import cv2  
import requests  
import time  
import threading  
import tkinter as tk  
from PIL import Image, ImageTk  
  
class LecteurVideo:  
 def \_\_init\_\_(self, parent, videos):  
 self.parent = parent  
 self.videos = videos  
 self.video\_index = 0  
 self.debut\_lecture = 0  
   
 self.arret\_demande = threading.Event()   
 self.playback\_thread = None   
   
 self.initialiser\_fenetre\_video()  
   
 def initialiser\_fenetre\_video(self):  
 self.fenetre\_video = tk.Toplevel(self.parent)  
 self.fenetre\_video.title("Lecture Vidéo")  
 self.fenetre\_video.geometry("800x600")  
 self.canvas\_video = tk.Label(self.fenetre\_video)  
 self.canvas\_video.pack()  
   
 def marquer\_video\_courante(self, id\_video):  
 requests.post('http://localhost:5000/video/current', json={"id\_video": id\_video})  
  
 def terminer\_video\_courante(self, id\_video, temps\_jouer):  
 requests.put('http://localhost:5000/video/current', json={"id\_video": id\_video, "temps\_jouer": temps\_jouer})  
 self.parent.stats = self.parent.obtenir\_stats\_jour()  
 self.parent.mise\_a\_jour\_ui\_avec\_stats(self.parent.stats, id\_video)  
  
 # On debute les threads pour la lecture des vidéos  
 def debuter\_video\_playback(self):  
 # Si un thread de lecture est déjà en cours, on l'arrête  
 if self.playback\_thread is not None and self.playback\_thread.is\_alive():  
 self.arret\_demande.set()   
 self.playback\_thread.join()   
   
 self.arret\_demande.clear()   
 self.playback\_thread = threading.Thread(target=self.demarrer\_video)   
 self.playback\_thread.start()  
  
  
 def demarrer\_video(self):  
 while self.video\_index < len(self.videos) and not self.arret\_demande.is\_set():  
 video = self.videos[self.video\_index]  
 self.parent.afficher\_nom\_video(video['nom\_video'])  
 video\_path = "./client/videos/" + video['nom\_video']  
 id\_video = video['id\_video']  
 print("demarrer video : " +str(id\_video) + " " + video\_path)  
   
 self.marquer\_video\_courante(id\_video)  
   
 cap = cv2.VideoCapture(video\_path)  
 self.debut\_lecture = time.time()  
  
 while cap.isOpened() and not self.arret\_demande.is\_set():  
 ret, frame = cap.read()  
 if ret:  
 # Convertir l'image en RGB pour l'affichage dans Tkinter  
 frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)  
 image = Image.fromarray(frame)  
 photo = ImageTk.PhotoImage(image=image)  
  
 self.canvas\_video.config(image=photo)  
 self.canvas\_video.image = photo  
 self.canvas\_video.update\_idletasks()   
 time.sleep(1/30)  
 if self.arret\_demande.is\_set():  
 break  
 if self.arret\_demande.is\_set():  
 break  
   
   
 # Calculer la durée de la lecture  
 duree\_lecture = time.time() - self.debut\_lecture  
 cap.release()  
   
 self.terminer\_video\_courante(id\_video, duree\_lecture)  
   
 self.video\_index += 1  
 if self.video\_index >= len(self.videos) or self.arret\_demande:  
 # Toutes les vidéos ont déjà été jouées ou arrêt demandé  
 print("Toutes les vidéos ont été jouées ou arrêt demandé.")  
 self.parent.afficher\_nom\_video(" ")  
 self.parent.lecteur\_video\_actuel = None  
 self.parent.videos\_en\_lecture = False  
 self.fenetre\_video.destroy()  
 break   
   
  
 def arreter\_lecture(self):  
 if self.video\_index > 0 and self.video\_index <= len(self.videos):  
 id\_video = self.videos[self.video\_index - 1]['id\_video']   
 temps\_jouer = time.time() - self.debut\_lecture   
 self.terminer\_video\_courante(id\_video, temps\_jouer)  
   
 self.arret\_demande = True  
 self.video\_index = 0   
 self.parent.videos\_en\_lecture = False   
 self.fenetre\_video.destroy()   
   
 def passer\_au\_video\_suivant(self):  
 if not self.playback\_thread or not self.playback\_thread.is\_alive():  
 print("Aucune vidéo en cours de lecture.")  
 return  
  
 if self.video\_index >= 0 and self.video\_index < len(self.videos):  
 id\_video = self.videos[self.video\_index]['id\_video']  
 temps\_jouer = time.time() - self.debut\_lecture   
 print("Passer au vidéo suivante : " + str(id\_video) + " " + str(temps\_jouer))  
 self.terminer\_video\_courante(id\_video, temps\_jouer)  
  
 self.arret\_demande.set()  
   
 self.verifier\_fin\_thread()  
  
  
 def verifier\_fin\_thread(self):  
 if self.playback\_thread.is\_alive():  
 self.fenetre\_video.after(100, self.verifier\_fin\_thread)  
 else:  
 self.on\_fin\_thread()  
  
 def on\_fin\_thread(self):  
 self.video\_index += 1  
 if self.video\_index >= len(self.videos):  
 print("Toutes les vidéos ont déjà été jouées.")  
 self.video\_index = 0   
  
 self.arret\_demande.clear()  
 print("Passer au vidéo suivante.")  
 self.debuter\_video\_playback()