import \* as React from "react";

import { format, parse, isValid } from "date-fns";

import { CalendarIcon } from "lucide-react";

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

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

import { Calendar } from "@/components/ui/calendar";

import {

Popover,

PopoverContent,

PopoverTrigger,

} from "@/components/ui/popover";

import { cn } from "@/lib/utils";

interface DateInputProps {

value: Date | undefined;

onChange: (date: Date | undefined) => void;

placeholder?: string;

className?: string;

}

export function DateInput({ value, onChange, placeholder = "MM/DD/YYYY", className }: DateInputProps) {

const [inputValue, setInputValue] = React.useState<string>(

value ? format(value, "MM/dd/yyyy") : ""

);

// Update the input value when the value prop changes

React.useEffect(() => {

if (value) {

setInputValue(format(value, "MM/dd/yyyy"));

} else {

setInputValue("");

}

}, [value]);

const handleInputChange = (e: React.ChangeEvent<HTMLInputElement>) => {

const newInputValue = e.target.value;

setInputValue(newInputValue);

// Try to parse the date if it matches MM/DD/YYYY format (with or without leading zeros)

const dateRegex = /^(\d{1,2})\/(\d{1,2})\/(\d{4})$/;

const match = newInputValue.match(dateRegex);

if (match) {

const [\_, month, day, year] = match;

// Create date string in format that parse can understand

const dateString = `${month.padStart(2, '0')}/${day.padStart(2, '0')}/${year}`;

const parsedDate = parse(dateString, "MM/dd/yyyy", new Date());

if (isValid(parsedDate)) {

console.log("Valid date parsed from input:", parsedDate);

onChange(parsedDate);

}

}

};

return (

<div className={cn("flex gap-2", className)}>

<Input

placeholder={placeholder}

value={inputValue}

onChange={handleInputChange}

className="flex-grow"

/>

<Popover>

<PopoverTrigger asChild>

<Button

variant="outline"

className={cn("px-2")}

type="button"

>

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

</Button>

</PopoverTrigger>

<PopoverContent className="w-auto p-0" align="end">

<Calendar

mode="single"

selected={value}

onSelect={(date) => {

console.log("Date selected from calendar:", date);

onChange(date);

if (date) {

setInputValue(format(date, 'MM/dd/yyyy'));

}

}}

initialFocus

className={cn("p-3 pointer-events-auto")}

/>

</PopoverContent>

</Popover>

</div>

);

}

import React from 'react';

import { useForm } from "react-hook-form";

import { z } from "zod";

import { zodResolver } from "@hookform/resolvers/zod";

import {

Form,

FormControl,

FormField,

FormItem,

FormLabel,

FormMessage,

} from "@/components/ui/form";

import {

Select,

SelectContent,

SelectItem,

SelectTrigger,

SelectValue,

} from "@/components/ui/select";

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

import { CurrencyInput } from '@/components/CurrencyInput';

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

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

import { DateInput } from '@/components/DateInput';

// Form schema using Zod for validation

const formSchema = z.object({

category: z.string().min(1, "Category is required"),

name: z.string().min(1, "Name is required"),

amount: z.number().min(0, "Amount cannot be negative"),

frequency: z.enum(["weekly", "biweekly", "monthly", "quarterly", "annually", "once"]).default("monthly"),

dueDate: z.date().optional(),

});

export type ExpenseFormValues = z.infer<typeof formSchema>;

// Payment frequency options

export const frequencyOptions = [

{ value: "once", label: "One Time" },

{ value: "weekly", label: "Weekly" },

{ value: "biweekly", label: "Biweekly" },

{ value: "monthly", label: "Monthly" },

{ value: "quarterly", label: "Quarterly" },

{ value: "annually", label: "Annually" },

];

// Expense categories

export const expenseCategories = [

{ value: "rent\_or\_mortgage", label: "Rent or Mortgage" },

{ value: "utilities", label: "Utilities" },

{ value: "mobile\_phone", label: "Mobile Phone" },

{ value: "internet", label: "Internet" },

{ value: "grocery", label: "Grocery" },

{ value: "loan\_1", label: "Loan 1" },

{ value: "loan\_2", label: "Loan 2" },

{ value: "loan\_3", label: "Loan 3" },

{ value: "daycare", label: "Daycare" },

{ value: "subscriptions", label: "Subscriptions" },

{ value: "child\_support", label: "Child Support" },

{ value: "auto\_gas", label: "Auto Gas" },

{ value: "auto\_payment", label: "Auto Payment" },

{ value: "auto\_insurance", label: "Auto Insurance" },

{ value: "health\_insurance", label: "Health Insurance" },

{ value: "restaurant\_meals", label: "Restaurant Meals" },

{ value: "personal\_care", label: "Personal Care" },

{ value: "bus\_fare", label: "Bus Fare" },

{ value: "fraternity\_sorority", label: "Fraternity/Sorority" },

{ value: "rideshare", label: "Rideshare" },

{ value: "cash\_to\_family", label: "Cash to Family" },

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

];

interface ExpenseFormProps {

onSubmit: (data: ExpenseFormValues) => void;

isSubmitting: boolean;

onCancel: () => void;

}

const ExpenseForm: React.FC<ExpenseFormProps> = ({ onSubmit, isSubmitting, onCancel }) => {

// Initialize form with react-hook-form

const form = useForm<ExpenseFormValues>({

resolver: zodResolver(formSchema),

defaultValues: {

category: "",

name: "",

amount: 0,

frequency: "monthly",

dueDate: undefined,

},

});

// Handle form submission - ensure dueDate is properly passed

const handleSubmit = (formData: ExpenseFormValues) => {

console.log('Expense form submitting with data:', formData);

// Ensure dueDate is passed correctly and is a Date object if it exists

const submissionData = {

...formData,

dueDate: formData.dueDate instanceof Date ? formData.dueDate : undefined

};

onSubmit(submissionData);

};

return (

<Card className="w-[90%] max-w-[500px] bg-white shadow-lg">

<Form {...form}>

<CardContent className="pt-6">

<form onSubmit={form.handleSubmit(handleSubmit)} className="space-y-6">

{/\* Category Selection \*/}

<FormField

control={form.control}

name="category"

render={({ field }) => (

<FormItem>

<FormLabel>Category</FormLabel>

<Select

onValueChange={field.onChange}

defaultValue={field.value}

>

<FormControl>

<SelectTrigger>

<SelectValue placeholder="Select a category" />

</SelectTrigger>

</FormControl>

<SelectContent>

{expenseCategories.map((category) => (

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

{category.label}

</SelectItem>

))}

</SelectContent>

</Select>

<FormMessage />

</FormItem>

)}

/>

{/\* Expense Name \*/}

<FormField

control={form.control}

name="name"

render={({ field }) => (

<FormItem>

<FormLabel>Expense Name</FormLabel>

<FormControl>

<Input {...field} placeholder="e.g., Rent, Netflix" />

</FormControl>

<FormMessage />

</FormItem>

)}

/>

{/\* Amount \*/}

<FormField

control={form.control}

name="amount"

render={({ field }) => (

<FormItem>

<FormLabel>Amount</FormLabel>

<FormControl>

<CurrencyInput

value={field.value}

onChange={field.onChange}

placeholder="$0.00"

/>

</FormControl>

<FormMessage />

</FormItem>

)}

/>

{/\* Frequency \*/}

<FormField

control={form.control}

name="frequency"

render={({ field }) => (

<FormItem>

<FormLabel>How often do you pay this expense?</FormLabel>

<Select

onValueChange={field.onChange}

defaultValue={field.value}

>

<FormControl>

<SelectTrigger>

<SelectValue placeholder="Select frequency" />

</SelectTrigger>

</FormControl>

<SelectContent>

{frequencyOptions.map((option) => (

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

{option.label}

</SelectItem>

))}

</SelectContent>

</Select>

<FormMessage />

</FormItem>

)}

/>

{/\* Due Date \*/}

<FormField

control={form.control}

name="dueDate"

render={({ field }) => (

<FormItem>

<FormLabel>Due Date</FormLabel>

<FormControl>

<DateInput

value={field.value}

onChange={(date) => {

console.log('Date selected in DateInput component:', date);

field.onChange(date);

}}

/>

</FormControl>

<FormMessage />

</FormItem>

)}

/>

<div className="flex gap-2 mt-6">

<Button

type="button"

variant="outline"

onClick={onCancel}

className="flex-1"

>

Cancel

</Button>

<Button

type="submit"

className="flex-1"

disabled={isSubmitting}

>

{isSubmitting ? "Saving..." : "Save Expense"}

</Button>

</div>

</form>

</CardContent>

</Form>

</Card>

);

};

export default ExpenseForm;