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

import axios from 'axios';

import './FacialAuth.css';

// Backend API base URL

const API\_BASE\_URL = 'http://localhost:5001';

interface FacialAuthProps {

  onAuthSuccess: (username: string) => void;

  onSwitchToSignUp: () => void;

}

const FacialAuth: React.FC<FacialAuthProps> = ({ onAuthSuccess, onSwitchToSignUp }) => {

  const videoRef = useRef<HTMLVideoElement>(null);

  const canvasRef = useRef<HTMLCanvasElement>(null);

  const [isCapturing, setIsCapturing] = useState(false);

  const [authMessage, setAuthMessage] = useState('');

  const [authStatus, setAuthStatus] = useState<'idle' | 'loading' | 'success' | 'error'>('idle');

  const [username, setUsername] = useState('');

  const [backendStatus, setBackendStatus] = useState<'checking' | 'online' | 'offline'>('checking');

  useEffect(() => {

    // Initialize webcam when component mounts

    const initWebcam = async () => {

      try {

        console.log('Initializing webcam...');

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

        if (videoRef.current) {

          videoRef.current.srcObject = stream;

          console.log('Webcam initialized successfully');

        }

      } catch (error) {

        console.error('Error accessing webcam:', error);

        setAuthMessage('Error accessing webcam. Please allow camera access.');

        setAuthStatus('error');

      }

    };

    // Check backend status

    const checkBackendStatus = async () => {

      try {

        console.log('Checking backend connection...');

        await axios.get(`${API\_BASE\_URL}/api/users`, { timeout: 3000 });

        console.log('Backend connection successful');

        setBackendStatus('online');

      } catch (error) {

        console.error('Error connecting to backend:', error);

        setBackendStatus('offline');

        setAuthMessage('Error connecting to server. Please try again later.');

        setAuthStatus('error');

      }

    };

    initWebcam();

    checkBackendStatus();

    // Cleanup function to ensure camera is stopped when component unmounts

    return () => {

      stopCamera();

    };

  }, []);

  // Function to stop the camera

  const stopCamera = () => {

    if (videoRef.current && videoRef.current.srcObject) {

      console.log('Stopping camera');

      const tracks = (videoRef.current.srcObject as MediaStream).getTracks();

      tracks.forEach(track => track.stop());

      videoRef.current.srcObject = null;

    }

  };

  const retryBackendConnection = async () => {

    setBackendStatus('checking');

    setAuthMessage('Checking server connection...');

    setAuthStatus('loading');

    try {

      await axios.get(`${API\_BASE\_URL}/api/users`, { timeout: 3000 });

      console.log('Backend reconnection successful');

      setBackendStatus('online');

      setAuthMessage('Connected to server successfully!');

      setAuthStatus('idle');

      setTimeout(() => {

        setAuthMessage('');

      }, 2000);

    } catch (error) {

      console.error('Backend reconnection failed:', error);

      setBackendStatus('offline');

      setAuthMessage('Error connecting to server. Please try again later.');

      setAuthStatus('error');

    }

  };

  const captureImage = () => {

    if (!username.trim()) {

      console.log('Authentication failed: No username entered');

      setAuthMessage('Please enter your username');

      setAuthStatus('error');

      return;

    }

    if (backendStatus === 'offline') {

      setAuthMessage('Server is currently unreachable. Please try again later.');

      setAuthStatus('error');

      return;

    }

    if (videoRef.current && canvasRef.current) {

      setIsCapturing(true);

      setAuthStatus('loading');

      setAuthMessage('Analyzing face...');

      const video = videoRef.current;

      const canvas = canvasRef.current;

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

      if (context) {

        try {

          console.log('Capturing image from webcam...');

          // Set canvas dimensions to match video

          canvas.width = video.videoWidth;

          canvas.height = video.videoHeight;

          // Draw video frame on canvas with horizontal flip for mirroring effect

          // First save the context state

          context.save();

          // Flip horizontally for the mirror effect

          context.scale(-1, 1);

          // Draw the video but offset by negative width (because of the scale flip)

          context.drawImage(video, -canvas.width, 0, canvas.width, canvas.height);

          // Restore context to normal state

          context.restore();

          console.log('Image captured successfully with mirror effect');

          // Convert canvas to blob

          canvas.toBlob(async (blob) => {

            if (blob) {

              console.log('Image converted to blob, size:', blob.size);

              try {

                // Create form data with image and username

                const formData = new FormData();

                formData.append('image', blob, 'face.jpg');

                formData.append('username', username);

                console.log(`Sending authentication request for user "${username}"...`);

                // Send to backend for facial recognition

                const response = await axios.post(`${API\_BASE\_URL}/api/facial-recognition`, formData, {

                  headers: {

                    'Content-Type': 'multipart/form-data',

                  },

                  timeout: 10000, // 10 seconds timeout

                });

                console.log('Authentication response:', response.data);

                if (response.data.authenticated) {

                  console.log('Authentication successful for user:', username);

                  setAuthMessage('Authentication successful!');

                  setAuthStatus('success');

                  // Turn off camera after successful authentication

                  stopCamera();

                  // Call the onAuthSuccess callback

                  setTimeout(() => {

                    onAuthSuccess(username);

                  }, 1500);

                } else {

                  console.log('Authentication failed:', response.data.message);

                  setAuthMessage(response.data.message || 'Authentication failed. Please try again.');

                  setAuthStatus('error');

                  setIsCapturing(false);

                }

              } catch (error: any) {

                console.error('Error during facial authentication:', error);

                if (error.response) {

                  console.error('Response data:', error.response.data);

                  console.error('Response status:', error.response.status);

                  setAuthMessage(`Authentication error: ${error.response.data.error || error.response.statusText}`);

                } else if (error.request) {

                  console.error('No response received:', error.request);

                  setAuthMessage('No response from server. Please check if the backend server is running.');

                  setBackendStatus('offline');

                } else {

                  console.error('Error message:', error.message);

                  setAuthMessage(`Authentication error: ${error.message}`);

                }

                setAuthStatus('error');

                setIsCapturing(false);

              }

            } else {

              console.error('Failed to create blob from canvas');

              setAuthMessage('Failed to capture image. Please try again.');

              setAuthStatus('error');

              setIsCapturing(false);

            }

          }, 'image/jpeg', 0.95); // Higher quality JPEG

        } catch (error) {

          console.error('Error capturing image:', error);

          setAuthMessage('Error capturing image. Please try again.');

          setAuthStatus('error');

          setIsCapturing(false);

        }

      }

    }

  };

  return (

    <div className="facial-auth-container">

      <h2 className="auth-title">Facial Authentication</h2>

      <p className="auth-instruction">

        Please enter your username and position your face in the frame, then click the authenticate button.

      </p>

      {backendStatus === 'offline' && (

        <div className="backend-status-container">

          <div className="backend-status offline">

            <span>Backend server is unreachable</span>

            <button onClick={retryBackendConnection} className="retry-button">

              Retry Connection

            </button>

          </div>

        </div>

      )}

      <input

        type="text"

        value={username}

        onChange={(e) => setUsername(e.target.value)}

        placeholder="Enter your username"

        className="username-input"

      />

      <div className="video-container">

        <video

          ref={videoRef}

          autoPlay

          playsInline

          muted

          className="webcam-video mirrored"

          style={{

            display: isCapturing ? 'none' : 'block',

            transform: 'scaleX(-1)' // Flip the video horizontally for mirror effect

          }}

        />

        <canvas

          ref={canvasRef}

          className="webcam-canvas"

          style={{ display: isCapturing ? 'block' : 'none' }}

        />

        {authStatus === 'loading' && (

          <div className="loading-overlay">

            <div className="loading-spinner"></div>

          </div>

        )}

      </div>

      {authMessage && (

        <div

          className={`auth-message ${

            authStatus === 'success'

              ? 'success-message'

              : authStatus === 'error'

              ? 'error-message'

              : 'info-message'

          }`}

        >

          {authMessage}

        </div>

      )}

      <div className="button-container">

        <button

          onClick={captureImage}

          className="auth-button"

          disabled={isCapturing || backendStatus === 'offline'}

        >

          Authenticate

        </button>

      </div>

      <div className="auth-footer">

        <span>Don't have an account?</span>

        <button

          onClick={onSwitchToSignUp}

          className="switch-button"

        >

          Sign Up

        </button>

      </div>

    </div>

  );

};

export default FacialAuth;