/\* styles.css \*/

/\* General Styles \*/

body {

    color: white;

    background-color: black;

    text-align: center;

  }

  /\* Game Area \*/

  .game-area {

    display: inline-block;

    border: 1px solid red;

    text-align: center;

    padding: 10px;

    margin-top: 100px;

    background-color: rgb(49, 43, 43);

  }

  /\* Buttons \*/

  .move-button {

    height: 100px;

    width: 100px;

    border: 3px solid white;

    border-radius: 50px;

    background-color: transparent;

    margin: 0px 20px;

    cursor: pointer;

  }

  .move-icon {

    height: 70px;

  }

  .small-move-icon {

    height: 30px;

  }

  /\* Autoplay Button \*/

  .autoplay-button {

    background-color: red;

    color: white;

    font-size: 16px;

    border: none;

    border-radius: 2px;

    padding: 10px;

    cursor: pointer;

    font-family: Arial, Helvetica, sans-serif;

  }

  .autoplay-button.green {

    background-color: green;

  }

  /\* Score and Result Text \*/

  .player-move,

  .game-result,

  .scoreboard {

    font-size: 20px;

    font-weight: bold;

  }

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

import './NewList.css'; // Import the CSS file

const NewList = () => {

  // Initialize score state

  const [score, setScore] = useState(

    JSON.parse(localStorage.getItem('myScore')) || {

      win: 0,

      loss: 0,

      tie: 0,

    }

  );

  const [isAutoPlaying, setIsAutoPlaying] = useState(false);

  const [intervalId, setIntervalId] = useState(null);

  const [computerMove, setComputerMove] = useState('');

  const [playerMove, setPlayerMove] = useState('');

  const [result, setResult] = useState(''); // Add a result state

  // Update the score in localStorage whenever it changes

  useEffect(() => {

    localStorage.setItem('myScore', JSON.stringify(score));

  }, [score]);

  // Function to play the game

  const playGame = (playerMove) => {

    const computerMove = generateComputerMove();

    setComputerMove(computerMove); // Store computer's move in state

    setPlayerMove(playerMove);

    let result = '';

    // Determine the result of the game

    switch (playerMove) {

      case 'ROCK':

        switch (computerMove) {

          case 'ROCK':

            result = 'TIE';

            break;

          case 'PAPER':

            result = 'LOSS';

            break;

          case 'SCISSORS':

            result = 'WIN';

            break;

        }

        break;

      case 'PAPER':

        switch (computerMove) {

          case 'ROCK':

            result = 'WIN';

            break;

          case 'PAPER':

            result = 'TIE';

            break;

          case 'SCISSORS':

            result = 'LOSS';

            break;

        }

        break;

      case 'SCISSORS':

        switch (computerMove) {

          case 'ROCK':

            result = 'LOSS';

            break;

          case 'PAPER':

            result = 'WIN';

            break;

          case 'SCISSORS':

            result = 'TIE';

            break;

        }

        break;

      default:

        break;

    }

    setResult(result); // Set the result of the game

    // Update the score based on the result

    setScore((prevScore) => {

      const updatedScore = { ...prevScore };

      switch (result) {

        case 'LOSS':

          updatedScore.loss += 1;

          break;

        case 'WIN':

          updatedScore.win += 1;

          break;

        case 'TIE':

          updatedScore.tie += 1;

          break;

        default:

          break;

      }

      return updatedScore;

    });

  };

  // Generate a random move for the computer

  const generateComputerMove = () => {

    const randomNumber = Math.random();

    if (randomNumber <= 1 / 3) return 'ROCK';

    if (randomNumber > 1 / 3 && randomNumber <= 2 / 3) return 'PAPER';

    return 'SCISSORS';

  };

  // Function to display the scoreboard

  const displayScoreBoard = () => {

    return `Scoreboard: WINS ${score.win} LOSS ${score.loss} TIE ${score.tie}`;

  };

  // Clear the score from localStorage and reset the state

  const clearScore = () => {

    localStorage.removeItem('myScore');

    setScore({

      win: 0,

      loss: 0,

      tie: 0,

    });

  };

  // Handle autoplay mode

  const playAutoMode = () => {

    if (!isAutoPlaying) {

      const id = setInterval(() => {

        const computerGeneratedPlayerMove = generateComputerMove();

        playGame(computerGeneratedPlayerMove);

      }, 1000);

      setIntervalId(id);

      setIsAutoPlaying(true);

    } else {

      clearInterval(intervalId);

      setIsAutoPlaying(false);

    }

  };

  return (

    <div>

      <div className="game-area">

        <button

          onClick={() => playGame('ROCK')}

          className="move-button"

        >

          <img src="/images/rock-emoji.png" className="move-icon" />

        </button>

        <button

          onClick={() => playGame('PAPER')}

          className="move-button"

        >

          <img src="/images/paper-emoji.png" className="move-icon" />

        </button>

        <button

          onClick={() => playGame('SCISSORS')}

          className="move-button"

        >

          <img src="/images/scissors-emoji.png" className="move-icon" />

        </button>

        <button

          onClick={clearScore}

          className="move-button"

        >

          <img src="/images/refresh-emoji.png" className="move-icon" />

        </button>

        <p className="player-move">

          Computer move

          <img src={`/images/${computerMove.toLowerCase()}-emoji.png`} className="small-move-icon" alt={`${computerMove}-emoji`} />

          <img src={`/images/${playerMove.toLowerCase()}-emoji.png`} className="small-move-icon" alt={`${playerMove}-emoji`} />

          {' '} Player move

        </p>

        <p className="game-result">

          Result: {result} {/\* Display the dynamic result \*/}

        </p>

        <p className="scoreboard">

          {displayScoreBoard()}

        </p>

        <button

          onClick={playAutoMode}

          className={`autoplay-button ${isAutoPlaying ? 'green' : ''}`}

        >

          {isAutoPlaying ? 'STOP AUTOPLAY' : 'AUTOPLAY'}

        </button>

      </div>

    </div>

  );

};

export default NewList;