">{type.description}</p>

</div>

))}

</div>

</CardContent>

</Card>

</div>

</div>

);

}