!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Verb Detective: The Missing Tense Case</title>

    <!-- Tailwind CSS CDN -->

    <script src="https://cdn.tailwindcss.com"></script>

    <style>

        /\* Custom styles for a detective theme \*/

        body {

            font-family: 'Inter', sans-serif;

            background-color: #1a202c; /\* Dark background for a mysterious feel \*/

            color: #e2e8f0; /\* Light text \*/

            display: flex;

            justify-content: center;

            align-items: center;

            min-height: 100vh;

            padding: 1rem;

        }

        .game-container {

            background-color: #2d3748; /\* Slightly lighter dark background for the container \*/

            border-radius: 1.5rem; /\* More rounded corners \*/

            box-shadow: 0 10px 15px rgba(0, 0, 0, 0.5); /\* Deeper shadow \*/

            max-width: 600px;

            width: 100%;

            padding: 2rem;

            text-align: center;

            border: 2px solid #4a5568; /\* Subtle border \*/

        }

        .question-box {

            background-color: #4a5568; /\* Darker box for the question \*/

            border-radius: 0.75rem;

            padding: 1.5rem;

            margin-bottom: 1.5rem;

            font-size: 1.25rem;

            font-weight: bold;

            color: #cbd5e0;

            min-height: 80px; /\* Ensure consistent height \*/

            display: flex;

            align-items: center;

            justify-content: center;

            text-align: center;

        }

        .option-button {

            background-color: #4299e1; /\* Blue button \*/

            color: white;

            padding: 0.75rem 1.5rem;

            border-radius: 0.75rem;

            font-weight: bold;

            transition: background-color 0.3s ease, transform 0.1s ease;

            cursor: pointer;

            width: 100%;

            margin-bottom: 0.75rem;

            border: none;

            outline: none;

        }

        .option-button:hover {

            background-color: #3182ce; /\* Darker blue on hover \*/

            transform: translateY(-2px); /\* Slight lift effect \*/

        }

        .option-button:active {

            transform: translateY(0);

        }

        .feedback {

            font-size: 1.1rem;

            font-weight: bold;

            margin-top: 1rem;

            min-height: 24px; /\* Space for feedback \*/

        }

        .score-display {

            font-size: 1.2rem;

            font-weight: bold;

            margin-top: 1rem;

            color: #a0aec0;

        }

        .start-button, .next-button {

            background-color: #38a169; /\* Green for start/next \*/

            color: white;

            padding: 0.75rem 2rem;

            border-radius: 0.75rem;

            font-weight: bold;

            transition: background-color 0.3s ease, transform 0.1s ease;

            cursor: pointer;

            margin-top: 1.5rem;

            border: none;

            outline: none;

        }

        .start-button:hover, .next-button:hover {

            background-color: #2f855a;

            transform: translateY(-2px);

        }

        .start-button:active, .next-button:active {

            transform: translateY(0);

        }

        .disabled-button {

            background-color: #6b7280; /\* Gray for disabled buttons \*/

            cursor: not-allowed;

        }

        .correct-answer {

            background-color: #48bb78 !important; /\* Green for correct \*/

        }

        .wrong-answer {

            background-color: #e53e3e !important; /\* Red for wrong \*/

        }

    </style>

</head>

<body>

    <div class="game-container">

        <h1 class="text-3xl font-extrabold mb-6 text-yellow-300">🕵️‍♀️ Verb Detective: The Missing Tense Case 🕵️‍♂️</h1>

        <p class="text-lg mb-4 text-gray-300">Help Detective Tim Time solve the case of the missing verbs!</p>

        <div id="gameScreen" class="hidden">

            <div id="questionBox" class="question-box"></div>

            <div id="optionsContainer" class="grid grid-cols-1 md:grid-cols-2 gap-4">

                <!-- Options will be dynamically loaded here -->

            </div>

            <div id="feedback" class="feedback"></div>

            <div id="scoreDisplay" class="score-display">Score: 0</div>

            <button id="nextButton" class="next-button hidden">Next Case</button>

        </div>

        <button id="startButton" class="start-button">Start Investigation</button>

    </div>

    <script>

        // Array of questions for the game

        // Each question object contains:

        // - sentence: The sentence with a blank to fill.

        // - correctAnswer: The correct verb form.

        // - options: An array of possible choices, including the correct one.

        // - tense: (Optional) The tense being tested, for internal reference.

        const questions = [

            {

                sentence: "Maggie \_\_\_ a curious girl.",

                correctAnswer: "is",

                options: ["is", "are", "am", "be"],

                tense: "Simple Present (to be)"

            },

            {

                sentence: "Yesterday, Tim Time \_\_\_ a new clue.",

                correctAnswer: "found",

                options: ["find", "finds", "found", "finding"],

                tense: "Simple Past"

            },

            {

                sentence: "Dr. Feathers always \_\_\_ wise advice.",

                correctAnswer: "gives",

                options: ["give", "gives", "given", "giving"],

                tense: "Simple Present"

            },

            {

                sentence: "Tomorrow, they \_\_\_ the mystery.",

                correctAnswer: "will solve",

                options: ["solve", "solved", "will solve", "solving"],

                tense: "Simple Future"

            },

            {

                sentence: "The students \_\_\_ hard on their grammar.",

                correctAnswer: "are working",

                options: ["is working", "are working", "works", "worked"],

                tense: "Present Continuous"

            },

            {

                sentence: "Last week, the verbs \_\_\_ missing.",

                correctAnswer: "went",

                options: ["go", "goes", "gone", "went"],

                tense: "Simple Past"

            },

            {

                sentence: "By next year, Maggie \_\_\_ all grammar puzzles.",

                correctAnswer: "will have solved",

                options: ["solves", "solved", "will solve", "will have solved"],

                tense: "Future Perfect"

            },

            {

                sentence: "She \_\_\_ her homework when the power went out.",

                correctAnswer: "was doing",

                options: ["is doing", "was doing", "does", "did"],

                tense: "Past Continuous"

            },

            {

                sentence: "They \_\_\_ English for two hours now.",

                correctAnswer: "have been studying",

                options: ["study", "are studying", "have studied", "have been studying"],

                tense: "Present Perfect Continuous"

            },

            {

                sentence: "He \_\_\_ to the library before he came here.",

                correctAnswer: "had gone",

                options: ["goes", "went", "has gone", "had gone"],

                tense: "Past Perfect"

            }

        ];

        // Get references to DOM elements

        const startButton = document.getElementById('startButton');

        const gameScreen = document.getElementById('gameScreen');

        const questionBox = document.getElementById('questionBox');

        const optionsContainer = document.getElementById('optionsContainer');

        const feedbackDisplay = document.getElementById('feedback');

        const scoreDisplay = document.getElementById('scoreDisplay');

        const nextButton = document.getElementById('nextButton');

        let currentQuestionIndex = 0; // Tracks the current question

        let score = 0; // Tracks the player's score

        let questionAnswered = false; // Flag to prevent multiple answers per question

        /\*\*

         \* Shuffles an array in place using the Fisher-Yates (Knuth) algorithm.

         \* @param {Array} array The array to shuffle.

         \*/

        function shuffleArray(array) {

            for (let i = array.length - 1; i > 0; i--) {

                const j = Math.floor(Math.random() \* (i + 1));

                [array[i], array[j]] = [array[j], array[i]]; // Swap elements

            }

        }

        /\*\*

         \* Loads the current question into the game interface.

         \*/

        function loadQuestion() {

            questionAnswered = false; // Reset flag for new question

            feedbackDisplay.textContent = ''; // Clear previous feedback

            nextButton.classList.add('hidden'); // Hide next button

            // Enable all option buttons and remove styling from previous question

            Array.from(optionsContainer.children).forEach(button => {

                button.classList.remove('correct-answer', 'wrong-answer', 'disabled-button');

                button.disabled = false;

            });

            // If all questions have been answered, show final score

            if (currentQuestionIndex >= questions.length) {

                questionBox.textContent = `Investigation Complete! You solved ${score} out of ${questions.length} cases!`;

                optionsContainer.innerHTML = ''; // Clear options

                nextButton.classList.add('hidden');

                startButton.textContent = 'Restart Investigation';

                startButton.classList.remove('hidden'); // Show restart button

                return;

            }

            const currentQuestion = questions[currentQuestionIndex];

            questionBox.textContent = currentQuestion.sentence.replace('\_\_\_', '\_\_\_\_\_\_'); // Show blank

            optionsContainer.innerHTML = ''; // Clear previous options

            // Create buttons for each option

            currentQuestion.options.forEach(option => {

                const button = document.createElement('button');

                button.textContent = option;

                button.classList.add('option-button');

                button.addEventListener('click', () => checkAnswer(option, button));

                optionsContainer.appendChild(button);

            });

        }

        /\*\*

         \* Checks if the selected answer is correct and provides feedback.

         \* @param {string} selectedOption The option selected by the user.

         \* @param {HTMLElement} clickedButton The button element that was clicked.

         \*/

        function checkAnswer(selectedOption, clickedButton) {

            if (questionAnswered) return; // Prevent answering multiple times

            questionAnswered = true;

            const currentQuestion = questions[currentQuestionIndex];

            // Disable all option buttons after an answer is selected

            Array.from(optionsContainer.children).forEach(button => {

                button.disabled = true;

                button.classList.add('disabled-button'); // Visually disable

            });

            if (selectedOption === currentQuestion.correctAnswer) {

                score++;

                feedbackDisplay.textContent = 'Correct! Excellent detective work!';

                feedbackDisplay.style.color = '#48bb78'; // Green text

                clickedButton.classList.remove('disabled-button'); // Remove disabled style

                clickedButton.classList.add('correct-answer'); // Highlight correct

            } else {

                feedbackDisplay.textContent = `Incorrect. The correct answer was "${currentQuestion.correctAnswer}".`;

                feedbackDisplay.style.color = '#e53e3e'; // Red text

                clickedButton.classList.remove('disabled-button'); // Remove disabled style

                clickedButton.classList.add('wrong-answer'); // Highlight wrong

                // Also highlight the correct answer

                Array.from(optionsContainer.children).forEach(button => {

                    if (button.textContent === currentQuestion.correctAnswer) {

                        button.classList.remove('disabled-button');

                        button.classList.add('correct-answer');

                    }

                });

            }

            scoreDisplay.textContent = `Score: ${score}`;

            nextButton.classList.remove('hidden'); // Show next button

        }

        /\*\*

         \* Starts the game or restarts it if it's already finished.

         \*/

        function startGame() {

            shuffleArray(questions); // Randomize question order

            currentQuestionIndex = 0;

            score = 0;

            scoreDisplay.textContent = `Score: ${score}`;

            startButton.classList.add('hidden'); // Hide start button

            gameScreen.classList.remove('hidden'); // Show game screen

            loadQuestion();

        }

        /\*\*

         \* Moves to the next question.

         \*/

        function nextQuestion() {

            currentQuestionIndex++;

            loadQuestion();

        }

        // Event listeners

        startButton.addEventListener('click', startGame);

        nextButton.addEventListener('click', nextQuestion);

        // Initial setup on page load (optional, can be done by startGame directly)

        // loadQuestion(); // Don't load until start button is clicked

    </script>

</body>

</html>