<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>People Team Toolbox Challenge</title>

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

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;500;600;700&display=swap" rel="stylesheet">

<!-- Chosen Palette: Warm Neutrals (Oat, Stone, Terracotta Accent) -->

<!-- Application Structure Plan: A three-section, tab-navigated SPA ('My Tool', 'Game Guide', 'Toolbox Log') was chosen to mirror a user's task flow during the event: 1. See their assigned identity, 2. Learn the rules and access clues, 3. Track their progress. This task-oriented structure is more intuitive and functional for a live game than a simple digital replication of the paper card. The interactive log, featuring a dynamic progress chart, gamifies the experience and provides clear, immediate feedback on their collection progress. -->

<!-- Visualization & Content Choices:

1. Report Info: Game progress (collecting 6 tools of 3 types). Goal: Inform/motivate user. Viz/Method: Doughnut Chart (Chart.js/Canvas). Interaction: Chart dynamically updates as user logs interactions via a modal form. Justification: A chart provides a more engaging, at-a-glance understanding of completion status than a list, enhancing the game feel. Library: Chart.js.

2. Report Info: Guided Clue Questions. Goal: Organize reference material. Viz/Method: Interactive Accordion (HTML/Tailwind/JS). Interaction: User clicks a tool name to expand/collapse and reveal its question. Justification: An accordion reduces cognitive load by hiding non-essential information, allowing users to quickly find the specific clue they need during a conversation. Library/Method: Vanilla JS. -->

<!-- CONFIRMATION: NO SVG graphics used. NO Mermaid JS used. -->

<style>

body {

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

background-color: #F5F5F0; /\* Oat \*/

color: #44403C; /\* Stone-800 \*/

}

.nav-active {

background-color: #E4775B; /\* Terracotta \*/

color: #FFFFFF;

}

.nav-inactive {

background-color: #FFFFFF;

color: #57534E; /\* Stone-600 \*/

}

.accordion-content {

max-height: 0;

overflow: hidden;

transition: max-height 0.3s ease-in-out;

}

.accordion-content.open {

max-height: 200px;

}

.chart-container {

position: relative;

width: 100%;

max-width: 320px;

margin-left: auto;

margin-right: auto;

height: 320px;

max-height: 80vw;

}

</style>

</head>

<body class="antialiased">

<div id="app" class="container mx-auto p-4 max-w-2xl">

<header class="text-center mb-6">

<h1 class="text-3xl font-bold text-stone-800">People Team</h1>

<h2 class="text-4xl font-bold text-transparent bg-clip-text bg-gradient-to-r from-orange-600 to-amber-700">Toolbox Challenge</h2>

</header>

<nav class="flex justify-center gap-2 mb-6 shadow-sm rounded-full p-1 bg-stone-200">

<button onclick="handleNavClick('myToolSection')" id="navMyTool" class="nav-active flex-1 py-2 px-4 rounded-full font-semibold transition-colors duration-300">My Tool</button>

<button onclick="handleNavClick('gameGuideSection')" id="navGameGuide" class="nav-inactive flex-1 py-2 px-4 rounded-full font-semibold transition-colors duration-300">Game Guide</button>

<button onclick="handleNavClick('toolboxLogSection')" id="navToolboxLog" class="nav-inactive flex-1 py-2 px-4 rounded-full font-semibold transition-colors duration-300">Toolbox Log</button>

</nav>

<main>

<!-- Section 1: My Tool -->

<section id="myToolSection">

<div class="bg-white p-6 rounded-2xl shadow-lg border border-stone-200 text-center">

<p class="text-stone-500 font-medium">Your primary tool is...</p>

<h3 id="assignedToolName" class="text-4xl font-bold my-3 text-stone-800"></h3>

<div id="assignedToolSymbol" class="text-8xl my-4"></div>

<div class="bg-stone-100 p-3 rounded-lg">

<p class="font-semibold text-stone-700">Your Core Skill:</p>

<p id="assignedToolSkill" class="text-stone-600"></p>

</div>

<p class="text-xs text-stone-400 mt-6 italic">Don't reveal your tool! Let others deduce it based on your answers.</p>

</div>

</section>

<!-- Section 2: Game Guide -->

<section id="gameGuideSection" class="hidden">

<div class="bg-white p-6 rounded-2xl shadow-lg border border-stone-200">

<h3 class="text-2xl font-bold text-center mb-4">How to Play</h3>

<div class="text-stone-700 space-y-3">

<p><strong class="text-orange-600">GOAL:</strong> Your mission is to complete your toolbox by finding colleagues with different tools. You need to successfully deduce and log \*\*two of each\*\* of the three tool types you're missing (6 total).</p>

<p><strong class="text-orange-600">HOW:</strong> Approach a colleague and ask one of the "Guided Clue Questions" below. Based on their answer, deduce their tool type. Record your interaction in the "Toolbox Log".</p>

</div>

<h4 class="text-xl font-bold text-center mt-8 mb-4">Guided Clue Questions</h4>

<div id="accordionContainer" class="space-y-3"></div>

</div>

</section>

<!-- Section 3: Toolbox Log -->

<section id="toolboxLogSection" class="hidden">

<div class="bg-white p-6 rounded-2xl shadow-lg border border-stone-200">

<h3 class="text-2xl font-bold text-center mb-2">My Toolbox Progress</h3>

<p id="progressText" class="text-center text-stone-500 mb-4">Collect 6 tools to win!</p>

<div class="chart-container mb-6">

<canvas id="progressChart"></canvas>

</div>

<div id="logEntries" class="space-y-3"></div>

<button onclick="openLogModal()" class="w-full mt-6 bg-orange-600 text-white font-bold py-3 px-4 rounded-xl hover:bg-orange-700 transition-colors duration-300 shadow-md">

+ Add Interaction

</button>

</div>

</section>

</main>

</div>

<!-- Modal for adding interactions -->

<div id="logModal" class="fixed inset-0 bg-black bg-opacity-60 flex items-center justify-center p-4 hidden z-50">

<div class="bg-white rounded-2xl p-6 w-full max-w-md shadow-2xl transform transition-all scale-95 opacity-0" id="modalContent">

<h4 class="text-2xl font-bold mb-4 text-stone-800">Log an Interaction</h4>

<div class="space-y-4">

<div>

<label for="colleagueName" class="font-semibold text-stone-600 block mb-1">Colleague's Name:</label>

<input type="text" id="colleagueName" class="w-full border-stone-300 border rounded-lg p-2 focus:ring-2 focus:ring-orange-500 focus:border-orange-500 outline-none">

</div>

<div>

<label for="deduction" class="font-semibold text-stone-600 block mb-1">My Deduction (Your Guess):</label>

<select id="deduction" class="w-full border-stone-300 border rounded-lg p-2 bg-white focus:ring-2 focus:ring-orange-500 focus:border-orange-500 outline-none">

<option value="">Select a tool...</option>

</select>

</div>

<div>

<label for="notes" class="font-semibold text-stone-600 block mb-1">Clue/Answer Notes:</label>

<textarea id="notes" rows="3" class="w-full border-stone-300 border rounded-lg p-2 focus:ring-2 focus:ring-orange-500 focus:border-orange-500 outline-none"></textarea>

</div>

</div>

<div class="flex gap-3 mt-6">

<button onclick="closeLogModal()" class="flex-1 bg-stone-200 text-stone-700 font-bold py-2 px-4 rounded-xl hover:bg-stone-300 transition-colors">Cancel</button>

<button onclick="saveInteraction()" class="flex-1 bg-orange-600 text-white font-bold py-2 px-4 rounded-xl hover:bg-orange-700 transition-colors">Save Log</button>

</div>

</div>

</div>

<script>

const state = {

currentSection: 'myToolSection',

myAssignedTool: null,

toolsToCollect: [],

loggedInteractions: [],

};

const allTools = {

Hammer: {

skill: 'Strategic Problem-Solving',

symbol: '🔨',

question: "If you had to draw a blueprint for a complex People Team initiative, what is the first thing you'd put on the paper?",

clue: "Look for answers about planning, fixing, or building a clear path."

},

'Tape Measure': {

skill: 'Empathy & Communication',

symbol: '📏',

question: "When listening to conflicting sides of a story, what do you primarily try to measure or understand?",

clue: "Look for answers about alignment, rapport, or gauging different perspectives."

},

Wrench: {

skill: 'Compliance & Implementation',

symbol: '🔧',

question: "What is your main focus when auditing a process to make sure it's running smoothly and correctly?",

clue: "Look for answers about precision, structure, or tightening up details."

},

'Multi-Tool': {

skill: 'Adaptability & Innovation',

symbol: '🛠️',

question: "If a sudden, unexpected challenge comes up, how do you decide which skill to use first to tackle it?",

clue: "Look for answers about versatility, quick pivoting, or creative solutions."

}

};

let progressChart = null;

function init() {

const toolNames = Object.keys(allTools);

state.myAssignedTool = toolNames[Math.floor(Math.random() \* toolNames.length)];

state.toolsToCollect = toolNames.filter(t => t !== state.myAssignedTool);

document.getElementById('assignedToolName').textContent = state.myAssignedTool;

document.getElementById('assignedToolSymbol').textContent = allTools[state.myAssignedTool].symbol;

document.getElementById('assignedToolSkill').textContent = allTools[state.myAssignedTool].skill;

renderAccordion();

populateDeductionDropdown();

initializeChart();

handleNavClick('myToolSection');

}

function handleNavClick(sectionId) {

state.currentSection = sectionId;

['myToolSection', 'gameGuideSection', 'toolboxLogSection'].forEach(id => {

document.getElementById(id).classList.add('hidden');

});

document.getElementById(sectionId).classList.remove('hidden');

['navMyTool', 'navGameGuide', 'navToolboxLog'].forEach(navId => {

document.getElementById(navId).classList.remove('nav-active', 'nav-inactive');

document.getElementById(navId).classList.add(navId.toLowerCase().includes(sectionId.toLowerCase()) ? 'nav-active' : 'nav-inactive');

});

}

function renderAccordion() {

const container = document.getElementById('accordionContainer');

container.innerHTML = '';

Object.keys(allTools).forEach(toolName => {

const tool = allTools[toolName];

const accordionItem = document.createElement('div');

accordionItem.className = 'border border-stone-200 rounded-lg';

accordionItem.innerHTML = `

<button onclick="toggleAccordion(this)" class="w-full flex justify-between items-center p-4 text-left font-semibold text-stone-700">

<span>${tool.symbol} ${toolName}</span>

<span class="transform transition-transform duration-300">▼</span>

</button>

<div class="accordion-content px-4 pb-4">

<p class="font-medium text-stone-800 mb-2">${tool.question}</p>

<p class="text-sm text-stone-500 italic">${tool.clue}</p>

</div>

`;

container.appendChild(accordionItem);

});

}

function toggleAccordion(buttonElement) {

const content = buttonElement.nextElementSibling;

const icon = buttonElement.querySelector('span:last-child');

if (content.classList.contains('open')) {

content.classList.remove('open');

icon.style.transform = 'rotate(0deg)';

} else {

document.querySelectorAll('.accordion-content.open').forEach(openContent => {

openContent.classList.remove('open');

openContent.previousElementSibling.querySelector('span:last-child').style.transform = 'rotate(0deg)';

});

content.classList.add('open');

icon.style.transform = 'rotate(180deg)';

}

}

function populateDeductionDropdown() {

const select = document.getElementById('deduction');

select.innerHTML = '<option value="">Select a tool...</option>';

state.toolsToCollect.forEach(toolName => {

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

option.value = toolName;

option.textContent = toolName;

select.appendChild(option);

});

}

function openLogModal() {

const modal = document.getElementById('logModal');

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

modal.classList.remove('hidden');

setTimeout(() => {

modalContent.classList.remove('scale-95', 'opacity-0');

modalContent.classList.add('scale-100', 'opacity-100');

}, 10);

}

function closeLogModal() {

const modal = document.getElementById('logModal');

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

modalContent.classList.remove('scale-100', 'opacity-100');

modalContent.classList.add('scale-95', 'opacity-0');

setTimeout(() => {

modal.classList.add('hidden');

document.getElementById('colleagueName').value = '';

document.getElementById('deduction').value = '';

document.getElementById('notes').value = '';

}, 200);

}

function saveInteraction() {

const name = document.getElementById('colleagueName').value;

const deduction = document.getElementById('deduction').value;

const notes = document.getElementById('notes').value;

if (!name || !deduction) {

alert('Please enter the colleague\'s name and your deduction.');

return;

}

const counts = state.loggedInteractions.reduce((acc, entry) => {

acc[entry.deduction] = (acc[entry.deduction] || 0) + 1;

return acc;

}, {});

if ((counts[deduction] || 0) >= 2) {

alert(`You have already collected two ${deduction} tools!`);

return;

}

state.loggedInteractions.push({ name, deduction, notes });

renderLogEntries();

updateProgressChart();

closeLogModal();

}

function renderLogEntries() {

const container = document.getElementById('logEntries');

container.innerHTML = '';

if(state.loggedInteractions.length === 0) {

container.innerHTML = '<p class="text-center text-stone-500 py-4">No interactions logged yet. Start mingling!</p>';

} else {

state.loggedInteractions.forEach(entry => {

const tool = allTools[entry.deduction];

const entryDiv = document.createElement('div');

entryDiv.className = 'bg-stone-100 p-4 rounded-lg border border-stone-200';

entryDiv.innerHTML = `

<div class="flex items-center justify-between">

<div class="flex items-center gap-3">

<span class="text-3xl">${tool.symbol}</span>

<div>

<p class="font-bold text-stone-800">${entry.name}</p>

<p class="text-sm text-stone-600">Deduction: ${entry.deduction}</p>

</div>

</div>

</div>

${entry.notes ? `<p class="text-sm text-stone-500 mt-2 pt-2 border-t border-stone-200">Notes: ${entry.notes}</p>` : ''}

`;

container.appendChild(entryDiv);

});

}

}

function initializeChart() {

const ctx = document.getElementById('progressChart').getContext('2d');

const counts = state.toolsToCollect.reduce((acc, tool) => {

acc[tool] = 0;

return acc;

}, {});

progressChart = new Chart(ctx, {

type: 'doughnut',

data: {

labels: state.toolsToCollect,

datasets: [{

label: 'Tools Collected',

data: state.toolsToCollect.map(tool => counts[tool] || 0),

backgroundColor: ['#F87171', '#60A5FA', '#FBBF24'],

borderColor: '#FFFFFF',

borderWidth: 4,

hoverOffset: 8

}]

},

options: {

responsive: true,

maintainAspectRatio: false,

cutout: '70%',

plugins: {

legend: {

position: 'bottom',

labels: {

font: {

size: 14,

family: 'Inter'

},

boxWidth: 20,

padding: 20,

}

},

tooltip: {

callbacks: {

label: function(context) {

let label = context.label || '';

if (label) {

label += ': ';

}

if (context.parsed !== null) {

label += `${context.parsed} / 2`;

}

return label;

}

}

}

}

}

});

updateProgressChart();

}

function updateProgressChart() {

const counts = state.toolsToCollect.reduce((acc, tool) => {

acc[tool] = 0;

return acc;

}, {});

state.loggedInteractions.forEach(entry => {

if(counts.hasOwnProperty(entry.deduction)) {

counts[entry.deduction]++;

}

});

const collectedCount = state.loggedInteractions.length;

document.getElementById('progressText').textContent = `Collected ${collectedCount} of 6 tools. Keep going!`;

if (collectedCount >= 6) {

document.getElementById('progressText').textContent = 'Congratulations! You completed your toolbox!';

}

if(progressChart) {

progressChart.data.datasets[0].data = state.toolsToCollect.map(tool => counts[tool]);

progressChart.update();

}

}

document.addEventListener('DOMContentLoaded', init);

</script>

</body>

</html>