```
// ================================================================================
// AI WILDLIFE MONITORING SYSTEM - COMPLETE CODE
// ================================================================================
// Description: Advanced wildlife monitoring and conservation tracking system
// Technology: React + Tailwind CSS + Claude AI API
// Author: Wildlife Conservation Team
// Last Updated: November 2025
// ================================================================================

import React, { useState, useRef } from 'react';
import { Camera, Upload, Loader2, AlertCircle, MapPin, Calendar, TrendingUp, Database } from 'lucide-react';

// ================================================================================
// MAIN COMPONENT
// ================================================================================

export default function WildlifeMonitor() {
  // --------------------------------------------------------------------------
  // STATE MANAGEMENT
  // --------------------------------------------------------------------------

  // Image handling states
  const [image, setImage] = useState(null);                  // Stores uploaded file object
  const [imagePreview, setImagePreview] = useState(null);     // Base64 preview string

  // Analysis states
  const [analyzing, setAnalyzing] = useState(false);          // Loading indicator
  const [result, setResult] = useState(null);                // AI analysis results
```

```
// Data management states
const [sightings, setSightings] = useState([]);           // Array of wildlife sightings
const [error, setError] = useState(null);                 // Error messages

// Reference for file input
const fileInputRef = useRef(null);                        // Hidden file input reference

// ---------------------------------------------------------------------------
// EVENT HANDLERS
// ---------------------------------------------------------------------------

/**
 * Handles image file upload and preview generation
 * @param {Event} e - File input change event
 */
const handleImageUpload = (e) => {
  const file = e.target.files[0];

  // Validate file exists and is an image
  if (file && file.type.startsWith('image/')) {
    setImage(file);

    // Create FileReader to convert image to base64 for preview
    const reader = new FileReader();
    reader.onloadend = () => {
      setImagePreview(reader.result);
    };
    reader.readAsDataURL(file);
```

```
    // Reset previous results
    setResult(null);
    setError(null);
  }
};

/**
 * Analyzes uploaded image using Claude AI API
 * Process: Convert to base64 → Send to API → Parse JSON response → Update state
 */
const analyzeImage = async () => {
  if (!image) return;

  setAnalyzing(true);
  setError(null);

  try {
    // Convert image to base64 for API transmission
    const reader = new FileReader();
    reader.onloadend = async () => {
      // Extract base64 data (remove data URL prefix)
      const base64Image = reader.result.split(',')[1];

      // API REQUEST: Send image to Claude for analysis
      const response = await fetch("https://api.anthropic.com/v1/messages", {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
```

```
        },
        body: JSON.stringify({
          model: "claude-sonnet-4-20250514",    // Claude Sonnet 4 model
          max_tokens: 1000,                     // Response length limit
          messages: [
            {
              role: "user",
              content: [
                // Image content block
                {
                  type: "image",
                  source: {
                    type: "base64",
                    media_type: image.type,      // e.g., "image/jpeg"
                    data: base64Image
                  }
                },
                // Text prompt for analysis
                {
                  type: "text",
                  text: `Analyze this wildlife image and provide a detailed monitoring report in JSON format.
Include:
{
  "species": "species name",
  "confidence": "percentage",
  "count": number of animals visible,
  "behavior": "observed behavior",
  "health": "health assessment",
  "habitat": "habitat type",
```

```
  "threats": ["list of potential threats"],
  "conservation_status": "status",
  "recommendations": ["monitoring recommendations"]
}

If no wildlife is detected, indicate that clearly. Be specific and scientific in your assessment.`
              }
             ]
            }
           ]
         })
       });

       // RESPONSE PROCESSING: Parse API response
       const data = await response.json();
       const textContent = data.content.find(c => c.type === 'text')?.text || '';

       // Extract JSON from response using regex
       const jsonMatch = textContent.match(/\{[\s\S]*\}/);
       if (jsonMatch) {
         const analysisResult = JSON.parse(jsonMatch[0]);
         setResult(analysisResult);

         // LOG SIGHTING: Create new sighting record
         const newSighting = {
           id: Date.now(),                          // Unique ID (Unix timestamp)
           timestamp: new Date().toISOString(),     // ISO 8601 format
           species: analysisResult.species,
           count: analysisResult.count,
```

```
        location: "Camera Trap #1",                    // Could be dynamic
        image: imagePreview
       };

       // Add to sightings array (keep last 10)
       setSightings(prev => [newSighting, ...prev].slice(0, 10));
      } else {
       setError("Unable to parse analysis results");
      }
    };
    reader.readAsDataURL(image);
  } catch (err) {
    setError("Analysis failed: " + err.message);
  } finally {
    setAnalyzing(false);
  }
};

// ---------------------------------------------------------------------------
// HELPER FUNCTIONS
// ---------------------------------------------------------------------------

/**
 * Maps conservation status to Tailwind color classes
 * @param {string} status - Conservation status
 * @returns {string} Tailwind CSS class
 */
const getStatusColor = (status) => {
  const colors = {
```

```
    'Critically Endangered': 'text-red-600',
    'Endangered': 'text-orange-600',
    'Vulnerable': 'text-yellow-600',
    'Near Threatened': 'text-blue-600',
    'Least Concern': 'text-green-600'
  };
  return colors[status] || 'text-gray-600';
};

// ---------------------------------------------------------------------
// RENDER COMPONENT
// ---------------------------------------------------------------------

return (
  <div className="min-h-screen bg-gradient-to-br from-green-50 to-emerald-100 p-6">
    <div className="max-w-7xl mx-auto">

      {/* ------------------------------------------------------------ */}
      {/* HEADER SECTION */}
      {/* ------------------------------------------------------------ */}
      <div className="text-center mb-8">
        <h1 className="text-4xl font-bold text-green-800 mb-2 flex items-center justify-center gap-3">
          <Camera className="w-10 h-10" />
          AI Wildlife Monitoring System
        </h1>
        <p className="text-green-700">Advanced species detection and conservation tracking</p>
      </div>

      {/* ------------------------------------------------------------ */}
```

```
      {/* MAIN CONTENT GRID */}
      {/* ------------------------------------------------------------- */}
      <div className="grid md:grid-cols-2 gap-6 mb-6">

        {/* ============================================================ */}
        {/* UPLOAD SECTION */}
        {/* ============================================================ */}
        <div className="bg-white rounded-xl shadow-lg p-6">
          <h2 className="text-xl font-bold text-gray-800 mb-4 flex items-center gap-2">
            <Upload className="w-5 h-5" />
            Upload Wildlife Image
          </h2>

          {/* Hidden file input (triggered by button) */}
          <input
            type="file"
            ref={fileInputRef}
            onChange={handleImageUpload}
            accept="image/*"
            className="hidden"
          />

          {/* File selection button */}
          <button
            onClick={() => fileInputRef.current?.click()}
            className="w-full bg-green-600 hover:bg-green-700 text-white py-3 rounded-lg font-semibold
mb-4 transition"
          >
            Select Image
```

```
        </button>

        {/* Image preview */}
        {imagePreview && (
          <div className="mb-4">
            <img
              src={imagePreview}
              alt="Preview"
              className="w-full h-64 object-cover rounded-lg border-4 border-green-200"
            />
          </div>
        )}

        {/* Analyze button (shown when image loaded and not analyzing) */}
        {image && !analyzing && (
          <button
            onClick={analyzeImage}
            className="w-full bg-emerald-600 hover:bg-emerald-700 text-white py-3 rounded-lg font-semibold transition"
          >
            Analyze Wildlife
          </button>
        )}

        {/* Loading indicator */}
        {analyzing && (
          <div className="flex items-center justify-center gap-2 text-green-700 py-3">
            <Loader2 className="w-5 h-5 animate-spin" />
            <span>Analyzing wildlife...</span>
```

```
  </div>
 )}

 {/* Error message */}
 {error && (
  <div className="flex items-center gap-2 text-red-600 bg-red-50 p-3 rounded-lg">
   <AlertCircle className="w-5 h-5" />
   <span>{error}</span>
  </div>
 )}
</div>

{/* ============================================================ */}
{/* ANALYSIS RESULTS SECTION */}
{/* ============================================================ */}
<div className="bg-white rounded-xl shadow-lg p-6">
 <h2 className="text-xl font-bold text-gray-800 mb-4 flex items-center gap-2">
  <TrendingUp className="w-5 h-5" />
  Analysis Results
 </h2>

 {result ? (
  <div className="space-y-4">

   {/* Species overview card */}
   <div className="bg-green-50 p-4 rounded-lg">
    <div className="flex justify-between items-start mb-2">
     <div>
      <h3 className="text-2xl font-bold text-green-800">{result.species}</h3>
```

```
      <p className="text-sm text-green-600">Confidence: {result.confidence}</p>
    </div>
    <div className="bg-green-600 text-white px-3 py-1 rounded-full text-sm font-semibold">
      Count: {result.count}
    </div>
  </div>
  <p className={`text-sm font-semibold ${getStatusColor(result.conservation_status)}`}>
    {result.conservation_status}
  </p>
</div>

{/* Behavior section */}
<div>
  <h4 className="font-semibold text-gray-700 mb-2">Behavior</h4>
  <p className="text-gray-600 text-sm">{result.behavior}</p>
</div>

{/* Health assessment */}
<div>
  <h4 className="font-semibold text-gray-700 mb-2">Health Assessment</h4>
  <p className="text-gray-600 text-sm">{result.health}</p>
</div>

{/* Habitat information */}
<div>
  <h4 className="font-semibold text-gray-700 mb-2">Habitat</h4>
  <p className="text-gray-600 text-sm">{result.habitat}</p>
</div>
```

```
    {/* Threats list */}
    {result.threats && result.threats.length > 0 && (
      <div>
        <h4 className="font-semibold text-red-700 mb-2">Potential Threats</h4>
        <ul className="list-disc list-inside text-sm text-gray-600">
          {result.threats.map((threat, idx) => (
            <li key={idx}>{threat}</li>
          ))}
        </ul>
      </div>
    )}

    {/* Recommendations list */}
    {result.recommendations && result.recommendations.length > 0 && (
      <div>
        <h4 className="font-semibold text-blue-700 mb-2">Recommendations</h4>
        <ul className="list-disc list-inside text-sm text-gray-600">
          {result.recommendations.map((rec, idx) => (
            <li key={idx}>{rec}</li>
          ))}
        </ul>
      </div>
    )}
  </div>
) : (
  /* Empty state */
  <div className="text-center text-gray-400 py-12">
    <Camera className="w-16 h-16 mx-auto mb-3 opacity-30" />
    <p>Upload and analyze an image to see results</p>
```

```
        </div>
      )}
    </div>
  </div>

  {/* ---------------------------------------------------------------- */}
  {/* SIGHTINGS LOG SECTION */}
  {/* ---------------------------------------------------------------- */}
  <div className="bg-white rounded-xl shadow-lg p-6">
    <h2 className="text-xl font-bold text-gray-800 mb-4 flex items-center gap-2">
      <Database className="w-5 h-5" />
      Recent Sightings Log
    </h2>

    {sightings.length > 0 ? (
      /* Sightings grid */
      <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4">
        {sightings.map((sighting) => (
          <div key={sighting.id} className="border border-gray-200 rounded-lg p-3 hover:shadow-md
transition">

            {/* Sighting thumbnail */}
            {sighting.image && (
              <img
                src={sighting.image}
                alt={sighting.species}
                className="w-full h-32 object-cover rounded mb-2"
              />
            )}
```

```
        {/* Sighting details */}
        <h3 className="font-bold text-gray-800">{sighting.species}</h3>

        {/* Timestamp */}
        <div className="flex items-center gap-2 text-sm text-gray-600 mt-1">
          <Calendar className="w-4 h-4" />
          <span>{new Date(sighting.timestamp).toLocaleString()}</span>
        </div>

        {/* Location */}
        <div className="flex items-center gap-2 text-sm text-gray-600 mt-1">
          <MapPin className="w-4 h-4" />
          <span>{sighting.location}</span>
        </div>

        {/* Animal count */}
        <p className="text-sm text-gray-500 mt-1">Count: {sighting.count}</p>
      </div>
    ))}
  </div>
 ) : (
  /* Empty state for sightings */
  <div className="text-center text-gray-400 py-8">
    <Database className="w-12 h-12 mx-auto mb-2 opacity-30" />
    <p>No sightings recorded yet</p>
  </div>
 )}
</div>
```

```
    </div>
  </div>
 );
}

// ============================================================================
// END OF FILE
// ============================================================================

/*
 * USAGE NOTES:
 *
 * 1. This component is ready to use in a React application
 * 2. Ensure Tailwind CSS is properly configured in your project
 * 3. Install required dependencies:
 *    npm install lucide-react
 *
 * 4. Import and use in your App.js:
 *    import WildlifeMonitor from './WildlifeMonitor';
 *
 *    function App() {
 *      return <WildlifeMonitor />;
 *    }
 *
 * 5. The Anthropic API is pre-configured for Claude Artifacts environment
 *    For local development, add API key handling as per README
 *
 * FEATURES:
 * - Image upload with preview
```

* - AI-powered species identification

* - Comprehensive wildlife analysis

* - Sightings logging and tracking

* - Responsive design

* - Error handling

*

* TECHNOLOGIES:

* - React 18+ with Hooks

* - Tailwind CSS for styling

* - Lucide React for icons

* - Claude Sonnet 4 API

* - FileReader API for image processing

* - Fetch API for HTTP requests

*/