import React, { useState, useRef } from 'react';

import { Play, Pause, Download } from 'lucide-react';

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

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

interface VideoPlayerProps {

  images: Array<{

    src: string;

    caption: string;

    date: string;

  }>;

  onVideoStateChange?: (isPlaying: boolean) => void;

}

const VideoPlayer = ({ images, onVideoStateChange }: VideoPlayerProps) => {

  const [isOpen, setIsOpen] = useState(false);

  const [isPlaying, setIsPlaying] = useState(false);

  const [currentImageIndex, setCurrentImageIndex] = useState(0);

  const [videoAudioPlaying, setVideoAudioPlaying] = useState(false);

  const videoAudioRef = useRef<HTMLAudioElement>(null);

  const handleOpenChange = (isOpen2: boolean) => {

    setCurrentImageIndex(0);

    setIsPlaying(!isOpen2);

    setIsOpen(isOpen2);

    onVideoStateChange(isOpen2);

  }

  const playVideo = () => {

    // If you have an actual audio file:

    if (videoAudioRef.current) {

      if (isPlaying) {

        videoAudioRef.current.pause();

      } else {

        videoAudioRef.current.play();

      }

    }

    if (!isPlaying && isOpen) {

      setIsPlaying(true);

      setVideoAudioPlaying(true);

      onVideoStateChange?.(true);

      let index = 0;

      const interval = setInterval(() => {

        if (index < images.length - 1) {

          index++;

          console.log(isOpen);

          setCurrentImageIndex(index);

        } else {

          setIsPlaying(false);

          setVideoAudioPlaying(false);

          onVideoStateChange?.(false);

          setCurrentImageIndex(0);

          clearInterval(interval);

        }

      }, 2000);

      (window as any).videoInterval = interval;

    } else {

    setCurrentImageIndex(0);

      setIsPlaying(false);

      setVideoAudioPlaying(false);

      onVideoStateChange?.(false);

      if ((window as any).videoInterval) {

        clearInterval((window as any).videoInterval);

      }

    }

  };

  const downloadVideo = () => {

    // Create canvas to generate video frames

    const canvas = document.createElement('canvas');

    const ctx = canvas.getContext('2d');

    canvas.width = 800;

    canvas.height = 600;

    // Simulate video download

    const link = document.createElement('a');

    link.download = 'notre-histoire-damour.mp4';

    link.href = '#'; // In real implementation, this would be the video blob URL

    // Show download notification

    alert('Téléchargement de la mini-vidéo démarré ! 💕');

    // In a real implementation, you would:

    // 1. Create video frames from images

    // 2. Generate MP4 using a library like FFmpeg.js

    // 3. Create blob URL and trigger download

  };

  // let currentImageIndex = currentImageIndex0;

  console.log(currentImageIndex);

  return (

    <>

      <Button

        variant="default"

        size="lg"

        onClick={() => setIsOpen(true)}

        className="bg-romantic-pink hover:bg-romantic-pink/90 text-white shadow-lg"

      >

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

        Voir la Mini-Vidéo

      </Button>

      <Dialog open={isOpen} onOpenChange={handleOpenChange}>

        <DialogContent className="max-w-lg">

          <DialogHeader>

            <DialogTitle className="text-center text-2xl" style={{ fontFamily: 'Playfair Display, serif' }}>

              Notre Histoire en Vidéo

            </DialogTitle>

          </DialogHeader>

          <div className="relative">

            {/\* Zone de lecture vidéo \*/}

            <div className="relative aspect-[3/4] bg-black rounded-lg overflow-hidden">

              {/\* Video Audio \*/}

              <audio ref={videoAudioRef} loop preload='auto'>

                {/\* Add your video music file here \*/}

                <source src="/audio/promesse.mp3" type="audio/mpeg" />

              </audio>

              <img

                src={images[currentImageIndex]?.src}

                alt={images[currentImageIndex]?.caption}

                className="w-full h-full object-cover transition-all duration-500"

              />

              {/\* Overlay avec info de l'image \*/}

              <div className="absolute bottom-0 left-0 right-0 bg-gradient-to-t from-black/80 to-transparent p-4">

                <p className="text-white font-medium">{images[currentImageIndex]?.caption}</p>

                <p className="text-white/70 text-sm">{images[currentImageIndex]?.date}</p>

              </div>

              {

                open && (

                    <div className="absolute top-4 left-4 right-4">

                      <div className="h-1 bg-white/30 rounded-full overflow-hidden">

                        <div

                          className="h-full bg-romantic-pink transition-all duration-100"

                          style={{ width: `${((currentImageIndex + 1) / images.length) \* 100}%` }}

                        />

                      </div>

                      <p className="text-white text-sm mt-2">

                        {currentImageIndex + 1} / {images.length}

                      </p>

                    </div>

                )

              }

            </div>

            {/\* Contrôles \*/}

            <div className="flex items-center justify-center gap-4 mt-6">

              <Button

                variant="outline"

                size="lg"

                onClick={() => {playVideo()}}

                className="flex items-center gap-2"

              >

                {isPlaying ? (

                  <>

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

                    Pause

                  </>

                ) : (

                  <>

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

                    Lire

                  </>

                )}

              </Button>

              <Button

                variant="outline"

                size="lg"

                onClick={downloadVideo}

                className="flex items-center gap-2"

              >

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

                Télécharger

              </Button>

            </div>

          </div>

        </DialogContent>

      </Dialog>

    </>

  );

};

export default VideoPlayer;

Pourquoi est ce que l’incrementation de currentImageIndex ne s’arrete pas apres la fermeture de dialog?