import { OpenAIStream, StreamingTextResponse } from "ai" import { Configuration, OpenAIApi } from "openai-edge"

const config = new Configuration({ apiKey: process.env.OPENAI\_API\_KEY, }) const openai = new OpenAIApi(config)

export async function POST(req: Request) { const { messages } = await req.json()

const response = await openai.createChatCompletion({ model: "gpt-4", stream: true, messages: [ { role: "system", content: "You are an AI investment assistant for the Philippines. Help investors find opportunities and navigate government processes. Provide accurate information about regions, incentives, and procedures.", }, ...messages, ], })

const stream = OpenAIStream(response) return new StreamingTextResponse(stream) }

import { NextResponse } from "next/server" import { OpenAIStream } from "ai" import { Configuration, OpenAIApi } from "openai-edge"

const config = new Configuration({ apiKey: process.env.OPENAI\_API\_KEY, }) const openai = new OpenAIApi(config)

export async function POST(req: Request) { try { const { requirement, regions } = await req.json()

// Create a prompt for the AI to analyze the match  
const prompt = `  
 Analyze the following business requirement and regional profiles to find the best matches:  
  
 Business Requirement:  
 ${JSON.stringify(requirement, null, 2)}  
  
 Regional Profiles:  
 ${JSON.stringify(regions, null, 2)}  
  
 Provide a detailed analysis of the top 5 matching regions, including:  
 1. Match score (0-100)  
 2. Key matching factors  
 3. Potential challenges  
 4. Recommendations  
`  
  
const response = await openai.createChatCompletion({  
 model: "gpt-4",  
 messages: [  
 {  
 role: "system",  
 content:  
 "You are an AI investment advisor specializing in regional development in the Philippines. Analyze business requirements and regional profiles to find optimal matches.",  
 },  
 {  
 role: "user",  
 content: prompt,  
 },  
 ],  
 stream: true,  
})  
  
const stream = OpenAIStream(response)  
return new Response(stream)

} catch (error) { console.error("Error in matching:", error) return NextResponse.json({ error: "Failed to process matching request" }, { status: 500 }) } }

import { NextResponse } from "next/server"

import type { MatchResult } from "@/types"

export async function POST(req: Request) {

try {

const { match, escalation } = await req.json()

// Send email notifications

await sendMatchNotification(match, escalation)

return NextResponse.json({ success: true })

} catch (error) {

console.error("Error in notification:", error)

return NextResponse.json({ error: "Failed to send notifications" }, { status: 500 })

}

}

async function sendMatchNotification(match: MatchResult, escalation: boolean) {

// Implement email sending logic here

// Use your preferred email service (SendGrid, AWS SES, etc.)

console.log("Sending notification for match:", match.id)

console.log("Escalation:", escalation)

}

@tailwind base;

@tailwind components;

@tailwind utilities;

@layer base {

:root {

--background: 0 0% 100%;

--foreground: 222.2 84% 4.9%;

--card: 0 0% 100%;

--card-foreground: 222.2 84% 4.9%;

--popover: 0 0% 100%;

--popover-foreground: 222.2 84% 4.9%;

--primary: 222.2 47.4% 11.2%;

--primary-foreground: 210 40% 98%;

--secondary: 210 40% 96.1%;

--secondary-foreground: 222.2 47.4% 11.2%;

--muted: 210 40% 96.1%;

--muted-foreground: 215.4 16.3% 46.9%;

--accent: 210 40% 96.1%;

--accent-foreground: 222.2 47.4% 11.2%;

--destructive: 0 84.2% 60.2%;

--destructive-foreground: 210 40% 98%;

--border: 214.3 31.8% 91.4%;

--input: 214.3 31.8% 91.4%;

--ring: 222.2 84% 4.9%;

--radius: 0.5rem;

}

.dark {

--background: 222.2 84% 4.9%;

--foreground: 210 40% 98%;

--card: 222.2 84% 4.9%;

--card-foreground: 210 40% 98%;

--popover: 222.2 84% 4.9%;

--popover-foreground: 210 40% 98%;

--primary: 210 40% 98%;

--primary-foreground: 222.2 47.4% 11.2%;

--secondary: 217.2 32.6% 17.5%;

--secondary-foreground: 210 40% 98%;

--muted: 217.2 32.6% 17.5%;

--muted-foreground: 215 20.2% 65.1%;

--accent: 217.2 32.6% 17.5%;

--accent-foreground: 210 40% 98%;

--destructive: 0 62.8% 30.6%;

--destructive-foreground: 210 40% 98%;

--border: 217.2 32.6% 17.5%;

--input: 217.2 32.6% 17.5%;

--ring: 212.7 26.8% 83.9%;

}

}

@layer base {

\* {

@apply border-border;

}

body {

@apply bg-background text-foreground;

}

}

import { Suspense } from "react"

import BusinessRequirementsForm from "@/components/business-requirements-form"

import { Card } from "@/components/ui/card"

import type { BusinessRequirement } from "@/types"

export default function Page() {

return (

<main className="container mx-auto p-6">

<div className="mx-auto max-w-4xl">

<h1 className="mb-8 text-3xl font-bold">Philippines Regional Investment Matching</h1>

<Suspense fallback={<Card className="h-[400px] animate-pulse" />}>

<BusinessRequirementsForm

onSubmit={async (data: BusinessRequirement) => {

console.log("Form submitted:", data)

// Implement form submission and matching logic

}}

/>

</Suspense>

</div>

</main>

)

}

import type \* as React from "react"

import { cva, type VariantProps } from "class-variance-authority"

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

const badgeVariants = cva(

"inline-flex items-center rounded-full border px-2.5 py-0.5 text-xs font-semibold transition-colors focus:outline-none focus:ring-2 focus:ring-ring focus:ring-offset-2",

{

variants: {

variant: {

default: "border-transparent bg-primary text-primary-foreground hover:bg-primary/80",

secondary: "border-transparent bg-secondary text-secondary-foreground hover:bg-secondary/80",

destructive: "border-transparent bg-destructive text-destructive-foreground hover:bg-destructive/80",

outline: "text-foreground",

},

},

defaultVariants: {

variant: "default",

},

},

)

export interface BadgeProps extends React.HTMLAttributes<HTMLDivElement>, VariantProps<typeof badgeVariants> {}

function Badge({ className, variant, ...props }: BadgeProps) {

return <div className={cn(badgeVariants({ variant }), className)} {...props} />

}

export { Badge, badgeVariants }

"use client"

import type { TooltipProps } from "recharts"

export function ChartTooltip({ active, payload, label }: TooltipProps<number, string>) {

if (!active || !payload) return null

return (

<div className="rounded-lg border bg-background p-2 shadow-sm">

<div className="grid grid-cols-2 gap-2">

<div className="flex flex-col">

<span className="text-[0.70rem] uppercase text-muted-foreground">{label}</span>

{payload.map((item) => (

<span key={item.name} className="font-bold">

{item.name}: {item.value}

</span>

))}

</div>

</div>

</div>

)

}

export function ChartTooltipContent({ active, payload, label }: TooltipProps<number, string>) {

if (!active || !payload) return null

return (

<div className="rounded-lg border bg-background p-2 shadow-sm">

<div className="grid gap-2">

<div className="flex flex-col">

<span className="text-[0.70rem] uppercase text-muted-foreground">{label}</span>

{payload.map((item) => (

<span key={item.name} className="font-bold text-sm">

{item.name}: {item.value}

</span>

))}

</div>

</div>

</div>

)

}

"use client"

import \* as React from "react"

import \* as ProgressPrimitive from "@radix-ui/react-progress"

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

const Progress = React.forwardRef<

React.ElementRef<typeof ProgressPrimitive.Root>,

React.ComponentPropsWithoutRef<typeof ProgressPrimitive.Root>

>(({ className, value, ...props }, ref) => (

<ProgressPrimitive.Root

ref={ref}

className={cn("relative h-2 w-full overflow-hidden rounded-full bg-primary/20", className)}

{...props}

>

<ProgressPrimitive.Indicator

className="h-full w-full flex-1 bg-primary transition-all"

style={{ transform: `translateX(-${100 - (value || 0)}%)` }}

/>

</ProgressPrimitive.Root>

))

Progress.displayName = ProgressPrimitive.Root.displayName

export { Progress }

"use client"

import \* as React from "react"

import \* as ScrollAreaPrimitive from "@radix-ui/react-scroll-area"

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

const ScrollArea = React.forwardRef<

React.ElementRef<typeof ScrollAreaPrimitive.Root>,

React.ComponentPropsWithoutRef<typeof ScrollAreaPrimitive.Root>

>(({ className, children, ...props }, ref) => (

<ScrollAreaPrimitive.Root ref={ref} className={cn("relative overflow-hidden", className)} {...props}>

<ScrollAreaPrimitive.Viewport className="h-full w-full rounded-[inherit]">{children}</ScrollAreaPrimitive.Viewport>

<ScrollBar />

<ScrollAreaPrimitive.Corner />

</ScrollAreaPrimitive.Root>

))

ScrollArea.displayName = ScrollAreaPrimitive.Root.displayName

const ScrollBar = React.forwardRef<

React.ElementRef<typeof ScrollAreaPrimitive.ScrollAreaScrollbar>,

React.ComponentPropsWithoutRef<typeof ScrollAreaPrimitive.ScrollAreaScrollbar>

>(({ className, orientation = "vertical", ...props }, ref) => (

<ScrollAreaPrimitive.ScrollAreaScrollbar

ref={ref}

orientation={orientation}

className={cn(

"flex touch-none select-none transition-colors",

orientation === "vertical" && "h-full w-2.5 border-l border-l-transparent p-[1px]",

orientation === "horizontal" && "h-2.5 border-t border-t-transparent p-[1px]",

className,

)}

{...props}

>

<ScrollAreaPrimitive.ScrollAreaThumb className="relative flex-1 rounded-full bg-border" />

</ScrollAreaPrimitive.ScrollAreaScrollbar>

))

ScrollBar.displayName = ScrollAreaPrimitive.ScrollAreaScrollbar.displayName

export { ScrollArea, ScrollBar }

import \* as React from "react"

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

const Table = React.forwardRef<HTMLTableElement, React.HTMLAttributes<HTMLTableElement>>(

({ className, ...props }, ref) => (

<div className="relative w-full overflow-auto">

<table ref={ref} className={cn("w-full caption-bottom text-sm", className)} {...props} />

</div>

),

)

Table.displayName = "Table"

const TableHeader = React.forwardRef<HTMLTableSectionElement, React.HTMLAttributes<HTMLTableSectionElement>>(

({ className, ...props }, ref) => <thead ref={ref} className={cn("[&\_tr]:border-b", className)} {...props} />,

)

TableHeader.displayName = "TableHeader"

const TableBody = React.forwardRef<HTMLTableSectionElement, React.HTMLAttributes<HTMLTableSectionElement>>(

({ className, ...props }, ref) => (

<tbody ref={ref} className={cn("[&\_tr:last-child]:border-0", className)} {...props} />

),

)

TableBody.displayName = "TableBody"

const TableFooter = React.forwardRef<HTMLTableSectionElement, React.HTMLAttributes<HTMLTableSectionElement>>(

({ className, ...props }, ref) => (

<tfoot ref={ref} className={cn("border-t bg-muted/50 font-medium [&>tr]:last:border-b-0", className)} {...props} />

),

)

TableFooter.displayName = "TableFooter"

const TableRow = React.forwardRef<HTMLTableRowElement, React.HTMLAttributes<HTMLTableRowElement>>(

({ className, ...props }, ref) => (

<tr

ref={ref}

className={cn("border-b transition-colors hover:bg-muted/50 data-[state=selected]:bg-muted", className)}

{...props}

/>

),

)

TableRow.displayName = "TableRow"

const TableHead = React.forwardRef<HTMLTableCellElement, React.ThHTMLAttributes<HTMLTableCellElement>>(

({ className, ...props }, ref) => (

<th

ref={ref}

className={cn(

"h-12 px-4 text-left align-middle font-medium text-muted-foreground [&:has([role=checkbox])]:pr-0",

className,

)}

{...props}

/>

),

)

TableHead.displayName = "TableHead"

const TableCell = React.forwardRef<HTMLTableCellElement, React.TdHTMLAttributes<HTMLTableCellElement>>(

({ className, ...props }, ref) => (

<td ref={ref} className={cn("p-4 align-middle [&:has([role=checkbox])]:pr-0", className)} {...props} />

),

)

TableCell.displayName = "TableCell"

const TableCaption = React.forwardRef<HTMLTableCaptionElement, React.HTMLAttributes<HTMLTableCaptionElement>>(

({ className, ...props }, ref) => (

<caption ref={ref} className={cn("mt-4 text-sm text-muted-foreground", className)} {...props} />

),

)

TableCaption.displayName = "TableCaption"

export { Table, TableHeader, TableBody, TableFooter, TableHead, TableRow, TableCell, TableCaption }

"use client"

import { useChat } from "ai/react"

import { Send } from "lucide-react"

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

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

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

import { ScrollArea } from "@/components/ui/scroll-area"

export default function AIChat() {

const { messages, input, handleInputChange, handleSubmit } = useChat({

api: "/api/chat",

})

return (

<Card className="h-[600px] flex flex-col">

<CardHeader>

<CardTitle>Investment Assistant</CardTitle>

</CardHeader>

<CardContent className="flex-1 flex flex-col">

<ScrollArea className="flex-1 pr-4">

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

{messages.map((message) => (

<div

key={message.id}

className={`flex ${message.role === "assistant" ? "justify-start" : "justify-end"}`}

>

<div

className={`rounded-lg px-4 py-2 max-w-[80%] ${

message.role === "assistant" ? "bg-muted" : "bg-primary text-primary-foreground"

}`}

>

{message.content}

</div>

</div>

))}

</div>

</ScrollArea>

<form onSubmit={handleSubmit} className="flex items-center space-x-2 mt-4">

<Input placeholder="Ask about investment opportunities..." value={input} onChange={handleInputChange} />

<Button type="submit" size="icon">

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

<span className="sr-only">Send message</span>

</Button>

</form>

</CardContent>

</Card>

)

}

"use client"

import { useState } from "react"

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

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

import { ArrowRight } from "lucide-react"

import \* as z from "zod"

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

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

import { Form, FormControl, FormDescription, FormField, FormItem, FormLabel, FormMessage } from "@/components/ui/form"

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

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

import { Separator } from "@/components/ui/separator"

import { Slider } from "@/components/ui/slider"

import { Switch } from "@/components/ui/switch"

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

import type { BusinessRequirement } from "@/types"

const formSchema = z.object({

companyName: z.string().min(2, "Company name is required"),

industry: z.string().min(2, "Industry is required"),

investmentSize: z.number().min(100000, "Minimum investment is $100,000"),

employmentTarget: z.number().min(1, "Employment target is required"),

infrastructureNeeds: z.object({

power: z.boolean(),

water: z.boolean(),

internet: z.boolean(),

transportation: z.boolean(),

ports: z.boolean(),

}),

workforceNeeds: z.object({

skilled: z.number(),

unskilled: z.number(),

technical: z.number(),

}),

spaceRequirement: z.object({

type: z.enum(["land", "office", "industrial"]),

size: z.number(),

}),

timeline: z.string(),

environmentalFactors: z.array(z.string()),

additionalRequirements: z.array(z.string()),

contactPerson: z.object({

name: z.string(),

position: z.string(),

email: z.string().email(),

phone: z.string(),

}),

})

interface BusinessRequirementsFormProps {

onSubmit: (data: BusinessRequirement) => void

}

export default function BusinessRequirementsForm({ onSubmit }: BusinessRequirementsFormProps) {

const [step, setStep] = useState(1)

const form = useForm<z.infer<typeof formSchema>>({

resolver: zodResolver(formSchema),

defaultValues: {

infrastructureNeeds: {

power: false,

water: false,

internet: false,

transportation: false,

ports: false,

},

workforceNeeds: {

skilled: 0,

unskilled: 0,

technical: 0,

},

},

})

const nextStep = () => setStep((prev) => prev + 1)

const prevStep = () => setStep((prev) => prev - 1)

return (

<Form {...form}>

<form onSubmit={form.handleSubmit(onSubmit)}>

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

{step === 1 && (

<Card>

<CardHeader>

<CardTitle>Company Information</CardTitle>

<CardDescription>Tell us about your company and investment plans</CardDescription>

</CardHeader>

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

<FormField

control={form.control}

name="companyName"

render={({ field }) => (

<FormItem>

<FormLabel>Company Name</FormLabel>

<FormControl>

<Input placeholder="Enter company name" {...field} />

</FormControl>

<FormMessage />

</FormItem>

)}

/>

<FormField

control={form.control}

name="industry"

render={({ field }) => (

<FormItem>

<FormLabel>Industry</FormLabel>

<Select onValueChange={field.onChange} defaultValue={field.value}>

<FormControl>

<SelectTrigger>

<SelectValue placeholder="Select industry" />

</SelectTrigger>

</FormControl>

<SelectContent>

<SelectItem value="manufacturing">Manufacturing</SelectItem>

<SelectItem value="technology">Technology</SelectItem>

<SelectItem value="agriculture">Agriculture</SelectItem>

<SelectItem value="tourism">Tourism</SelectItem>

<SelectItem value="energy">Energy</SelectItem>

</SelectContent>

</Select>

<FormMessage />

</FormItem>

)}

/>

<FormField

control={form.control}

name="investmentSize"

render={({ field }) => (

<FormItem>

<FormLabel>Investment Size (USD)</FormLabel>

<FormControl>

<div className="flex items-center space-x-4">

<Slider

min={100000}

max={10000000}

step={100000}

value={[field.value]}

onValueChange={([value]) => field.onChange(value)}

/>

<span className="min-w-[100px] text-right">

{new Intl.NumberFormat("en-US", {

style: "currency",

currency: "USD",

maximumFractionDigits: 0,

}).format(field.value)}

</span>

</div>

</FormControl>

<FormMessage />

</FormItem>

)}

/>

</CardContent>

<CardFooter className="justify-end">

<Button onClick={nextStep}>

Next

<ArrowRight className="ml-2 h-4 w-4" />

</Button>

</CardFooter>

</Card>

)}

{step === 2 && (

<Card>

<CardHeader>

<CardTitle>Infrastructure & Workforce</CardTitle>

<CardDescription>Specify your infrastructure and workforce requirements</CardDescription>

</CardHeader>

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

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

<h3 className="font-medium">Infrastructure Needs</h3>

{Object.keys(form.getValues().infrastructureNeeds).map((need) => (

<FormField

key={need}

control={form.control}

name={`infrastructureNeeds.${need}`}

render={({ field }) => (

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

<FormLabel className="capitalize">{need}</FormLabel>

<FormControl>

<Switch checked={field.value} onCheckedChange={field.onChange} />

</FormControl>

</FormItem>

)}

/>

))}

</div>

<Separator />

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

<h3 className="font-medium">Workforce Requirements</h3>

{Object.entries(form.getValues().workforceNeeds).map(([type, value]) => (

<FormField

key={type}

control={form.control}

name={`workforceNeeds.${type}`}

render={({ field }) => (

<FormItem>

<FormLabel className="capitalize">{type}</FormLabel>

<FormControl>

<Input

type="number"

min={0}

{...field}

onChange={(e) => field.onChange(Number.parseInt(e.target.value))}

/>

</FormControl>

</FormItem>

)}

/>

))}

</div>

</CardContent>

<CardFooter className="justify-between">

<Button variant="outline" onClick={prevStep}>

Previous

</Button>

<Button onClick={nextStep}>

Next

<ArrowRight className="ml-2 h-4 w-4" />

</Button>

</CardFooter>

</Card>

)}

{step === 3 && (

<Card>

<CardHeader>

<CardTitle>Additional Information</CardTitle>

<CardDescription>Provide contact details and any additional requirements</CardDescription>

</CardHeader>

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

<div className="grid gap-4">

<FormField

control={form.control}

name="contactPerson.name"

render={({ field }) => (

<FormItem>

<FormLabel>Contact Name</FormLabel>

<FormControl>

<Input {...field} />

</FormControl>

<FormMessage />

</FormItem>

)}

/>

<FormField

control={form.control}

name="contactPerson.position"

render={({ field }) => (

<FormItem>

<FormLabel>Position</FormLabel>

<FormControl>

<Input {...field} />

</FormControl>

<FormMessage />

</FormItem>

)}

/>

<FormField

control={form.control}

name="contactPerson.email"

render={({ field }) => (

<FormItem>

<FormLabel>Email</FormLabel>

<FormControl>

<Input type="email" {...field} />

</FormControl>

<FormMessage />

</FormItem>

)}

/>

<FormField

control={form.control}

name="contactPerson.phone"

render={({ field }) => (

<FormItem>

<FormLabel>Phone</FormLabel>

<FormControl>

<Input {...field} />

</FormControl>

<FormMessage />

</FormItem>

)}

/>

</div>

<FormField

control={form.control}

name="additionalRequirements"

render={({ field }) => (

<FormItem>

<FormLabel>Additional Requirements</FormLabel>

<FormControl>

<Textarea

placeholder="Enter any additional requirements or preferences"

className="h-32"

onChange={(e) => field.onChange(e.target.value.split("\n"))}

/>

</FormControl>

<FormDescription>Enter each requirement on a new line</FormDescription>

<FormMessage />

</FormItem>

)}

/>

</CardContent>

<CardFooter className="justify-between">

<Button variant="outline" onClick={prevStep}>

Previous

</Button>

<Button type="submit">Submit Requirements</Button>

</CardFooter>

</Card>

)}

</div>

</form>

</Form>

)

}

"use client"

import { useEffect, useState } from "react" import { Bell, Building2, Mail, Phone, Search } from "lucide-react"

import { Badge } from "@/components/ui/badge" import { Button } from "@/components/ui/button" import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card" import { Progress } from "@/components/ui/progress" import { ScrollArea } from "@/components/ui/scroll-area" import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select" import { Separator } from "@/components/ui/separator" import { Slider } from "@/components/ui/slider" import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow } from "@/components/ui/table"

// Types interface Alert { id: string type: string message: string time: string }

interface Contact { id: string name: string position: string department: string office: string email: string phone: string }

interface Project { id: string title: string description: string department: string budget: number progress: number status: "planned" | "ongoing" | "completed" }

// Sample data const alerts: Alert[] = [ { id: "1", type: "Market Update", message: "PSEi up by 2.3% in morning trading", time: "2 mins ago", }, { id: "2", type: "Investment Alert", message: "New tax incentives announced for tech sector", time: "5 mins ago", }, { id: "3", type: "Regional Update", message: "Clark Freeport Zone opens new facilities", time: "10 mins ago", }, ]

const contacts: Contact[] = [ { id: "1", name: "Maria Santos", position: "Regional Director", department: "DTI", office: "NCR Regional Office", email: "[maria.santos@dti.gov.ph](mailto:maria.santos@dti.gov.ph)", phone: "+63 2 8751 0384", }, { id: "2", name: "Juan Dela Cruz", position: "Investment Specialist", department: "BOI", office: "Central Office", email: "[juan.delacruz@boi.gov.ph](mailto:juan.delacruz@boi.gov.ph)", phone: "+63 2 8575 3500", }, ]

const projects: Project[] = [ { id: "1", title: "Clark Green City Development", description: "Sustainable urban development project in Clark, Pampanga", department: "BCDA", budget: 50000000000, progress: 45, status: "ongoing", }, { id: "2", title: "Mindanao Railway Project", description: "Railway system connecting key cities in Mindanao", department: "DOTr", budget: 82000000000, progress: 25, status: "ongoing", }, ]

export default function Dashboard() { const [mounted, setMounted] = useState(false) const [selectedDepartment, setSelectedDepartment] = useState("") const [investmentSize, setInvestmentSize] = useState(1000000)

useEffect(() => { setMounted(true) }, [])

if (!mounted) { return null }

return (

# **Philippines Investment Portal**

<div className="grid gap-6 md:grid-cols-2">  
 {/\* Real-time Alerts \*/}  
 <Card>  
 <CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">  
 <CardTitle className="text-base font-medium">Real-Time Alerts</CardTitle>  
 <Bell className="h-4 w-4 text-muted-foreground" />  
 </CardHeader>  
 <CardContent>  
 <Table>  
 <TableHeader>  
 <TableRow>  
 <TableHead>Type</TableHead>  
 <TableHead>Alert</TableHead>  
 <TableHead>Time</TableHead>  
 </TableRow>  
 </TableHeader>  
 <TableBody>  
 {alerts.map((alert) => (  
 <TableRow key={alert.id}>  
 <TableCell className="font-medium">{alert.type}</TableCell>  
 <TableCell>{alert.message}</TableCell>  
 <TableCell>{alert.time}</TableCell>  
 </TableRow>  
 ))}  
 </TableBody>  
 </Table>  
 </CardContent>  
 </Card>  
  
 {/\* Investment Matcher \*/}  
 <Card>  
 <CardHeader>  
 <CardTitle className="text-base font-medium">Investment Matching</CardTitle>  
 </CardHeader>  
 <CardContent>  
 <div className="space-y-4">  
 <div className="space-y-2">  
 <label className="text-sm font-medium">Industry Sector</label>  
 <Select onValueChange={(value) => console.log(value)}>  
 <SelectTrigger>  
 <SelectValue placeholder="Select sector" />  
 </SelectTrigger>  
 <SelectContent>  
 <SelectItem value="manufacturing">Manufacturing</SelectItem>  
 <SelectItem value="technology">Technology</SelectItem>  
 <SelectItem value="agriculture">Agriculture</SelectItem>  
 <SelectItem value="tourism">Tourism</SelectItem>  
 </SelectContent>  
 </Select>  
 </div>  
  
 <div className="space-y-2">  
 <label className="text-sm font-medium">Investment Size (USD)</label>  
 <div className="flex items-center space-x-4">  
 <Slider  
 min={100000}  
 max={10000000}  
 step={100000}  
 value={[investmentSize]}  
 onValueChange={([value]) => setInvestmentSize(value)}  
 />  
 <span className="min-w-[100px] text-right">  
 {new Intl.NumberFormat("en-US", {  
 style: "currency",  
 currency: "USD",  
 maximumFractionDigits: 0,  
 }).format(investmentSize)}  
 </span>  
 </div>  
 </div>  
  
 <Button className="w-full" onClick={() => console.log("Matching...")}>  
 <Search className="mr-2 h-4 w-4" />  
 Find Opportunities  
 </Button>  
 </div>  
 </CardContent>  
 </Card>  
  
 {/\* Project Tracker \*/}  
 <Card>  
 <CardHeader>  
 <CardTitle className="text-base font-medium">Government Projects</CardTitle>  
 </CardHeader>  
 <CardContent>  
 <ScrollArea className="h-[300px] pr-4">  
 <div className="space-y-4">  
 {projects.map((project) => (  
 <div key={project.id} className="rounded-lg border p-4 hover:bg-accent">  
 <div className="flex items-start justify-between">  
 <div>  
 <h3 className="font-semibold">{project.title}</h3>  
 <p className="text-sm text-muted-foreground">{project.department}</p>  
 </div>  
 <Badge variant="outline">{project.status}</Badge>  
 </div>  
 <div className="mt-4 space-y-2">  
 <div className="flex items-center justify-between text-sm">  
 <span>Budget</span>  
 <span>  
 {new Intl.NumberFormat("en-US", {  
 style: "currency",  
 currency: "PHP",  
 maximumFractionDigits: 0,  
 }).format(project.budget)}  
 </span>  
 </div>  
 <Progress value={project.progress} className="h-2" />  
 <p className="text-xs text-muted-foreground">Progress: {project.progress}%</p>  
 </div>  
 </div>  
 ))}  
 </div>  
 </ScrollArea>  
 </CardContent>  
 </Card>  
  
 {/\* Government Contacts \*/}  
 <Card>  
 <CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">  
 <CardTitle className="text-base font-medium">Government Contacts</CardTitle>  
 <Select value={selectedDepartment} onValueChange={setSelectedDepartment}>  
 <SelectTrigger className="w-[180px]">  
 <SelectValue placeholder="All Departments" />  
 </SelectTrigger>  
 <SelectContent>  
 <SelectItem value="All">All Departments</SelectItem>  
 <SelectItem value="DTI">DTI</SelectItem>  
 <SelectItem value="BOI">BOI</SelectItem>  
 <SelectItem value="PEZA">PEZA</SelectItem>  
 </SelectContent>  
 </Select>  
 </CardHeader>  
 <CardContent>  
 <ScrollArea className="h-[300px] pr-4">  
 <div className="space-y-4">  
 {contacts  
 .filter((contact) => !selectedDepartment || contact.department === selectedDepartment)  
 .map((contact) => (  
 <div key={contact.id} className="rounded-lg border p-4 hover:bg-accent">  
 <div className="flex items-start justify-between">  
 <div>  
 <h3 className="font-semibold">{contact.name}</h3>  
 <p className="text-sm text-muted-foreground">{contact.position}</p>  
 </div>  
 <Badge>{contact.department}</Badge>  
 </div>  
 <Separator className="my-2" />  
 <div className="grid gap-2">  
 <div className="flex items-center text-sm">  
 <Building2 className="mr-2 h-4 w-4" />  
 {contact.office}  
 </div>  
 <div className="flex items-center text-sm">  
 <Mail className="mr-2 h-4 w-4" />  
 {contact.email}  
 </div>  
 <div className="flex items-center text-sm">  
 <Phone className="mr-2 h-4 w-4" />  
 {contact.phone}  
 </div>  
 </div>  
 </div>  
 ))}  
 </div>  
 </ScrollArea>  
 </CardContent>  
 </Card>  
 </div>  
</div>

) }

import { AlertCircle } from "lucide-react"

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

interface ErrorStateProps {

title: string

description: string

}

export default function ErrorState({ title, description }: ErrorStateProps) {

return (

<Alert variant="destructive">

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

<AlertTitle>{title}</AlertTitle>

<AlertDescription>{description}</AlertDescription>

</Alert>

)

}

"use client"

import { Building2, Mail, Phone } from "lucide-react"

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

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

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

import { Separator } from "@/components/ui/separator"

import type { GovernmentContact } from "@/types"

interface GovernmentContactsProps {

contacts: GovernmentContact[]

}

export default function GovernmentContacts({ contacts }: GovernmentContactsProps) {

return (

<Card>

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

<CardTitle>Government Contacts Directory</CardTitle>

<div className="flex space-x-4">

<Select>

<SelectTrigger className="w-[200px]">

<SelectValue placeholder="Select Department" />

</SelectTrigger>

<SelectContent>

<SelectItem value="dti">Department of Trade and Industry</SelectItem>

<SelectItem value="doi">Department of Interior</SelectItem>

<SelectItem value="da">Department of Agriculture</SelectItem>

</SelectContent>

</Select>

<Select>

<SelectTrigger className="w-[200px]">

<SelectValue placeholder="Select Region" />

</SelectTrigger>

<SelectContent>

<SelectItem value="ncr">National Capital Region</SelectItem>

<SelectItem value="r1">Region I</SelectItem>

<SelectItem value="r2">Region II</SelectItem>

</SelectContent>

</Select>

</div>

</CardHeader>

<CardContent className="grid gap-4">

{contacts.map((contact) => (

<div key={contact.id} className="rounded-lg border p-4 hover:bg-accent">

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

<div>

<h3 className="font-semibold">{contact.name}</h3>

<p className="text-sm text-muted-foreground">{contact.position}</p>

</div>

<Badge>{contact.department}</Badge>

</div>

<Separator className="my-2" />

<div className="grid gap-2">

<div className="flex items-center text-sm">

<Building2 className="mr-2 h-4 w-4" />

{contact.office}

</div>

<div className="flex items-center text-sm">

<Mail className="mr-2 h-4 w-4" />

{contact.email}

</div>

<div className="flex items-center text-sm">

<Phone className="mr-2 h-4 w-4" />

{contact.phone}

</div>

</div>

</div>

))}

</CardContent>

</Card>

)

}

"use client"

import { AlertTriangle } from "lucide-react"

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

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

import { ScrollArea } from "@/components/ui/scroll-area"

import type { InvestmentAlert } from "@/types"

interface InvestmentAlertsProps {

alerts: InvestmentAlert[]

}

export default function InvestmentAlerts({ alerts }: InvestmentAlertsProps) {

// Fallback data if no alerts are provided

const defaultAlerts: InvestmentAlert[] = [

{

id: "1",

title: "New Investment Opportunity",

description: "Tech sector showing strong growth potential in NCR",

severity: "medium",

timestamp: new Date().toISOString(),

},

]

const displayAlerts = alerts.length > 0 ? alerts : defaultAlerts

return (

<Card>

<CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">

<CardTitle className="text-base font-medium">Investment Alerts</CardTitle>

<AlertTriangle className="h-4 w-4 text-muted-foreground" />

</CardHeader>

<CardContent>

<ScrollArea className="h-[300px] pr-4">

{displayAlerts.map((alert) => (

<Alert

key={alert.id}

variant={alert.severity === "high" ? "destructive" : alert.severity === "medium" ? "default" : "outline"}

className="mb-3"

>

<AlertTitle>{alert.title}</AlertTitle>

<AlertDescription>{alert.description}</AlertDescription>

</Alert>

))}

</ScrollArea>

</CardContent>

</Card>

)

}

"use client"

import { useEffect, useState } from "react"

import dynamic from "next/dynamic"

import { Card } from "@/components/ui/card"

import { Tooltip, TooltipContent, TooltipProvider, TooltipTrigger } from "@/components/ui/tooltip"

import { fetchInvestmentAlerts, fetchMarketSentiment, fetchRegionalData, fetchRiskAnalysis } from "@/lib/api"

import type { InvestmentAlert, RegionalData, RiskAnalysis } from "@/types"

// Dynamically import components

const RealTimeAlerts = dynamic(() => import("@/components/real-time-alerts"), {

ssr: false,

loading: () => <Card className="h-[300px] animate-pulse bg-muted" />,

})

const InvestmentAlerts = dynamic(() => import("@/components/investment-alerts"), {

ssr: false,

loading: () => <Card className="h-[300px] animate-pulse bg-muted" />,

})

const RiskAnalysisCard = dynamic(() => import("@/components/risk-analysis-card"), {

ssr: false,

loading: () => <Card className="h-[300px] animate-pulse bg-muted" />,

})

const MarketSentimentChart = dynamic(() => import("@/components/market-sentiment-chart"), {

ssr: false,

loading: () => <Card className="h-[300px] animate-pulse bg-muted" />,

})

const RegionalMap = dynamic(() => import("@/components/regional-map"), {

ssr: false,

loading: () => <Card className="h-[300px] animate-pulse bg-muted" />,

})

const PhilippinesMap = dynamic(() => import("@/components/philippines-map"), {

ssr: false,

loading: () => <Card className="h-[300px] animate-pulse bg-muted" />,

})

const LoadingDashboard = dynamic(() => import("@/components/loading-states"), {

ssr: false,

})

const ErrorState = dynamic(() => import("@/components/empty-states"), {

ssr: false,

})

export default function RealTimeDashboard() {

const [regionalData, setRegionalData] = useState<RegionalData | null>(null)

const [investmentAlerts, setInvestmentAlerts] = useState<InvestmentAlert[]>([])

const [riskAnalysis, setRiskAnalysis] = useState<RiskAnalysis | null>(null)

const [marketSentiment, setMarketSentiment] = useState<number[]>([])

const [loading, setLoading] = useState(true)

const [error, setError] = useState<string | null>(null)

useEffect(() => {

const fetchData = async () => {

try {

const [regional, alerts, risk, sentiment] = await Promise.all([

fetchRegionalData("NCR"),

fetchInvestmentAlerts(),

fetchRiskAnalysis("NCR"),

fetchMarketSentiment(),

])

setRegionalData(regional)

setInvestmentAlerts(alerts)

setRiskAnalysis(risk)

setMarketSentiment(sentiment)

} catch (err) {

setError("Failed to load real-time data. Please try again later.")

} finally {

setLoading(false)

}

}

fetchData()

}, [])

if (loading) {

return <LoadingDashboard />

}

if (error) {

return <ErrorState title="Error loading dashboard" description={error} />

}

return (

<TooltipProvider>

<div className="grid gap-6 p-6 md:grid-cols-2 lg:grid-cols-3">

<RealTimeAlerts />

<InvestmentAlerts alerts={investmentAlerts} />

<RiskAnalysisCard

data={

riskAnalysis || {

political: 0,

crime: 0,

economy: 0,

details: { political: [], crime: [], economy: [] },

lastUpdated: "",

}

}

/>

<MarketSentimentChart data={marketSentiment} />

<Tooltip>

<TooltipTrigger asChild>

<div>

<RegionalMap data={regionalData?.regions || []} onRegionSelect={() => {}} />

</div>

</TooltipTrigger>

<TooltipContent>Click on a region to view details</TooltipContent>

</Tooltip>

<Tooltip>

<TooltipTrigger asChild>

<div>

<PhilippinesMap

provinces={regionalData?.provinces || []}

opportunities={regionalData?.opportunities || []}

onProvinceSelect={() => {}}

onOpportunitySelect={() => {}}

/>

</div>

</TooltipTrigger>

<TooltipContent>Click on a province to view investment opportunities</TooltipContent>

</Tooltip>

</div>

</TooltipProvider>

)

}

"use client"

import { Gift } from "lucide-react"

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

import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow } from "@/components/ui/table"

import { Tooltip, TooltipContent, TooltipProvider, TooltipTrigger } from "@/components/ui/tooltip"

import type { InvestmentIncentive } from "@/types"

interface InvestmentIncentivesProps {

incentives: InvestmentIncentive[]

}

export default function InvestmentIncentives({ incentives }: InvestmentIncentivesProps) {

return (

<Card>

<CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">

<CardTitle className="text-base font-medium">Government Incentives</CardTitle>

<Gift className="h-4 w-4 text-muted-foreground" />

</CardHeader>

<CardContent>

<TooltipProvider>

<Table>

<TableHeader>

<TableRow>

<TableHead>Type</TableHead>

<TableHead>Title</TableHead>

<TableHead>Region</TableHead>

<TableHead>Expiry</TableHead>

</TableRow>

</TableHeader>

<TableBody>

{incentives.map((incentive) => (

<TableRow key={incentive.id}>

<TableCell>{incentive.type}</TableCell>

<TableCell>

<Tooltip>

<TooltipTrigger className="text-left">{incentive.title}</TooltipTrigger>

<TooltipContent>

<div className="max-w-xs">

<p className="font-medium">Benefits:</p>

<ul className="list-disc pl-4 text-sm">

{incentive.benefits.map((benefit, index) => (

<li key={index}>{benefit}</li>

))}

</ul>

</div>

</TooltipContent>

</Tooltip>

</TableCell>

<TableCell>{incentive.region}</TableCell>

<TableCell>{new Date(incentive.expiryDate).toLocaleDateString()}</TableCell>

</TableRow>

))}

</TableBody>

</Table>

</TooltipProvider>

</CardContent>

</Card>

)

}

"use client"

import type React from "react"

import { useEffect, useState } from "react"

import { Search } from "lucide-react"

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

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

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

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

import { Separator } from "@/components/ui/separator"

import { Slider } from "@/components/ui/slider"

import type { CompanyRequirement, RegionData } from "@/types"

interface InvestmentMatcherProps {

regions: RegionData[]

onMatch: (requirements: CompanyRequirement) => void

}

const initialRequirements: CompanyRequirement = {

industry: "",

investmentSize: 100000,

employmentTarget: 0,

infrastructureNeeds: [],

resourceRequirements: [],

timeline: "",

}

export default function InvestmentMatcher({ regions, onMatch }: InvestmentMatcherProps) {

const [mounted, setMounted] = useState(false)

const [requirements, setRequirements] = useState<CompanyRequirement>(initialRequirements)

useEffect(() => {

setMounted(true)

}, [])

const handleSubmit = (e: React.FormEvent) => {

e.preventDefault()

onMatch(requirements)

}

if (!mounted) {

return null // Prevent hydration mismatch

}

return (

<Card>

<CardHeader>

<CardTitle>Investment Matching System</CardTitle>

<CardDescription>Enter your requirements to find the perfect investment location</CardDescription>

</CardHeader>

<CardContent>

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

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

<label className="text-sm font-medium">Industry</label>

<Select

value={requirements.industry}

onValueChange={(value) => setRequirements({ ...requirements, industry: value })}

>

<SelectTrigger>

<SelectValue placeholder="Select Industry" />

</SelectTrigger>

<SelectContent>

{["Manufacturing", "Technology", "Agriculture", "Tourism", "Energy"].map((industry) => (

<SelectItem key={industry} value={industry.toLowerCase()}>

{industry}

</SelectItem>

))}

</SelectContent>

</Select>

</div>

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

<label className="text-sm font-medium">Investment Size (USD)</label>

<div className="flex items-center space-x-4">

<Slider

min={100000}

max={10000000}

step={100000}

value={[requirements.investmentSize]}

onValueChange={([value]) => setRequirements({ ...requirements, investmentSize: value })}

/>

<span className="min-w-[100px] text-right">

{new Intl.NumberFormat("en-US", {

style: "currency",

currency: "USD",

maximumFractionDigits: 0,

}).format(requirements.investmentSize)}

</span>

</div>

</div>

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

<label className="text-sm font-medium">Employment Target</label>

<Input

type="number"

min={0}

value={requirements.employmentTarget}

onChange={(e) =>

setRequirements({

...requirements,

employmentTarget: Number.parseInt(e.target.value) || 0,

})

}

/>

</div>

<Separator />

<div className="flex justify-end">

<Button type="submit">

<Search className="mr-2 h-4 w-4" />

Find Matches

</Button>

</div>

</form>

</CardContent>

</Card>

)

}

"use client"

import { Briefcase } from "lucide-react"

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

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

import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow } from "@/components/ui/table"

import type { InvestmentOpportunity } from "@/types"

interface InvestmentOpportunitiesProps {

opportunities: InvestmentOpportunity[]

onSelect: (opportunity: InvestmentOpportunity) => void

}

export default function InvestmentOpportunities({ opportunities, onSelect }: InvestmentOpportunitiesProps) {

return (

<Card>

<CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">

<CardTitle className="text-base font-medium">Investment Opportunities</CardTitle>

<Briefcase className="h-4 w-4 text-muted-foreground" />

</CardHeader>

<CardContent>

<Table>

<TableHeader>

<TableRow>

<TableHead>Sector</TableHead>

<TableHead>Region</TableHead>

<TableHead>Investment (USD)</TableHead>

<TableHead>Jobs</TableHead>

<TableHead>Status</TableHead>

</TableRow>

</TableHeader>

<TableBody>

{opportunities.map((opportunity) => (

<TableRow

key={opportunity.id}

className="cursor-pointer hover:bg-accent"

onClick={() => onSelect(opportunity)}

>

<TableCell className="font-medium">{opportunity.sector}</TableCell>

<TableCell>{opportunity.region}</TableCell>

<TableCell>

{opportunity.investmentSize.toLocaleString("en-US", {

style: "currency",

currency: "USD",

minimumFractionDigits: 0,

maximumFractionDigits: 0,

})}

</TableCell>

<TableCell>{opportunity.jobsCreated.toLocaleString()}</TableCell>

<TableCell>

<Badge

variant={

opportunity.status === "open"

? "default"

: opportunity.status === "pending"

? "secondary"

: "outline"

}

>

{opportunity.status}

</Badge>

</TableCell>

</TableRow>

))}

</TableBody>

</Table>

</CardContent>

</Card>

)

}

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

import { Skeleton } from "@/components/ui/skeleton"

export default function LoadingDashboard() {

return (

<div className="grid gap-6 md:grid-cols-2 lg:grid-cols-3">

{Array.from({ length: 6 }).map((\_, i) => (

<Card key={i} className="overflow-hidden">

<CardHeader className="space-y-2 p-4">

<Skeleton className="h-4 w-1/2" />

<Skeleton className="h-4 w-3/4" />

</CardHeader>

<CardContent className="p-4">

<Skeleton className="h-[200px]" />

</CardContent>

</Card>

))}

</div>

)

}

"use client"

import { TrendingUp } from "lucide-react"

import { Line, LineChart, ResponsiveContainer, Tooltip, XAxis, YAxis } from "recharts"

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

import { ChartTooltipContent } from "@/components/ui/chart"

interface MarketSentimentChartProps {

data: number[]

}

export default function MarketSentimentChart({ data }: MarketSentimentChartProps) {

// Transform the data array into the format required by recharts

const chartData = data.map((value, index) => ({

timestamp: new Date(Date.now() - (data.length - 1 - index) \* 3600000).toISOString(),

sentiment: value,

}))

return (

<Card>

<CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">

<CardTitle className="text-base font-medium">Market Sentiment</CardTitle>

<TrendingUp className="h-4 w-4 text-muted-foreground" />

</CardHeader>

<CardContent>

<div className="h-[300px]">

<ResponsiveContainer width="100%" height="100%">

<LineChart data={chartData}>

<XAxis

dataKey="timestamp"

tickFormatter={(value) => {

return new Date(value).toLocaleTimeString([], {

hour: "2-digit",

minute: "2-digit",

})

}}

/>

<YAxis />

<Line type="monotone" dataKey="sentiment" stroke="hsl(var(--primary))" strokeWidth={2} dot={false} />

<Tooltip content={<ChartTooltipContent />} />

</LineChart>

</ResponsiveContainer>

</div>

</CardContent>

</Card>

)

}

"use client"

import { TrendingUp } from "lucide-react"

import { Area, AreaChart, ResponsiveContainer, Tooltip, XAxis, YAxis } from "recharts"

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

import { ChartTooltipContent } from "@/components/ui/chart"

import type { MarketTrend } from "@/types"

interface MarketTrendsProps {

data: MarketTrend[]

}

export default function MarketTrends({ data }: MarketTrendsProps) {

return (

<Card>

<CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">

<CardTitle className="text-base font-medium">Market Trends</CardTitle>

<TrendingUp className="h-4 w-4 text-muted-foreground" />

</CardHeader>

<CardContent>

<div className="h-[300px]">

<ResponsiveContainer width="100%" height="100%">

<AreaChart data={data}>

<XAxis

dataKey="date"

tickFormatter={(value) => {

return new Date(value).toLocaleDateString(undefined, {

month: "short",

year: "2-digit",

})

}}

/>

<YAxis />

<Tooltip content={<ChartTooltipContent />} />

<Area

type="monotone"

dataKey="fdi"

name="FDI (USD Millions)"

stroke="hsl(var(--primary))"

fill="hsl(var(--primary))"

fillOpacity={0.2}

/>

<Area

type="monotone"

dataKey="gdpGrowth"

name="GDP Growth (%)"

stroke="hsl(var(--secondary))"

fill="hsl(var(--secondary))"

fillOpacity={0.2}

/>

<Area

type="monotone"

dataKey="employmentRate"

name="Employment Rate (%)"

stroke="hsl(var(--accent))"

fill="hsl(var(--accent))"

fillOpacity={0.2}

/>

</AreaChart>

</ResponsiveContainer>

</div>

</CardContent>

</Card>

)

}

"use client"

import { Award, Mail, MapPin } from "lucide-react"

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

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

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

import { Progress } from "@/components/ui/progress"

import { ScrollArea } from "@/components/ui/scroll-area"

import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow } from "@/components/ui/table"

import { Tooltip, TooltipContent, TooltipProvider, TooltipTrigger } from "@/components/ui/tooltip"

import type { MatchResult, RegionalProfile } from "@/types"

interface MatchResultsProps {

matches: MatchResult[]

regions: RegionalProfile[]

onContactRegion: (match: MatchResult) => void

}

export default function MatchResults({ matches, regions, onContactRegion }: MatchResultsProps) {

const getRegionName = (regionId: string) => {

const region = regions.find((r) => r.id === regionId)

return region ? region.name : "Unknown Region"

}

const getScoreColor = (score: number) => {

if (score >= 80) return "text-green-500"

if (score >= 60) return "text-yellow-500"

return "text-red-500"

}

return (

<Card>

<CardHeader>

<CardTitle className="flex items-center gap-2">

<Award className="h-5 w-5" />

Top Regional Matches

</CardTitle>

</CardHeader>

<CardContent>

<ScrollArea className="h-[600px] pr-4">

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

{matches.map((match, index) => (

<Card key={match.id}>

<CardHeader className="pb-2">

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

<div>

<h3 className="font-semibold">{getRegionName(match.regionId)}</h3>

<p className="text-sm text-muted-foreground">

Match Score:{" "}

<span className={getScoreColor(match.matchScore)}>{match.matchScore.toFixed(1)}%</span>

</p>

</div>

<Badge variant={index < 3 ? "default" : "secondary"}>Rank #{index + 1}</Badge>

</div>

</CardHeader>

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

<Table>

<TableHeader>

<TableRow>

<TableHead>Factor</TableHead>

<TableHead>Score</TableHead>

<TableHead className="w-[100px]">Rating</TableHead>

</TableRow>

</TableHeader>

<TableBody>

{match.matchFactors.map((factor) => (

<TableRow key={factor.factor}>

<TableCell className="font-medium">

<TooltipProvider>

<Tooltip>

<TooltipTrigger className="cursor-help">{factor.factor}</TooltipTrigger>

<TooltipContent>

<p className="max-w-xs">{factor.details}</p>

</TooltipContent>

</Tooltip>

</TooltipProvider>

</TableCell>

<TableCell>{factor.score.toFixed(1)}%</TableCell>

<TableCell>

<Progress

value={factor.score}

className={

factor.score >= 80

? "bg-green-500"

: factor.score >= 60

? "bg-yellow-500"

: "bg-red-500"

}

/>

</TableCell>

</TableRow>

))}

</TableBody>

</Table>

<div className="flex justify-between">

<Button

variant="outline"

size="sm"

className="gap-2"

onClick={() => {

// Open region details

}}

>

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

View Details

</Button>

<Button size="sm" className="gap-2" onClick={() => onContactRegion(match)}>

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

Contact Region

</Button>

</div>

</CardContent>

</Card>

))}

</div>

</ScrollArea>

</CardContent>

</Card>

)

}

"use client"

import { useEffect, useState } from "react"

import { Bot } from "lucide-react"

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

import { Progress } from "@/components/ui/progress"

import type { BusinessRequirement, MatchResult, RegionalProfile } from "@/types"

interface MatchingEngineProps {

requirement: BusinessRequirement

regions: RegionalProfile[]

onMatchComplete: (matches: MatchResult[]) => void

}

export default function MatchingEngine({ requirement, regions, onMatchComplete }: MatchingEngineProps) {

const [progress, setProgress] = useState(0)

const [status, setStatus] = useState("Initializing matching engine...")

useEffect(() => {

const matchRegions = async () => {

setStatus("Analyzing business requirements...")

setProgress(20)

await new Promise((resolve) => setTimeout(resolve, 1000))

setStatus("Evaluating regional profiles...")

setProgress(40)

await new Promise((resolve) => setTimeout(resolve, 1000))

setStatus("Calculating match scores...")

setProgress(60)

await new Promise((resolve) => setTimeout(resolve, 1000))

setStatus("Ranking potential matches...")

setProgress(80)

await new Promise((resolve) => setTimeout(resolve, 1000))

setStatus("Finalizing results...")

setProgress(100)

// Calculate matches

const matches = regions

.map((region) => {

const infrastructureScore = calculateInfrastructureScore(requirement, region)

const workforceScore = calculateWorkforceScore(requirement, region)

const locationScore = calculateLocationScore(requirement, region)

const incentivesScore = calculateIncentivesScore(region)

const totalScore = (infrastructureScore + workforceScore + locationScore + incentivesScore) / 4

return {

id: `match-${region.id}`,

businessId: requirement.id,

regionId: region.id,

matchScore: totalScore,

matchFactors: [

{

factor: "Infrastructure",

score: infrastructureScore,

details: "Based on available facilities and utilities",

},

{

factor: "Workforce",

score: workforceScore,

details: "Based on available skilled and unskilled labor",

},

{

factor: "Location",

score: locationScore,

details: "Based on accessibility and market proximity",

},

{

factor: "Incentives",

score: incentivesScore,

details: "Based on available government incentives",

},

],

status: "pending",

timeline: {

created: new Date().toISOString(),

},

notes: [],

} as MatchResult

})

.sort((a, b) => b.matchScore - a.matchScore)

.slice(0, 5)

onMatchComplete(matches)

}

matchRegions()

}, [requirement, regions, onMatchComplete])

return (

<Card>

<CardHeader>

<CardTitle className="flex items-center gap-2">

<Bot className="h-5 w-5" />

AI Matching Engine

</CardTitle>

</CardHeader>

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

<Progress value={progress} className="h-2" />

<p className="text-sm text-muted-foreground">{status}</p>

</CardContent>

</Card>

)

}

// Scoring functions

function calculateInfrastructureScore(requirement: BusinessRequirement, region: RegionalProfile): number {

let score = 0

const { infrastructureNeeds } = requirement

if (infrastructureNeeds.power && region.infrastructure.power.available) {

score += 25

}

if (infrastructureNeeds.water && region.infrastructure.water.available) {

score += 25

}

if (infrastructureNeeds.internet && region.infrastructure.internet.available) {

score += 25

}

if (

infrastructureNeeds.transportation &&

(region.infrastructure.transportation.airports > 0 || region.infrastructure.transportation.highways > 0)

) {

score += 25

}

return score

}

function calculateWorkforceScore(requirement: BusinessRequirement, region: RegionalProfile): number {

const requiredTotal =

requirement.workforceNeeds.skilled + requirement.workforceNeeds.unskilled + requirement.workforceNeeds.technical

const availableTotal = region.workforce.skilled + region.workforce.unskilled + region.workforce.technical

return Math.min((availableTotal / requiredTotal) \* 100, 100)

}

function calculateLocationScore(requirement: BusinessRequirement, region: RegionalProfile): number {

// Basic location score based on space availability

let score = 0

const { spaceRequirement } = requirement

if (spaceRequirement.type === "land") {

score = (region.landAvailability.industrial / spaceRequirement.size) \* 100

} else if (spaceRequirement.type === "office") {

score = (region.landAvailability.commercial / spaceRequirement.size) \* 100

}

return Math.min(score, 100)

}

function calculateIncentivesScore(region: RegionalProfile): number {

// Score based on number of incentives

return Math.min((region.incentives.tax.length + region.incentives.other.length) \* 10, 100)

}

"use client"

import { Map } from "lucide-react"

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

import type { Opportunity, Province } from "@/types"

interface PhilippinesMapProps {

provinces: Province[]

opportunities: Opportunity[]

onProvinceSelect: (province: Province) => void

onOpportunitySelect: (opportunity: Opportunity) => void

}

export default function PhilippinesMap({

provinces,

opportunities,

onProvinceSelect,

onOpportunitySelect,

}: PhilippinesMapProps) {

return (

<Card>

<CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">

<CardTitle className="text-base font-medium">Philippines Investment Map</CardTitle>

<Map className="h-4 w-4 text-muted-foreground" />

</CardHeader>

<CardContent>

<div className="flex h-[300px] items-center justify-center">

<p className="text-sm text-muted-foreground">Interactive map implementation required</p>

</div>

</CardContent>

</Card>

)

}

"use client"

import { useEffect, useState } from "react"

import { Clock, Target } from "lucide-react"

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

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

import { Progress } from "@/components/ui/progress"

import { Tooltip, TooltipContent, TooltipProvider, TooltipTrigger } from "@/components/ui/tooltip"

import type { GovernmentProject } from "@/types"

interface ProjectTrackerProps {

projects: GovernmentProject[]

}

export default function ProjectTracker({ projects }: ProjectTrackerProps) {

const [mounted, setMounted] = useState(false)

useEffect(() => {

setMounted(true)

}, [])

const getStatusColor = (status: GovernmentProject["status"]) => {

switch (status) {

case "planned":

return "bg-yellow-500"

case "ongoing":

return "bg-green-500"

case "completed":

return "bg-blue-500"

default:

return "bg-gray-500"

}

}

const calculateProgress = (project: GovernmentProject) => {

if (!mounted) return 0 // Return 0 during SSR

const start = new Date(project.startDate).getTime()

const end = new Date(project.endDate).getTime()

const now = new Date().getTime()

const progress = ((now - start) / (end - start)) \* 100

return Math.min(Math.max(progress, 0), 100)

}

if (!mounted) {

return null // Prevent hydration mismatch

}

return (

<Card>

<CardHeader>

<CardTitle>Government Projects</CardTitle>

</CardHeader>

<CardContent className="grid gap-4">

{projects.map((project) => (

<TooltipProvider key={project.id}>

<Tooltip>

<TooltipTrigger asChild>

<div className="rounded-lg border p-4 hover:bg-accent">

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

<div>

<h3 className="font-semibold">{project.title}</h3>

<p className="text-sm text-muted-foreground">

{project.department} - {project.region}

</p>

</div>

<Badge variant="outline">{project.status}</Badge>

</div>

<div className="mt-4 space-y-2">

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

<div className="flex items-center">

<Clock className="mr-2 h-4 w-4" />

Timeline

</div>

<span>

{new Date(project.startDate).toLocaleDateString()} -{" "}

{new Date(project.endDate).toLocaleDateString()}

</span>

</div>

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

<div className="flex items-center">

<Target className="mr-2 h-4 w-4" />

Budget

</div>

<span>

{new Intl.NumberFormat("en-US", {

style: "currency",

currency: "PHP",

}).format(project.budget)}

</span>

</div>

<Progress value={calculateProgress(project)} className={getStatusColor(project.status)} />

</div>

</div>

</TooltipTrigger>

<TooltipContent>

<p className="max-w-xs">{project.description}</p>

</TooltipContent>

</Tooltip>

</TooltipProvider>

))}

</CardContent>

</Card>

)

}

"use client"

import { Bell } from "lucide-react"

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

import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow } from "@/components/ui/table"

interface Alert {

id: string

type: string

message: string

time: string

}

const alerts: Alert[] = [

{

id: "1",

type: "Market Update",

message: "PSEi up by 2.3% in morning trading",

time: "2 mins ago",

},

{

id: "2",

type: "Currency Alert",

message: "PHP strengthens against USD",

time: "5 mins ago",

},

{

id: "3",

type: "Trading Alert",

message: "Unusual volume detected in banking sector",

time: "10 mins ago",

},

]

export default function RealTimeAlerts() {

return (

<Card>

<CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">

<CardTitle className="text-base font-medium">Real-Time Market Alerts</CardTitle>

<Bell className="h-4 w-4 text-muted-foreground" />

</CardHeader>

<CardContent>

<Table>

<TableHeader>

<TableRow>

<TableHead>Type</TableHead>

<TableHead>Alert</TableHead>

<TableHead>Time</TableHead>

</TableRow>

</TableHeader>

<TableBody>

{alerts.map((alert) => (

<TableRow key={alert.id}>

<TableCell className="font-medium">{alert.type}</TableCell>

<TableCell>{alert.message}</TableCell>

<TableCell>{alert.time}</TableCell>

</TableRow>

))}

</TableBody>

</Table>

</CardContent>

</Card>

)

}

"use client"

import { useMemo } from "react"

import { Map } from "lucide-react"

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

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

import type { RegionalData } from "@/types"

interface RegionalMapProps {

data: RegionalData[]

onRegionSelect: (region: RegionalData) => void

}

export default function RegionalMap({ data, onRegionSelect }: RegionalMapProps) {

const regions = useMemo(

() => [

{ id: "NCR", name: "National Capital Region" },

{ id: "R1", name: "Ilocos Region" },

{ id: "R2", name: "Cagayan Valley" },

{ id: "R3", name: "Central Luzon" },

{ id: "R4A", name: "CALABARZON" },

{ id: "R4B", name: "MIMAROPA" },

{ id: "R5", name: "Bicol Region" },

{ id: "R6", name: "Western Visayas" },

{ id: "R7", name: "Central Visayas" },

{ id: "R8", name: "Eastern Visayas" },

{ id: "R9", name: "Zamboanga Peninsula" },

{ id: "R10", name: "Northern Mindanao" },

{ id: "R11", name: "Davao Region" },

{ id: "R12", name: "SOCCSKSARGEN" },

{ id: "R13", name: "Caraga" },

{ id: "BARMM", name: "Bangsamoro" },

],

[],

)

return (

<Card>

<CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">

<CardTitle className="text-base font-medium">Regional Investment Map</CardTitle>

<Map className="h-4 w-4 text-muted-foreground" />

</CardHeader>

<CardContent>

<div className="mb-4">

<Select

onValueChange={(value) => {

const region = data.find((r) => r.id === value)

if (region) onRegionSelect(region)

}}

>

<SelectTrigger>

<SelectValue placeholder="Select a region" />

</SelectTrigger>

<SelectContent>

{regions.map((region) => (

<SelectItem key={region.id} value={region.id}>

{region.name}

</SelectItem>

))}

</SelectContent>

</Select>

</div>

<div className="grid gap-4">

{data.map((region) => (

<div

key={region.id}

className="flex items-center justify-between rounded-lg border p-4 hover:bg-accent"

role="button"

onClick={() => onRegionSelect(region)}

>

<div>

<h3 className="font-medium">{region.name}</h3>

<p className="text-sm text-muted-foreground">GDP Growth: {region.gdpGrowth}%</p>

</div>

<div className="text-right">

<p className="text-sm font-medium">Investment Score: {region.investmentScore}</p>

<p className="text-sm text-muted-foreground">Labor Force: {region.laborForce.toLocaleString()}</p>

</div>

</div>

))}

</div>

</CardContent>

</Card>

)

}

"use client"

import { Shield } from "lucide-react"

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

import { Tooltip, TooltipContent, TooltipProvider, TooltipTrigger } from "@/components/ui/tooltip"

import type { RiskAnalysis } from "@/types"

interface RiskAnalysisCardProps {

data: RiskAnalysis

}

export default function RiskAnalysisCard({ data }: RiskAnalysisCardProps) {

const getRiskColor = (value: number) => {

if (value < 30) return "text-green-500"

if (value < 70) return "text-yellow-500"

return "text-red-500"

}

return (

<Card>

<CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">

<CardTitle className="text-base font-medium">Risk Analysis</CardTitle>

<Shield className="h-4 w-4 text-muted-foreground" />

</CardHeader>

<CardContent>

<TooltipProvider>

<div className="grid gap-4">

<Tooltip>

<TooltipTrigger asChild>

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

<span>Political Risk</span>

<span className={getRiskColor(data.political)}>{data.political}%</span>

</div>

</TooltipTrigger>

<TooltipContent>

<ul className="list-disc pl-4">

{data.details.political.map((detail, i) => (

<li key={i}>{detail}</li>

))}

</ul>

</TooltipContent>

</Tooltip>

<Tooltip>

<TooltipTrigger asChild>

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

<span>Crime Risk</span>

<span className={getRiskColor(data.crime)}>{data.crime}%</span>

</div>

</TooltipTrigger>

<TooltipContent>

<ul className="list-disc pl-4">

{data.details.crime.map((detail, i) => (

<li key={i}>{detail}</li>

))}

</ul>

</TooltipContent>

</Tooltip>

<Tooltip>

<TooltipTrigger asChild>

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

<span>Economic Risk</span>

<span className={getRiskColor(data.economy)}>{data.economy}%</span>

</div>

</TooltipTrigger>

<TooltipContent>

<ul className="list-disc pl-4">

{data.details.economy.map((detail, i) => (

<li key={i}>{detail}</li>

))}

</ul>

</TooltipContent>

</Tooltip>

</div>

</TooltipProvider>

<div className="mt-4 text-xs text-muted-foreground">

Last updated: {new Date(data.lastUpdated).toLocaleString()}

</div>

</CardContent>

</Card>

)

}

"use client"

import { Shield } from "lucide-react"

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

import { Progress } from "@/components/ui/progress"

import { Tooltip, TooltipContent, TooltipProvider, TooltipTrigger } from "@/components/ui/tooltip"

import type { RiskMetrics } from "@/types"

interface RiskAnalysisProps {

data: RiskMetrics

}

export default function RiskAnalysis({ data }: RiskAnalysisProps) {

const getRiskColor = (value: number) => {

if (value >= 70) return "bg-green-500"

if (value >= 40) return "bg-yellow-500"

return "bg-red-500"

}

return (

<Card>

<CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">

<CardTitle className="text-base font-medium">Risk Analysis</CardTitle>

<Shield className="h-4 w-4 text-muted-foreground" />

</CardHeader>

<CardContent>

<TooltipProvider>

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

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

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

<span className="text-sm font-medium">Political Stability</span>

<span className="text-sm text-muted-foreground">{data.political}%</span>

</div>

<Tooltip>

<TooltipTrigger asChild>

<Progress value={data.political} className={getRiskColor(data.political)} />

</TooltipTrigger>

<TooltipContent>

<ul className="list-disc pl-4">

{data.details.political.map((detail, i) => (

<li key={i}>{detail}</li>

))}

</ul>

</TooltipContent>

</Tooltip>

</div>

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

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

<span className="text-sm font-medium">Economic Outlook</span>

<span className="text-sm text-muted-foreground">{data.economic}%</span>

</div>

<Tooltip>

<TooltipTrigger asChild>

<Progress value={data.economic} className={getRiskColor(data.economic)} />

</TooltipTrigger>

<TooltipContent>

<ul className="list-disc pl-4">

{data.details.economic.map((detail, i) => (

<li key={i}>{detail}</li>

))}

</ul>

</TooltipContent>

</Tooltip>

</div>

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

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

<span className="text-sm font-medium">Infrastructure</span>

<span className="text-sm text-muted-foreground">{data.infrastructure}%</span>

</div>

<Tooltip>

<TooltipTrigger asChild>

<Progress value={data.infrastructure} className={getRiskColor(data.infrastructure)} />

</TooltipTrigger>

<TooltipContent>

<ul className="list-disc pl-4">

{data.details.infrastructure.map((detail, i) => (

<li key={i}>{detail}</li>

))}

</ul>

</TooltipContent>

</Tooltip>

</div>

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

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

<span className="text-sm font-medium">Workforce Availability</span>

<span className="text-sm text-muted-foreground">{data.workforce}%</span>

</div>

<Tooltip>

<TooltipTrigger asChild>

<Progress value={data.workforce} className={getRiskColor(data.workforce)} />

</TooltipTrigger>

<TooltipContent>

<ul className="list-disc pl-4">

{data.details.workforce.map((detail, i) => (

<li key={i}>{detail}</li>

))}

</ul>

</TooltipContent>

</Tooltip>

</div>

</div>

</TooltipProvider>

</CardContent>

</Card>

)

}

// lib/api.ts

import type { InvestmentAlert, RegionalData, RiskAnalysis } from "@/types"

export async function fetchInvestmentAlerts(): Promise<InvestmentAlert[]> {

// Replace with your actual API call

return []

}

export async function fetchMarketSentiment(): Promise<number[]> {

// Replace with your actual API call

return []

}

export async function fetchRegionalData(region: string): Promise<RegionalData | null> {

// Replace with your actual API call

return null

}

export async function fetchRiskAnalysis(region: string): Promise<RiskAnalysis | null> {

// Replace with your actual API call

return null

}

import { clsx, type ClassValue } from "clsx"

import { twMerge } from "tailwind-merge"

export function cn(...inputs: ClassValue[]) {

return twMerge(clsx(inputs))

}

export interface BusinessRequirement {

id: string

companyName: string

industry: string

investmentSize: number

employmentTarget: number

preferredLocations?: string[]

infrastructureNeeds: {

power: boolean

water: boolean

internet: boolean

transportation: boolean

ports: boolean

}

workforceNeeds: {

skilled: number

unskilled: number

technical: number

}

spaceRequirement: {

type: "land" | "office" | "industrial"

size: number // in square meters

}

timeline: string

environmentalFactors: string[]

additionalRequirements: string[]

contactPerson: {

name: string

position: string

email: string

phone: string

}

}

export interface RegionalProfile {

id: string

name: string

province: string

population: number

workforce: {

skilled: number

unskilled: number

technical: number

}

infrastructure: {

power: {

available: boolean

capacity: number

reliability: number

}

water: {

available: boolean

capacity: number

quality: number

}

internet: {

available: boolean

speed: number

providers: number

}

transportation: {

airports: number

seaports: number

highways: number

}

}

landAvailability: {

industrial: number

commercial: number

agricultural: number

}

incentives: {

tax: string[]

other: string[]

}

contacts: {

government: GovernmentContact[]

department: DepartmentContact[]

}

naturalResources: string[]

majorIndustries: string[]

educationalInstitutions: number

costOfLiving: number

qualityOfLife: number

}

export interface GovernmentContact {

id: string

name: string

position: string

department: string

email: string

phone: string

responseTime?: number // in hours

}

export interface DepartmentContact {

id: string

name: string

role: string

email: string

phone: string

}

export interface MatchResult {

id: string

businessId: string

regionId: string

matchScore: number

matchFactors: {

factor: string

score: number

details: string

}[]

status: "pending" | "contacted" | "responded" | "escalated" | "matched" | "rejected"

timeline: {

created: string

contacted?: string

responded?: string

escalated?: string

matched?: string

rejected?: string

}

notes: string[]

}

export interface EmailTemplate {

type: "business" | "government" | "escalation"

subject: string

body: string

attachments?: string[]

}

"use client"

import { useEffect, useState } from "react"

import type { InvestmentIncentive, InvestmentOpportunity, MarketTrend, RegionalData, RiskMetrics } from "@/types"

import InvestmentIncentives from "@/components/investment-incentives"

import InvestmentOpportunities from "@/components/investment-opportunities"

import MarketTrends from "@/components/market-trends"

import RegionalMap from "@/components/regional-map"

import RiskAnalysis from "@/components/risk-analysis"

// Simulated data fetching functions

const fetchMockData = () => {

const regionalData: RegionalData[] = [

{

id: "NCR",

name: "National Capital Region",

population: 13484462,

gdpGrowth: 7.2,

laborForce: 6500000,

averageWage: 22000,

infrastructureScore: 85,

investmentScore: 90,

coordinates: [14.6091, 120.9876],

},

// Add more regions...

]

const incentives: InvestmentIncentive[] = [

{

id: "1",

type: "BOI",

title: "Tax Holiday for Tech Companies",

description: "4-6 year income tax holiday for tech companies",

requirements: ["Minimum investment of $1M", "Create 50 local jobs"],

benefits: ["Income tax holiday", "Duty-free importation"],

region: "NCR",

expiryDate: "2024-12-31",

},

// Add more incentives...

]

const riskMetrics: RiskMetrics = {

political: 75,

economic: 82,

infrastructure: 68,

workforce: 88,

details: {

political: ["Stable government", "Strong foreign relations"],

economic: ["Growing GDP", "Controlled inflation"],

infrastructure: ["Improving transport", "Digital infrastructure"],

workforce: ["Young population", "High education rate"],

},

}

const marketTrends: MarketTrend[] = Array.from({ length: 12 }, (\_, i) => ({

date: new Date(2024, i, 1).toISOString(),

fdi: Math.random() \* 1000 + 500,

gdpGrowth: Math.random() \* 3 + 5,

employmentRate: Math.random() \* 10 + 85,

}))

const opportunities: InvestmentOpportunity[] = [

{

id: "1",

sector: "Technology",

region: "NCR",

investmentSize: 1000000,

jobsCreated: 100,

incentives: ["Tax holiday", "Duty-free importation"],

description: "Tech hub development project",

status: "open",

},

// Add more opportunities...

]

return {

regionalData,

incentives,

riskMetrics,

marketTrends,

opportunities,

}

}

export default function InvestmentDashboard() {

const [data, setData] = useState<{

regionalData: RegionalData[]

incentives: InvestmentIncentive[]

riskMetrics: RiskMetrics

marketTrends: MarketTrend[]

opportunities: InvestmentOpportunity[]

} | null>(null)

useEffect(() => {

const mockData = fetchMockData()

setData(mockData)

}, [])

if (!data) {

return <div>Loading...</div>

}

return (

<div className="container mx-auto p-6">

<h1 className="mb-6 text-2xl font-bold">Philippines Investment Dashboard</h1>

<div className="grid gap-6 md:grid-cols-2 lg:grid-cols-3">

<RegionalMap

data={data.regionalData}

onRegionSelect={(region) => {

console.log("Selected region:", region)

}}

/>

<RiskAnalysis data={data.riskMetrics} />

<MarketTrends data={data.marketTrends} />

<InvestmentIncentives incentives={data.incentives} />

<div className="md:col-span-2">

<InvestmentOpportunities

opportunities={data.opportunities}

onSelect={(opportunity) => {

console.log("Selected opportunity:", opportunity)

}}

/>

</div>

</div>

</div>

)

}

/\*\* @type {import('tailwindcss').Config} */ module.exports = { darkMode: ["class"], content: [ "./pages/\*\*/*.{ts,tsx}", "./components/**/\*.{ts,tsx}", "./app/**/*.{ts,tsx}", "./src/\*\*/*.{ts,tsx}", "\*.{js,ts,jsx,tsx,mdx}", ], theme: { container: { center: true, padding: "2rem", screens: { "2xl": "1400px", }, }, extend: { colors: { border: "hsl(var(--border))", input: "hsl(var(--input))", ring: "hsl(var(--ring))", background: "hsl(var(--background))", foreground: "hsl(var(--foreground))", primary: { DEFAULT: "hsl(var(--primary))", foreground: "hsl(var(--primary-foreground))", }, secondary: { DEFAULT: "hsl(var(--secondary))", foreground: "hsl(var(--secondary-foreground))", }, destructive: { DEFAULT: "hsl(var(--destructive))", foreground: "hsl(var(--destructive-foreground))", }, muted: { DEFAULT: "hsl(var(--muted))", foreground: "hsl(var(--muted-foreground))", }, accent: { DEFAULT: "hsl(var(--accent))", foreground: "hsl(var(--accent-foreground))", }, popover: { DEFAULT: "hsl(var(--popover))", foreground: "hsl(var(--popover-foreground))", }, card: { DEFAULT: "hsl(var(--card))", foreground: "hsl(var(--card-foreground))", }, }, borderRadius: { lg: "var(--radius)", md: "calc(var(--radius) - 2px)", sm: "calc(var(--radius) - 4px)", }, keyframes: { "accordion-down": { from: { height: 0 }, to: { height: "var(--radix-accordion-content-height)" }, }, "accordion-up": { from: { height: "var(--radix-accordion-content-height)" }, to: { height: 0 }, }, }, animation: { "accordion-down": "accordion-down 0.2s ease-out", "accordion-up": "accordion-up 0.2s ease-out", }, }, }, plugins: [require("tailwindcss-animate")], }

export interface RegionalData {

id: string

name: string

population: number

gdpGrowth: number

laborForce: number

averageWage: number

infrastructureScore: number

investmentScore: number

coordinates: [number, number]

}

export interface InvestmentIncentive {

id: string

type: "BOI" | "PEZA" | "LGU"

title: string

description: string

requirements: string[]

benefits: string[]

region: string

expiryDate: string

}

export interface RiskMetrics {

political: number

economic: number

infrastructure: number

workforce: number

details: {

political: string[]

economic: string[]

infrastructure: string[]

workforce: string[]

}

}

export interface InvestmentOpportunity {

id: string

sector: string

region: string

investmentSize: number

jobsCreated: number

incentives: string[]

description: string

status: "open" | "pending" | "closed"

}

export interface MarketTrend {

date: string

fdi: number

gdpGrowth: number

employmentRate: number

}

"use client";

import { NextResponse } from "next/server"; import { MongoClient } from "mongodb"; import type { MatchResult } from "@/types"; import nodemailer from "nodemailer";

// MongoDB Connection Setup const client = new MongoClient(process.env.MONGO\_URI!); const db = client.db("investment-matching"); const matchesCollection = db.collection("matches");

// Email Transporter Setup const transporter = nodemailer.createTransport({ service: "[braydenmwalls1972@gmail.com](mailto:braydenmwalls1972@gmail.com)", auth: { user: process.env.EMAIL\_USER, pass: process.env.EMAIL\_PASS, }, });

export async function POST(req: Request) { try { const { match, escalation } = await req.json();

// Ensure match ID and necessary fields exist  
if (!match?.id || !match?.companyName) {  
 return NextResponse.json({ error: "Match ID and company name are required" }, { status: 400 });  
}  
  
// Store match in MongoDB  
await matchesCollection.updateOne(  
 { "match.id": match.id },  
 { $set: { match, escalation, createdAt: new Date() } },  
 { upsert: true }  
);  
  
// Send email notification  
const emailSent = await sendMatchNotification(match, escalation);  
  
if (!emailSent) {  
 // Schedule escalation if email fails  
 setTimeout(() => escalateNotification(match.id), 24 \* 60 \* 60 \* 1000); // 24-hour delay  
}  
  
return NextResponse.json({ success: true });

} catch (error) { console.error("Error in notification:", error); return NextResponse.json({ error: "Failed to send notifications" }, { status: 500 }); } }

async function sendMatchNotification(match: MatchResult, escalation: boolean): Promise { try { if (!match?.governmentEmail) { console.error("No government email provided for match:", match.id); return false; }

const mailOptions = {  
 from: process.env.EMAIL\_USER,  
 to: match.governmentEmail,  
 subject: `New Investment Match - ${match.companyName}`,  
 text: `A new business match has been found for ${match.companyName}. Please respond within 24 hours.`,  
};  
  
await transporter.sendMail(mailOptions);  
console.log(`Notification sent for match: ${match.id}`);  
return true;

} catch (error) { console.error(Failed to send email for match: ${match.id}, error); return false; } }

// Escalation Function (if no response in 24 hours) async function escalateNotification(matchId: string) { const matchEntry = await matchesCollection.findOne({ "match.id": matchId }); if (!matchEntry || matchEntry.escalation || !matchEntry.match) return;

console.log(Escalating match ${matchId} to mayor/governor.); await matchesCollection.updateOne({ "match.id": matchId }, { $set: { escalation: true } });

const recipientEmail = matchEntry.match.mayorEmail || matchEntry.match.governorEmail; if (!recipientEmail) { console.error(No escalation contact found for match: ${matchId}); return; }

await sendMatchNotification({ ...matchEntry.match, governmentEmail: recipientEmail, }, true); }

import { Configuration, OpenAIApi } from "openai-edge"; import { Redis } from "@upstash/redis";

const redis = new Redis({ url: process.env.REDIS\_URL, token: process.env.REDIS\_TOKEN });

const config = new Configuration({ apiKey: process.env.OPENAI\_API\_KEY }); const openai = new OpenAIApi(config);

export default async function handler(req, res) { if (req.method !== "POST") return res.status(405).json({ error: "Method not allowed" });

try { const { requirement, governmentNeeds, inquiryId } = req.body;

// Fetch regional investment data (cached)  
let regions = await redis.get("investment-data");  
if (!regions) {  
 regions = await fetchInvestmentData();  
 await redis.set("investment-data", regions, { ex: 3600 });  
}  
  
// AI-Powered Matching for Investors  
const matchResults = await getInvestmentMatch(requirement, regions);  
const privateMatches = matchResults.filter(m => m.type === "private").slice(0, 10);  
const governmentMatches = matchResults.filter(m => m.type === "government").slice(0, 10);  
  
// AI Chat Interaction  
const chatResponse = await getAIChat(requirement);  
  
// Investment Failure Analysis  
const failureAnalysis = analyzeInvestmentFailures(matchResults);  
  
// Global Government Matching  
const globalMatches = await getInvestmentMatch(governmentNeeds, regions);  
  
// Handle 72-Hour Auto-Escalation  
if (inquiryId) await handleEscalation(inquiryId);  
  
res.status(200).json({  
 privateMatches,  
 governmentMatches,  
 chatResponse,  
 failureAnalysis,  
 globalMatches  
});

} catch (error) { console.error("Error:", error); res.status(500).json({ error: "Failed to process investment request" }); } }

// Fetch Investment Data from DB async function fetchInvestmentData() { return [ { name: "Region A", type: "private", infrastructureNeeds: 80, incentives: 60, growthPotential: 90 }, { name: "Region B", type: "government", infrastructureNeeds: 70, incentives: 80, growthPotential: 85 } ]; }

// AI Investment Matching async function getInvestmentMatch(requirement, regions) { const prompt = Match this investor with the best region for investment: Investor Requirement: ${JSON.stringify(requirement, null, 2)} Available Regions: ${JSON.stringify(regions, null, 2)} ;

const response = await openai.createChatCompletion({ model: "gpt-4", messages: [{ role: "user", content: prompt }], stream: false });

return JSON.parse(response.data.choices[0]?.message?.content || "[]"); }

// AI Chat for Interaction async function getAIChat(requirement) { const chatPrompt = Investor has asked about: ${JSON.stringify(requirement, null, 2)}; const response = await openai.createChatCompletion({ model: "gpt-4", messages: [{ role: "user", content: chatPrompt }], stream: false });

return response.data.choices[0]?.message?.content; }

// Investment Failure Analysis function analyzeInvestmentFailures(pastInvestments) { const failureReasons = {}; pastInvestments.forEach(investment => { if (!investment.success) { investment.issues.forEach(issue => { failureReasons[issue] = (failureReasons[issue] || 0) + 1; }); } }); return Object.entries(failureReasons).sort((a, b) => b[1] - a[1]).map(([issue, count]) => ({ issue, count })); }

// 72-Hour Auto-Escalation async function handleEscalation(inquiryId) { let inquiry = await redis.get(inquiry:${inquiryId}); if (!inquiry) return;

const elapsedTime = Date.now() - inquiry.timestamp; if (elapsedTime > 72 \* 60 \* 60 \* 1000) { inquiry.assignedTo = inquiry.escalated ? "another region" : "higher official"; inquiry.escalated = true; await redis.set(inquiry:${inquiryId}, inquiry); } }