<https://llamacoder.together.ai/share/v2/eDjJvgJC376ykiM3>

import { useState, useEffect } from 'react'

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

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

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

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

import { Bell } from "lucide-react"

import { format } from "date-fns"

const mockBitcoinPrice = () => {

// Simulate a real-time Bitcoin price update around $96,587.57

const basePrice = 96587.57

const fluctuation = Math.random() \* 1000 - 500 // Random fluctuation between -500 and 500

return basePrice + fluctuation

}

export default function BitcoinPriceMonitor() {

const [bitcoinPrice, setBitcoinPrice] = useState<number>(mockBitcoinPrice())

const [lowTarget, setLowTarget] = useState<number | null>(null)

const [highTarget, setHighTarget] = useState<number | null>(null)

const [notifications, setNotifications] = useState<{ date: string; message: string; type: 'low' | 'high' }[]>([])

useEffect(() => {

const interval = setInterval(() => {

const newPrice = mockBitcoinPrice()

setBitcoinPrice(newPrice)

const now = new Date()

const formattedTime = format(now, "yyyy-MM-dd HH:mm:ss")

if (lowTarget !== null && newPrice <= lowTarget) {

const message = `Bitcoin price hit low target: $${lowTarget.toLocaleString(undefined, { minimumFractionDigits: 2, maximumFractionDigits: 2 })} USD`

setNotifications((prevNotifications) => [...prevNotifications, { date: formattedTime, message, type: 'low' }])

}

if (highTarget !== null && newPrice >= highTarget) {

const message = `Bitcoin price hit high target: $${highTarget.toLocaleString(undefined, { minimumFractionDigits: 2, maximumFractionDigits: 2 })} USD`

setNotifications((prevNotifications) => [...prevNotifications, { date: formattedTime, message, type: 'high' }])

}

}, 5000) // Update every 5 seconds

return () => clearInterval(interval)

}, [lowTarget, highTarget])

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

const value = parseFloat(e.target.value)

setLowTarget(isNaN(value) ? null : value)

}

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

const value = parseFloat(e.target.value)

setHighTarget(isNaN(value) ? null : value)

}

const clearNotifications = () => {

setNotifications([])

}

return (

<div className="min-h-screen bg-white flex flex-col items-center justify-center p-4">

<Card className="w-full max-w-md">

<CardHeader>

<CardTitle>Bitcoin Price Monitor</CardTitle>

<CardDescription>Monitor Bitcoin prices in real-time and set price targets for notifications.</CardDescription>

</CardHeader>

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

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

<div className="text-2xl font-bold">${bitcoinPrice.toLocaleString(undefined, { minimumFractionDigits: 2, maximumFractionDigits: 2 })} USD</div>

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

<Bell className="mr-2 h-4 w-4" /> Clear Notifications

</Button>

</div>

{notifications.length > 0 && (

<div className="mt-4">

<h3 className="text-lg font-bold mb-2">Notifications</h3>

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

{notifications.map((notification, index) => (

<li key={index} className={`p-2 rounded ${notification.type === 'low' ? 'bg-red-100 text-red-700' : 'bg-green-100 text-green-700'}`}>

<p className="font-bold">{notification.date}</p>

<p>{notification.message}</p>

</li>

))}

</ul>

</div>

)}

<div className="grid grid-cols-2 gap-4">

<div>

<Label htmlFor="low-target">Low Target (USD)</Label>

<Input

id="low-target"

type="number"

value={lowTarget !== null ? lowTarget.toString() : ''}

onChange={handleLowTargetChange}

placeholder="Enter low target"

/>

</div>

<div>

<Label htmlFor="high-target">High Target (USD)</Label>

<Input

id="high-target"

type="number"

value={highTarget !== null ? highTarget.toString() : ''}

onChange={handleHighTargetChange}

placeholder="Enter high target"

/>

</div>

</div>

</CardContent>

</Card>

</div>

)

}