import React, { useRef, useEffect } from "react";

export default function CameraDetection() {

  const videoRef = useRef(null);

  const canvasRef = useRef(null);

  const wsRef = useRef(null);

  const detectionsRef = useRef({ bbox: [], keypoints: [] });

  useEffect(() => {

    // Connect WebSocket

    const ws = new WebSocket("ws://127.0.0.1:8000/ws");

    wsRef.current = ws;

    ws.onopen = () => console.log("✅ Connected to backend WebSocket");

    ws.onerror = (err) => console.error("❌ WebSocket error:", err);

    ws.onclose = () => console.warn("⚠️ WebSocket closed");

    ws.onmessage = (event) => {

      try {

        detectionsRef.current = JSON.parse(event.data);

      } catch (err) {

        console.error("Invalid JSON:", event.data);

      }

    };

    // Setup camera

    async function setupCamera() {

      try {

        const stream = await navigator.mediaDevices.getUserMedia({ video: true });

        if (videoRef.current) {

          videoRef.current.srcObject = stream;

          videoRef.current.onloadedmetadata = () => {

            videoRef.current.play().catch(err => console.warn("Play error:", err));

            canvasRef.current.width = videoRef.current.videoWidth;

            canvasRef.current.height = videoRef.current.videoHeight;

          };

        }

      } catch (err) {

        console.error("Camera error:", err);

      }

    }

    setupCamera();

    // Draw loop

    function draw() {

      if (!videoRef.current || !canvasRef.current) return;

      const ctx = canvasRef.current.getContext("2d");

      ctx.drawImage(videoRef.current, 0, 0, canvasRef.current.width, canvasRef.current.height);

      const { bbox = [], keypoints = [] } = detectionsRef.current;

      ctx.strokeStyle = "lime";

      ctx.lineWidth = 2;

      bbox.forEach(([x1, y1, x2, y2]) => {

        ctx.strokeRect(x1, y1, x2 - x1, y2 - y1);

      });

      ctx.fillStyle = "red";

      keypoints.forEach((kpList) => {

        kpList.forEach(([x, y]) => {

          ctx.beginPath();

          ctx.arc(x, y, 4, 0, 2 \* Math.PI);

          ctx.fill();

        });

      });

      requestAnimationFrame(draw);

    }

    requestAnimationFrame(draw);

    // Send frames

    const interval = setInterval(() => {

      if (!videoRef.current || ws.readyState !== WebSocket.OPEN) return;

      if (videoRef.current.videoWidth === 0 || videoRef.current.videoHeight === 0) return;

      const hiddenCanvas = document.createElement("canvas");

      hiddenCanvas.width = videoRef.current.videoWidth;

      hiddenCanvas.height = videoRef.current.videoHeight;

      const ctx = hiddenCanvas.getContext("2d");

      ctx.drawImage(videoRef.current, 0, 0);

      const base64Data = hiddenCanvas.toDataURL("image/jpeg").split(",")[1];

      ws.send(base64Data);

    }, 200);

    return () => {

      clearInterval(interval);

      ws.close();

    };

  }, []);

  return (

    <div style={{ position: "relative", display: "inline-block" }}>

      <canvas ref={canvasRef} style={{ border: "2px solid black" }} />

      <video ref={videoRef} autoPlay playsInline style={{ display: "none" }} />

    </div>

  );

}