import React from 'react';

// Sample data simulating predictions from various websites

// In a real application, this data would be fetched from a server that scrapes the websites.

const dummyMatchData = [

  {

    id: 'match\_1',

    homeTeam: 'Manchester Utd',

    awayTeam: 'Chelsea',

    date: 'Oct 15, 2025',

    predictions: {

      'Betensured.com': 'Home Win',

      'Zulubet': 'Draw',

      'Betwizard': 'Home Win',

      'Statarea.com': 'Home Win',

      'Soccervista': 'Draw',

      'Soccer Vital': 'Home Win',

      'Free Super Tips': 'Away Win',

      'Whoscored.com': 'Home Win',

      'Bet Genuine': 'Draw',

      'Forebet': 'Home Win',

    },

  },

  {

    id: 'match\_2',

    homeTeam: 'Liverpool',

    awayTeam: 'Man City',

    date: 'Oct 16, 2025',

    predictions: {

      'Betensured.com': 'Home Win',

      'Zulubet': 'Home Win',

      'Betwizard': 'Draw',

      'Statarea.com': 'Away Win',

      'Soccervista': 'Home Win',

      'Soccer Vital': 'Home Win',

      'Free Super Tips': 'Draw',

      'Whoscored.com': 'Home Win',

      'Bet Genuine': 'Draw',

      'Forebet': 'Home Win',

    },

  },

  {

    id: 'match\_3',

    homeTeam: 'Arsenal',

    awayTeam: 'Tottenham',

    date: 'Oct 17, 2025',

    predictions: {

      'Betensured.com': 'Away Win',

      'Zulubet': 'Away Win',

      'Betwizard': 'Away Win',

      'Statarea.com': 'Draw',

      'Soccervista': 'Away Win',

      'Soccer Vital': 'Draw',

      'Free Super Tips': 'Away Win',

      'Whoscored.com': 'Away Win',

      'Bet Genuine': 'Home Win',

      'Forebet': 'Away Win',

    },

  },

];

const PredictionAnalyzer = () => {

  const analyzePrediction = (predictions) => {

    // Count the occurrences of each prediction type

    const counts = {

      'Home Win': 0,

      'Draw': 0,

      'Away Win': 0,

    };

    const totalPredictions = Object.keys(predictions).length;

    for (const site in predictions) {

      const outcome = predictions[site];

      if (counts[outcome] !== undefined) {

        counts[outcome]++;

      }

    }

    // Find the most common prediction

    let mostCommonOutcome = 'No clear consensus';

    let maxCount = 0;

    for (const outcome in counts) {

      if (counts[outcome] > maxCount) {

        maxCount = counts[outcome];

        mostCommonOutcome = outcome;

      } else if (counts[outcome] === maxCount && maxCount > 0) {

        // Handle ties

        mostCommonOutcome = 'Tie';

      }

    }

    // Calculate the probability

    const probability = (maxCount / totalPredictions \* 100).toFixed(0);

    return {

      mostCommonOutcome,

      probability: `${probability}%`,

      breakdown: counts

    };

  };

  return (

    <div className="bg-gray-100 min-h-screen font-sans p-4 md:p-8">

      <div className="max-w-4xl mx-auto bg-white p-6 rounded-xl shadow-lg">

        <header className="text-center mb-8">

          <h1 className="text-3xl md:text-4xl font-bold text-gray-900 mb-2">Aggregate Football Prediction</h1>

          <p className="text-gray-600">Analyzing predictions from top football websites.</p>

        </header>

        <section>

          {dummyMatchData.map(match => {

            const analysis = analyzePrediction(match.predictions);

            const mostLikelyOutcome = analysis.mostCommonOutcome;

            const probability = analysis.probability;

            let resultColor = 'text-gray-600';

            if (mostLikelyOutcome === 'Home Win') {

              resultColor = 'text-green-600';

            } else if (mostLikelyOutcome === 'Away Win') {

              resultColor = 'text-red-600';

            } else if (mostLikelyOutcome === 'Draw') {

              resultColor = 'text-blue-600';

            } else if (mostLikelyOutcome === 'Tie') {

              resultColor = 'text-purple-600';

            }

            return (

              <div key={match.id} className="bg-gray-50 p-6 rounded-xl shadow-md border border-gray-200 mb-6">

                <div className="flex flex-col md:flex-row justify-between items-center mb-4">

                  <div>

                    <h2 className="text-xl md:text-2xl font-bold text-gray-800">{match.homeTeam} vs {match.awayTeam}</h2>

                    <p className="text-sm text-gray-500">{match.date}</p>

                  </div>

                  <div className="mt-4 md:mt-0 text-center md:text-right">

                    <p className="text-lg font-semibold text-gray-700">Most Likely Outcome:</p>

                    <p className={`text-2xl font-extrabold ${resultColor}`}>{mostLikelyOutcome} ({probability})</p>

                  </div>

                </div>

                <div className="bg-white p-4 rounded-lg shadow-inner">

                  <h3 className="font-semibold text-gray-700 mb-2 border-b pb-1">Predictions Breakdown</h3>

                  <ul className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-2 text-sm text-gray-700">

                    {Object.keys(match.predictions).map(site => (

                      <li key={site} className="flex justify-between items-center">

                        <span className="font-medium">{site}:</span>

                        <span className="font-bold">{match.predictions[site]}</span>

                      </li>

                    ))}

                  </ul>

                </div>

              </div>

            );

          })}

        </section>

      </div>

    </div>

  );

};

export default PredictionAnalyzer;