<!DOCTYPE html>

<html lang="pt-BR">

<head>

<meta charset="UTF-8">

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

<title>Jornada da Nutrição: Autoavaliação com IA</title>

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

<script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/3.7.0/chart.min.js"></script>

<link rel="preconnect" href="https://fonts.googleapis.com">

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

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

<style>

body {

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

}

/\* Style for number input arrows \*/

input[type=number]::-webkit-inner-spin-button,

input[type=number]::-webkit-outer-spin-button {

-webkit-appearance: none;

margin: 0;

}

input[type=number] {

-moz-appearance: textfield;

}

.prose h3 {

font-size: 1.25rem;

font-weight: 600;

margin-top: 1.5em;

margin-bottom: 0.5em;

}

.prose h4 {

font-size: 1.1rem;

font-weight: 600;

margin-top: 1.2em;

margin-bottom: 0.4em;

}

.prose blockquote {

border-left: 4px solid #d1d5db;

padding-left: 1rem;

font-style: italic;

color: #4b5563;

}

.key-input {

width: 4.5rem; /\* Adjusted for better spacing on mobile \*/

text-align: center;

font-size: 1.125rem;

text-transform: uppercase;

border-radius: 0.375rem;

border: 1px solid #d1d5db;

padding: 0.5rem;

}

@media (min-width: 640px) {

.key-input {

width: 5rem;

}

}

</style>

</head>

<body class="bg-gray-50 text-gray-800">

<div id="app" class="max-w-4xl mx-auto p-4 md:p-6">

<!-- Loading Indicator -->

<div id="loading" class="fixed inset-0 bg-white bg-opacity-75 flex flex-col items-center justify-center z-50">

<svg class="animate-spin h-10 w-10 text-blue-600" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24">

<circle class="opacity-25" cx="12" cy="12" r="10" stroke="currentColor" stroke-width="4"></circle>

<path class="opacity-75" fill="currentColor" d="M4 12a8 8 0 018-8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962 0 014 12H0c0 3.042 1.135 5.824 3 7.938l3-2.647z"></path>

</svg>

<p class="mt-4 text-lg font-medium text-gray-700">A iniciar aplicação...</p>

</div>

<!-- Login Screen -->

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

<div class="bg-white p-6 sm:p-8 rounded-xl shadow-lg max-w-lg mx-auto mt-10">

<h2 class="text-2xl font-bold text-center text-gray-800 mb-2">Chave de Acesso</h2>

<p class="text-center text-gray-500 mb-6">Insira a sua chave de 16 caracteres para aceder.</p>

<div id="key-input-container" class="flex justify-center flex-wrap sm:flex-nowrap gap-2 mb-4">

<input type="text" maxlength="4" class="key-input" data-index="0">

<span class="self-center text-gray-400 hidden sm:inline">-</span>

<input type="text" maxlength="4" class="key-input" data-index="1">

<span class="self-center text-gray-400 hidden sm:inline">-</span>

<input type="text" maxlength="4" class="key-input" data-index="2">

<span class="self-center text-gray-400 hidden sm:inline">-</span>

<input type="text" maxlength="4" class="key-input" data-index="3">

</div>

<div id="login-error" class="text-red-600 text-center mb-4 h-5"></div>

<button onclick="loginWithKey()" class="w-full bg-blue-600 text-white font-bold py-3 px-4 rounded-lg shadow-md hover:bg-blue-700 transition duration-300">

Validar Chave

</button>

<div class="mt-6 pt-4 border-t text-center text-sm text-gray-600">

<p class="font-semibold">Como obter uma chave de acesso?</p>

<p>Para adquirir uma licença com validade de 1 semana, 1 mês ou 3 meses, por favor entre em contato pelo e-mail:</p>

<a href="mailto:olairjr@gmail.com" class="text-blue-600 font-medium hover:underline">olairjr@gmail.com</a>

</div>

</div>

</div>

<!-- Main App Content (hidden by default) -->

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

<!-- Header -->

<header class="mb-6 flex justify-between items-center">

<h1 class="text-2xl sm:text-3xl font-bold text-blue-700">Jornada da Nutrição</h1>

<div class="flex items-center gap-2 sm:gap-4">

<div id="user-info" class="text-right text-xs text-gray-500">

<p>ID do Utilizador:</p>

<p id="userIdDisplay" class="font-mono text-xxs sm:text-xs truncate"></p>

</div>

<button onclick="logout()" class="bg-red-500 text-white font-semibold py-2 px-3 rounded-lg shadow hover:bg-red-600 transition text-xs">Sair</button>

</div>

</header>

<!-- Welcome Screen -->

<div id="welcomeScreen">

<div class="bg-white p-6 rounded-lg shadow-md">

<h2 class="text-2xl font-semibold mb-4 text-gray-900">Bem-vindo(a)!</h2>

<p class="mb-6 text-gray-600">Este é o seu espaço para refletir e celebrar cada passo em direção a uma vida mais saudável.</p>

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

<button onclick="startAssessment()" class="w-full bg-blue-600 text-white font-semibold py-3 px-4 rounded-lg shadow hover:bg-blue-700 transition">Nova Avaliação</button>

<button onclick="showScreen('progressScreen')" class="w-full bg-green-600 text-white font-semibold py-3 px-4 rounded-lg shadow hover:bg-green-700 transition">Meu Progresso</button>

<button onclick="showAiPlanner()" class="w-full bg-purple-600 text-white font-semibold py-3 px-4 rounded-lg shadow hover:bg-purple-700 transition flex items-center justify-center gap-2">✨ Plano com IA</button>

<button onclick="showScreen('knowledgeScreen')" class="w-full bg-gray-600 text-white font-semibold py-3 px-4 rounded-lg shadow hover:bg-gray-700 transition">Conhecimento</button>

</div>

</div>

</div>

<!-- App Screens -->

<div id="assessmentScreen" class="hidden"></div>

<div id="progressScreen" class="hidden"></div>

<div id="aiPlanScreen" class="hidden"></div>

<div id="knowledgeScreen" class="hidden"></div>

</div>

<!-- Modal for confirmation/information -->

<div id="modal" class="fixed inset-0 bg-gray-900 bg-opacity-50 flex items-center justify-center hidden z-50">

<div class="bg-white rounded-lg shadow-xl p-6 w-full max-w-sm mx-4">

<h3 class="text-lg font-medium text-gray-900" id="modalTitle"></h3>

<p class="mt-2 text-sm text-gray-600" id="modalMessage"></p>

<div class="mt-4 flex justify-end space-x-3" id="modalButtons"></div>

</div>

</div>

</div>

<script type="module">

// Firebase Imports

import { initializeApp } from "https://www.gstatic.com/firebasejs/11.6.1/firebase-app.js";

import { getAuth, signInAnonymously, onAuthStateChanged, signInWithCustomToken } from "https://www.gstatic.com/firebasejs/11.6.1/firebase-auth.js";

import { getFirestore, doc, getDoc, updateDoc, onSnapshot, collection, addDoc, query, getDocs, Timestamp } from "https://www.gstatic.com/firebasejs/11.6.1/firebase-firestore.js";

// --- App Configuration & State ---

let db, auth, userId, assessments = [];

let licenseListener = null;

let overallProgressChartInstance = null;

let radarChartInstance = null;

const appId = typeof \_\_app\_id !== 'undefined' ? \_\_app\_id : 'default-app-id';

const firebaseConfig = typeof \_\_firebase\_config !== 'undefined' ? JSON.parse(\_\_firebase\_config) : { apiKey: "YOUR\_API\_KEY", authDomain: "...", projectId: "..." };

// --- Questions data ---

const questions = [

{ category: "Nutrição e Consciência Alimentar", statements: [ "A alimentação é praticada com calma, atenção plena e em ambiente tranquilo.", "A mastigação dos alimentos é completa, descansando os talheres entre garfadas.", "Reconhece-se a diferença entre fome física, sede e vontade emocional.", "As refeições principais respeitam a janela de 20 minutos para o sinal de saciedade.", "A refeição termina com uma leve sensação de que se poderia comer mais.", "Evita-se o consumo de líquidos durante as refeições (30 min antes, 60 min depois).", "Aplica-se o jejum intermitente de forma regular (ex: 14 horas de jejum)." ] },

{ category: "Qualidade e Escolha dos Alimentos", statements: [ "A base da alimentação é composta por alimentos crus, frescos e da época.", "Consomem-se diariamente frutas de cores variadas, de preferência antes das refeições.", "A escolha favorece alimentos mais nutritivos em vez de apenas saborosos.", "Evitam-se alimentos ultraprocessados (embutidos, conservas, sucos artificiais).", "Prefere-se o uso de temperos naturais em vez de sal e condimentos industrializados.", "Reduz-se o consumo de farinha de trigo refinada e açúcar refinado.", "Evita-se o consumo de leite bovino e derivados, especialmente os de longa vida.", "Realizam-se lanches saudáveis entre as refeições, como barras de proteína ou cereais." ] },

{ category: "Saúde Corporal e Estilo de Vida", statements: [ "O sono restaurador e regular é uma prioridade na rotina diária.", "Mantém-se a prática de exercício físico moderado e consistente.", "Praticam-se banhos de sol diários de forma segura.", "A hidratação com água pura e filtrada é constante ao longo do dia.", "Mantém-se a higiene respiratória, como a limpeza das narinas.", "Evitam-se alimentos muito gordurosos e frituras.", "A prevenção de verminoses é praticada com hábitos de higiene pessoal e alimentar." ] },

{ category: "Saúde Mental e Emocional", statements: [ "Praticam-se pausas conscientes ou meditação para equilíbrio mental.", "Adota-se um diálogo interno positivo, construtivo e encorajador.", "A mente é mantida ativa com leitura, estudo e aprendizado contínuo.", "As emoções são observadas com autocompaixão, sem culpa ou castigo.", "Busca-se compreender os gatilhos emocionais que afetam as escolhas diárias.", "Pratica-se o otimismo deliberado para potencializar a resiliência." ] },

{ category: "Consciência e Relações", statements: [ "Agradece-se pelo alimento e pelas pessoas envolvidas na sua produção.", "Pratica-se a escuta ativa, a cordialidade e o elogio sincero nas interações.", "A espiritualidade é vivenciada, seja em comunidade ou em práticas individuais.", "Pratica-se o perdão e a compaixão nas relações.", "Existe a consciência da interconexão entre as próprias ações e o bem-estar coletivo.", "As tarefas necessárias são executadas com aceitação e da melhor forma possível.", "Escrevem-se metas e progressos para consolidar novas condutas." ] }

];

// --- UI & Navigation ---

const mainScreens = ['welcomeScreen', 'assessmentScreen', 'progressScreen', 'knowledgeScreen', 'aiPlanScreen'];

const loadingEl = document.getElementById('loading');

window.showScreen = (screenId) => {

mainScreens.forEach(id => document.getElementById(id).classList.add('hidden'));

const screen = document.getElementById(screenId);

if(screen) {

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

screen.innerHTML = ''; // Clear previous content to avoid stale data

screen.classList.remove('hidden');

// Re-render content when showing screen

if (screenId === 'progressScreen') renderProgress();

else if (screenId === 'assessmentScreen') renderAssessment();

else if (screenId === 'aiPlanScreen') showAiPlanner();

else if (screenId === 'knowledgeScreen') renderKnowledgeBase();

else if (screenId === 'welcomeScreen') {

// Find the welcome screen div and render its static content

const welcomeContainer = document.getElementById('welcomeScreen');

welcomeContainer.innerHTML = `

<div class="bg-white p-6 rounded-lg shadow-md">

<h2 class="text-2xl font-semibold mb-4 text-gray-900">Bem-vindo(a)!</h2>

<p class="mb-6 text-gray-600">Este é o seu espaço para refletir e celebrar cada passo em direção a uma vida mais saudável.</p>

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

<button onclick="startAssessment()" class="w-full bg-blue-600 text-white font-semibold py-3 px-4 rounded-lg shadow hover:bg-blue-700 transition">Nova Avaliação</button>

<button onclick="showScreen('progressScreen')" class="w-full bg-green-600 text-white font-semibold py-3 px-4 rounded-lg shadow hover:bg-green-700 transition">Meu Progresso</button>

<button onclick="showAiPlanner()" class="w-full bg-purple-600 text-white font-semibold py-3 px-4 rounded-lg shadow hover:bg-purple-700 transition flex items-center justify-center gap-2">✨ Plano com IA</button>

<button onclick="showScreen('knowledgeScreen')" class="w-full bg-gray-600 text-white font-semibold py-3 px-4 rounded-lg shadow hover:bg-gray-700 transition">Conhecimento</button>

</div>

</div>`;

}

}

};

const showLoginScreen = () => {

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

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

loadingEl.classList.add('hidden');

};

const showMainApp = () => {

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

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

showScreen('welcomeScreen');

};

// --- Modals ---

const showModal = (title, message, buttons) => {

document.getElementById('modalTitle').textContent = title;

document.getElementById('modalMessage').textContent = message;

const buttonsContainer = document.getElementById('modalButtons');

buttonsContainer.innerHTML = '';

buttons.forEach(btn => {

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

buttonEl.textContent = btn.text;

buttonEl.className = btn.classes;

buttonEl.onclick = btn.onClick;

buttonsContainer.appendChild(buttonEl);

});

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

};

const hideModal = () => document.getElementById('modal').classList.add('hidden');

// --- Authentication & Session Logic ---

window.loginWithKey = async () => {

const loginError = document.getElementById('login-error');

loginError.textContent = '';

loadingEl.classList.remove('hidden');

let keyParts = Array.from(document.querySelectorAll('.key-input')).map(input => input.value.toUpperCase());

if (keyParts.some(part => part.length !== 4)) {

loginError.textContent = 'A chave deve ter 16 caracteres.';

loadingEl.classList.add('hidden');

return;

}

const key = keyParts.join('-');

try {

const licenseRef = doc(db, "licenses", key);

const licenseSnap = await getDoc(licenseRef);

if (!licenseSnap.exists()) throw new Error("Chave de acesso inválida.");

const licenseData = licenseSnap.data();

if (licenseData.status !== 'active') throw new Error("Esta chave de acesso não está ativa.");

if (licenseData.validUntil.toDate() < new Date()) throw new Error("Esta chave de acesso expirou.");

if (licenseData.userId && licenseData.userId !== userId) throw new Error("Chave já está associada a outra instalação.");

if (!licenseData.userId) {

await updateDoc(licenseRef, { userId: userId });

}

const sessionId = crypto.randomUUID();

await updateDoc(licenseRef, { currentSessionId: sessionId, lastLogin: Timestamp.now() });

localStorage.setItem('licenseKey', key);

localStorage.setItem('sessionId', sessionId);

await startAppSession(key, sessionId);

} catch (error) {

loginError.textContent = error.message;

loadingEl.classList.add('hidden');

}

};

window.logout = () => {

showModal('Confirmar Saída', 'Tem a certeza que deseja sair?', [

{ text: 'Cancelar', classes: 'px-4 py-2 bg-gray-200 text-gray-800 rounded-md hover:bg-gray-300', onClick: hideModal },

{ text: 'Sair', classes: 'px-4 py-2 bg-red-600 text-white rounded-md hover:bg-red-700', onClick: () => {

if (licenseListener) licenseListener();

localStorage.clear();

hideModal();

location.reload();

}}

]);

};

async function validateSession(key, sessionId) {

const licenseRef = doc(db, "licenses", key);

const licenseSnap = await getDoc(licenseRef);

if (!licenseSnap.exists() || licenseSnap.data().currentSessionId !== sessionId || licenseSnap.data().validUntil.toDate() < new Date()) {

if (licenseListener) licenseListener();

localStorage.clear();

showLoginScreen();

showModal('Sessão Expirada', 'A sua sessão foi terminada ou a chave expirou. Por favor, entre novamente.', [

{text: 'OK', classes: 'px-4 py-2 bg-blue-600 text-white rounded-md hover:bg-blue-700', onClick: hideModal}

]);

return false;

}

await startAppSession(key, sessionId);

return true;

}

async function startAppSession(key, sessionId) {

console.log(`Sessão iniciada com sucesso para a chave ${key}`);

// Fetch user's assessments from Firestore

const q = query(collection(db, `artifacts/${appId}/users/${userId}/assessments`));

const querySnapshot = await getDocs(q);

assessments = [];

querySnapshot.forEach((doc) => {

assessments.push({ id: doc.id, ...doc.data() });

});

assessments.sort((a, b) => a.createdAt.toMillis() - b.createdAt.toMillis());

showMainApp();

const licenseRef = doc(db, "licenses", key);

if(licenseListener) licenseListener();

// Listen for changes on the license document (e.g., remote logout)

licenseListener = onSnapshot(licenseRef, (doc) => {

if (!doc.exists() || doc.data().currentSessionId !== sessionId) {

if (licenseListener) licenseListener();

localStorage.clear();

showLoginScreen();

showModal('Sessão Terminada', 'Esta chave foi usada noutro dispositivo. A sua sessão aqui foi encerrada.', [

{text: 'OK', classes: 'px-4 py-2 bg-blue-600 text-white rounded-md hover:bg-blue-700', onClick: hideModal}

]);

}

});

}

// --- Assessment Logic ---

window.startAssessment = () => showScreen('assessmentScreen');

const renderAssessment = () => {

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

container.innerHTML = `

<div class="bg-white p-6 rounded-lg shadow-md">

<div class="flex justify-between items-start mb-4">

<div>

<h2 class="text-2xl font-semibold text-gray-900">Autoavaliação de Hábitos</h2>

<p class="text-gray-600">Atribua uma nota de 0 (hábito a iniciar) a 99 (hábito integrado).</p>

</div>

<button onclick="showScreen('welcomeScreen')" class="text-gray-500 hover:text-gray-800 text-2xl font-bold">&times; Voltar</button>

</div>

<div id="questionsContainer" class="space-y-8"></div>

<div class="mt-8 flex justify-end gap-4">

<button onclick="showScreen('welcomeScreen')" class="bg-gray-200 text-gray-800 font-semibold py-2 px-6 rounded-lg hover:bg-gray-300 transition">Cancelar</button>

<button id="submitAssessmentBtn" class="bg-blue-600 text-white font-semibold py-2 px-6 rounded-lg shadow hover:bg-blue-700 transition">Finalizar Avaliação</button>

</div>

</div>`;

const questionsContainer = container.querySelector('#questionsContainer');

let questionIndex = 0;

questions.forEach(category => {

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

categoryDiv.innerHTML = `<h3 class="text-xl font-semibold text-blue-800 border-b-2 border-blue-200 pb-2 mb-4">${category.category}</h3>`;

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

statementsList.className = 'space-y-6';

category.statements.forEach(statement => {

const id = `q${questionIndex}`;

statementsList.innerHTML += `

<div class="question-item">

<label for="${id}" class="block text-md font-medium text-gray-700 mb-2">${statement}</label>

<div class="flex items-center gap-4">

<input type="range" id="${id}" min="0" max="99" value="50" class="w-full h-2 bg-gray-200 rounded-lg appearance-none cursor-pointer">

<input type="number" id="${id}-value" min="0" max="99" value="50" class="font-bold text-blue-700 text-lg w-20 p-1 text-center bg-gray-100 rounded-md border-gray-300 focus:ring-blue-500 focus:border-blue-500">

</div>

</div>`;

questionIndex++;

});

categoryDiv.appendChild(statementsList);

questionsContainer.appendChild(categoryDiv);

});

questionsContainer.querySelectorAll('.question-item').forEach(item => {

const slider = item.querySelector('input[type="range"]');

const numberInput = item.querySelector('input[type="number"]');

slider.addEventListener('input', (e) => { numberInput.value = e.target.value; });

numberInput.addEventListener('input', (e) => {

let value = parseInt(e.target.value, 10);

if (isNaN(value)) return;

if (value > 99) { value = 99; e.target.value = 99; }

if (value < 0) { value = 0; e.target.value = 0; }

slider.value = value;

});

numberInput.addEventListener('blur', (e) => {

if(e.target.value === '') { e.target.value = 0; slider.value = 0; }

});

});

document.getElementById('submitAssessmentBtn').onclick = () => {

showModal('Finalizar Avaliação', 'Tem a certeza que deseja finalizar? As suas respostas serão guardadas.', [

{text: 'Cancelar', classes: 'px-4 py-2 bg-gray-200 text-gray-800 rounded-md hover:bg-gray-300', onClick: hideModal},

{text: 'Confirmar', classes: 'px-4 py-2 bg-blue-600 text-white rounded-md hover:bg-blue-700', onClick: async () => {

hideModal();

loadingEl.classList.remove('hidden');

const scores = Array.from(questionsContainer.querySelectorAll('input[type="range"]')).map(s => parseInt(s.value, 10));

const newAssessment = { createdAt: Timestamp.now(), scores: {}, overallScore: scores.reduce((a, b) => a + b, 0) / scores.length, categoryScores: {} };

let scoreIndex = 0;

questions.forEach(cat => {

const catScores = cat.statements.map(() => scores[scoreIndex++]);

newAssessment.categoryScores[cat.category] = catScores.reduce((a, b) => a + b, 0) / catScores.length;

});

scoreIndex = 0;

questions.forEach(cat => cat.statements.forEach(st => { newAssessment.scores[st] = scores[scoreIndex++]; }));

try {

const docRef = await addDoc(collection(db, `artifacts/${appId}/users/${userId}/assessments`), newAssessment);

assessments.push({ id: docRef.id, ...newAssessment });

assessments.sort((a, b) => a.createdAt.toMillis() - b.createdAt.toMillis());

showScreen('progressScreen');

} catch (e) { console.error("Error saving assessment:", e); showModal('Erro', 'Ocorreu um erro ao guardar a sua avaliação.', [{text: 'OK', classes: 'px-4 py-2 bg-blue-600 text-white rounded-md', onClick: hideModal}]); }

finally { loadingEl.classList.add('hidden'); }

}}

]);

};

};

// --- Progress Screen ---

const renderProgress = () => {

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

container.innerHTML = `

<div class="bg-white p-6 rounded-lg shadow-md">

<div class="flex justify-between items-start mb-6">

<h2 class="text-2xl font-semibold text-gray-900">O Seu Progresso</h2>

<button onclick="showScreen('welcomeScreen')" class="text-gray-500 hover:text-gray-800 text-2xl font-bold">&times; Voltar</button>

</div>

<div id="progressContent"></div>

</div>`;

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

if (assessments.length === 0) {

progressContent.innerHTML = `

<div class="text-center py-10">

<svg class="mx-auto h-12 w-12 text-gray-400" fill="none" viewBox="0 0 24 24" stroke="currentColor" aria-hidden="true"><path vector-effect="non-scaling-stroke" stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9 19v-6a2 2 0 00-2-2H5a2 2 0 00-2 2v6a2 2 0 002 2h2a2 2 0 002-2zm0 0V9a2 2 0 012-2h2a2 2 0 012 2v10m-6 0a2 2 0 002 2h2a2 2 0 002-2m0 0V5a2 2 0 012-2h2a2 2 0 012 2v14a2 2 0 01-2 2h-2a2 2 0 01-2-2z" /></svg>

<h3 class="mt-2 text-sm font-medium text-gray-900">Sem Avaliações</h3>

<p class="mt-1 text-sm text-gray-500">Comece a sua jornada fazendo a sua primeira avaliação.</p>

<div class="mt-6">

<button onclick="startAssessment()" class="inline-flex items-center px-4 py-2 border border-transparent shadow-sm text-sm font-medium rounded-md text-white bg-blue-600 hover:bg-blue-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-blue-500">

Fazer a minha primeira avaliação

</button>

</div>

</div>`;

return;

}

// If there are assessments, render the charts

progressContent.innerHTML = `

<div class="grid grid-cols-1 lg:grid-cols-5 gap-6">

<div class="lg:col-span-3 p-4 border rounded-lg bg-gray-50">

<h3 class="text-lg font-semibold text-gray-800 mb-4">Evolução Geral</h3>

<canvas id="overallProgressChart"></canvas>

</div>

<div class="lg:col-span-2 p-4 border rounded-lg bg-gray-50">

<h3 class="text-lg font-semibold text-gray-800 mb-4">Última Avaliação</h3>

<canvas id="latestAssessmentRadarChart"></canvas>

</div>

</div>

`;

// Destroy previous chart instances to prevent memory leaks on re-render

if (overallProgressChartInstance) overallProgressChartInstance.destroy();

if (radarChartInstance) radarChartInstance.destroy();

// Chart 1: Overall progress line chart

const overallCtx = document.getElementById('overallProgressChart').getContext('2d');

overallProgressChartInstance = new Chart(overallCtx, {

type: 'line',

data: {

labels: assessments.map(a => new Date(a.createdAt.seconds \* 1000).toLocaleDateString('pt-BR')),

datasets: [{

label: 'Pontuação Geral',

data: assessments.map(a => a.overallScore.toFixed(1)),

backgroundColor: 'rgba(59, 130, 246, 0.2)',

borderColor: 'rgba(59, 130, 246, 1)',

borderWidth: 2,

tension: 0.1,

fill: true

}]

},

options: {

responsive: true,

maintainAspectRatio: false,

scales: { y: { beginAtZero: true, max: 100 } }

}

});

// Chart 2: Latest assessment radar chart

const latestAssessment = assessments[assessments.length - 1];

const radarCtx = document.getElementById('latestAssessmentRadarChart').getContext('2d');

radarChartInstance = new Chart(radarCtx, {

type: 'radar',

data: {

labels: Object.keys(latestAssessment.categoryScores).map(c => c.split(' e ')[0]), // Shorten labels

datasets: [{

label: 'Pontuação por Categoria',

data: Object.values(latestAssessment.categoryScores).map(s => s.toFixed(1)),

backgroundColor: 'rgba(139, 92, 246, 0.2)',

borderColor: 'rgba(139, 92, 246, 1)',

borderWidth: 2

}]

},

options: {

responsive: true,

maintainAspectRatio: false,

scales: { r: { beginAtZero: true, max: 100, pointLabels: { font: { size: 10 } } } }

}

});

};

// --- Placeholder functions for other screens ---

window.showAiPlanner = () => {

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

container.innerHTML = `<div class="bg-white p-6 rounded-lg shadow-md"><div class="flex justify-between items-start mb-4"><h2 class="text-2xl font-semibold text-gray-900">Plano com IA</h2><button onclick="showScreen('welcomeScreen')" class="text-gray-500 hover:text-gray-800 text-2xl font-bold">&times; Voltar</button></div><p>Esta funcionalidade está em desenvolvimento. Em breve, uma IA irá analisar os seus resultados e criar um plano de ação personalizado para si.</p></div>`;

showScreen('aiPlanScreen');

};

const renderKnowledgeBase = () => {

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

container.innerHTML = `<div class="bg-white p-6 rounded-lg shadow-md"><div class="flex justify-between items-start mb-4"><h2 class="text-2xl font-semibold text-gray-900">Base de Conhecimento</h2><button onclick="showScreen('welcomeScreen')" class="text-gray-500 hover:text-gray-800 text-2xl font-bold">&times; Voltar</button></div><p>Esta área está em desenvolvimento. Em breve, encontrará aqui artigos, dicas e informações para apoiar a sua jornada.</p></div>`;

showScreen('knowledgeScreen');

};

// --- Firebase Initialization & App Start ---

async function initialize() {

try {

const app = initializeApp(firebaseConfig);

auth = getAuth(app);

db = getFirestore(app);

onAuthStateChanged(auth, async (user) => {

if (user) {

// User is signed in.

userId = user.uid;

document.getElementById('userIdDisplay').textContent = user.uid;

const storedKey = localStorage.getItem('licenseKey');

const storedSession = localStorage.getItem('sessionId');

if (storedKey && storedSession) {

await validateSession(storedKey, storedSession);

} else {

showLoginScreen();

}

}

// If no user, the sign-in attempt below will trigger this listener again.

});

// Attempt to sign in if there's no current user.

if (!auth.currentUser) {

if (typeof \_\_initial\_auth\_token !== 'undefined' && \_\_initial\_auth\_token) {

await signInWithCustomToken(auth, \_\_initial\_auth\_token);

} else {

await signInAnonymously(auth);

}

}

} catch (error) {

console.error("Firebase Initialization Error:", error);

loadingEl.innerHTML = `<p class="text-red-600">Erro ao inicializar. Verifique a consola ou recarregue a página.</p>`;

}

}

window.onload = () => {

// License Key Input Auto-Tabbing

document.getElementById('key-input-container').addEventListener('input', (e) => {

const target = e.target;

const index = parseInt(target.dataset.index, 10);

if (target.value.length >= 4 && index < 3) {

// Find the next input element to focus on

const nextInput = document.querySelector(`.key-input[data-index='${index + 1}']`);

if(nextInput) nextInput.focus();

}

});

initialize();

};

</script>

</body>

</html>