<!DOCTYPE html>

<html lang="pl">

<head>

<meta charset="UTF-8">

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

<title>Hub AI - Symulacja</title>

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

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

<style>

body {

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

background-color: #0d0d1a;

color: #ffffff;

overflow: auto;

display: flex;

flex-direction: column;

justify-content: flex-start;

align-items: center;

min-height: 100vh;

padding: 1rem;

gap: 2rem;

}

.container {

position: relative;

width: 100%;

height: 100%;

display: flex;

flex-direction: column;

justify-content: flex-start;

align-items: center;

max-width: 800px;

}

.core {

width: 15rem;

height: 15rem;

background: linear-gradient(135deg, #4c00ff, #b400ff);

border-radius: 50%;

position: relative;

box-shadow: 0 0 40px rgba(180, 0, 255, 0.7);

display: flex;

justify-content: center;

align-items: center;

flex-direction: column;

animation: pulse 4s infinite cubic-bezier(0.65, 0.05, 0.36, 1);

transition: all 0.5s ease-in-out;

border: 2px solid transparent;

}

.core:hover {

box-shadow: 0 0 60px rgba(180, 0, 255, 1);

transform: scale(1.05);

}

@keyframes pulse {

0% {

box-shadow: 0 0 40px rgba(180, 0, 255, 0.7);

}

50% {

box-shadow: 0 0 60px rgba(180, 0, 255, 1), 0 0 80px rgba(76, 0, 255, 0.5);

}

100% {

box-shadow: 0 0 40px rgba(180, 0, 255, 0.7);

}

}

.core-glow {

width: 15rem;

height: 15rem;

background: rgba(180, 0, 255, 0.2);

border-radius: 50%;

position: absolute;

animation: innerGlow 3s infinite ease-in-out;

}

@keyframes innerGlow {

0% {

transform: scale(1);

opacity: 0.2;

}

50% {

transform: scale(1.1);

opacity: 0.5;

}

100% {

transform: scale(1);

opacity: 0.2;

}

}

/\* Additional edge glow effect \*/

.core-edge-glow {

width: 15.5rem;

height: 15.5rem;

border-radius: 50%;

position: absolute;

top: -0.25rem;

left: -0.25rem;

background: linear-gradient(45deg, #ff00ff, #00ffff);

z-index: -1;

filter: blur(15px);

opacity: 0;

animation: edgeGlow 5s infinite;

}

@keyframes edgeGlow {

0%, 100% { opacity: 0.4; }

50% { opacity: 0.8; }

}

.core-content {

text-align: center;

font-size: 1.25rem;

font-weight: 700;

color: #ffffff;

z-index: 10;

}

.value-box {

background-color: rgba(255, 255, 255, 0.1);

padding: 0.5rem 1rem;

border-radius: 0.5rem;

margin: 0.5rem 0;

font-size: 1rem;

min-width: 100px;

text-align: center;

}

.btn {

background: linear-gradient(90deg, #4c00ff, #b400ff);

color: white;

padding: 0.75rem 1.5rem;

border-radius: 9999px;

font-weight: 700;

transition: all 0.3s ease;

box-shadow: 0 4px 15px rgba(180, 0, 255, 0.4);

border: none;

cursor: pointer;

}

.btn:hover {

box-shadow: 0 4px 25px rgba(180, 0, 255, 0.7);

transform: translateY(-2px);

}

.btn-destructive {

background: linear-gradient(90deg, #ff0000, #ff4d4d);

box-shadow: 0 4px 15px rgba(255, 0, 0, 0.4);

}

.btn-destructive:hover {

box-shadow: 0 4px 25px rgba(255, 0, 0, 0.7);

}

.output-box {

background-color: #1a1a2e;

border: 2px solid #2e2e4a;

border-radius: 1rem;

padding: 1.5rem;

}

.status-box {

background-color: #2a2a40;

border-radius: 0.5rem;

padding: 0.75rem 1rem;

display: flex;

align-items: center;

gap: 1rem;

margin-top: 1rem;

transition: background-color 0.3s ease;

}

.status-box.success {

background-color: #0c3326;

}

.status-box.error {

background-color: #4b1f1f;

}

.status-indicator {

width: 1rem;

height: 1rem;

border-radius: 50%;

background-color: #8c8c9e;

transition: background-color 0.3s ease;

}

.status-box.success .status-indicator {

background-color: #21a153;

}

.status-box.error .status-indicator {

background-color: #e53e3e;

}

.gok-ai-title {

font-size: 2.25rem;

font-weight: 700;

background: linear-gradient(90deg, #6b46c1, #b400ff, #e53e3e);

-webkit-background-clip: text;

-webkit-text-fill-color: transparent;

text-align: center;

}

/\* Calculator section \*/

.calculator-container {

width: 100%;

background-color: #1a1a2e;

padding: 2rem;

border-radius: 1.5rem;

box-shadow: 0 10px 30px rgba(0, 0, 0, 0.3);

}

.input-group {

display: flex;

align-items: center;

margin-bottom: 1rem;

gap: 1rem;

}

.input-group label {

width: 3rem;

font-weight: 700;

}

.input-group input[type="range"] {

-webkit-appearance: none;

width: 100%;

height: 8px;

background: #2e2e4a;

border-radius: 5px;

outline: none;

transition: opacity .2s;

}

.input-group input[type="range"]::-webkit-slider-thumb {

-webkit-appearance: none;

appearance: none;

width: 20px;

height: 20px;

border-radius: 50%;

background: #b400ff;

cursor: pointer;

box-shadow: 0 0 5px rgba(180, 0, 255, 0.7);

}

.input-group input[type="range"]::-moz-range-thumb {

width: 20px;

height: 20px;

border-radius: 50%;

background: #b400ff;

cursor: pointer;

}

.input-group span {

width: 2rem;

text-align: center;

}

.results-box {

background-color: #2a2a40;

padding: 1.5rem;

border-radius: 1rem;

margin-top: 1.5rem;

}

.results-box p {

margin-bottom: 0.5rem;

}

.glow-button {

position: relative;

background-color: transparent;

border: 2px solid;

border-image: linear-gradient(45deg, #4c00ff, #b400ff) 1;

padding: 0.75rem 1.5rem;

border-radius: 9999px;

color: white;

font-weight: 700;

overflow: hidden;

transition: all 0.5s ease;

cursor: pointer;

}

.glow-button::before {

content: '';

position: absolute;

top: -50%;

left: -50%;

width: 200%;

height: 200%;

background: radial-gradient(circle, #b400ff, transparent 50%);

opacity: 0;

transform: scale(0);

transition: all 0.8s ease;

}

.glow-button:hover::before {

transform: scale(1);

opacity: 0.7;

}

.glow-button:hover {

box-shadow: 0 0 20px #b400ff, inset 0 0 10px #4c00ff;

}

/\* Chat section \*/

.chat-container {

width: 100%;

background-color: #1a1a2e;

padding: 2rem;

border-radius: 1.5rem;

box-shadow: 0 10px 30px rgba(0, 0, 0, 0.3);

display: flex;

flex-direction: column;

gap: 1rem;

}

.chat-history {

height: 300px;

overflow-y: auto;

background-color: #0d0d1a;

border-radius: 0.75rem;

padding: 1rem;

display: flex;

flex-direction: column;

gap: 0.75rem;

}

.chat-message {

padding: 0.75rem 1rem;

border-radius: 1rem;

max-width: 80%;

}

.chat-message.user {

background-color: #3b3b55;

align-self: flex-end;

}

.chat-message.ai {

background-color: #b400ff;

align-self: flex-start;

}

.chat-input-container {

display: flex;

gap: 1rem;

}

.chat-input-container input {

flex-grow: 1;

background-color: #2a2a40;

border: none;

padding: 0.75rem 1rem;

border-radius: 9999px;

color: white;

outline: none;

}

.loading-dots {

align-self: flex-start;

background-color: #b400ff;

padding: 0.75rem 1rem;

border-radius: 1rem;

display: inline-flex;

gap: 0.25rem;

}

.dot {

width: 8px;

height: 8px;

background-color: white;

border-radius: 50%;

animation: bounce 1.4s infinite ease-in-out both;

}

.dot:nth-child(1) { animation-delay: -0.32s; }

.dot:nth-child(2) { animation-delay: -0.16s; }

@keyframes bounce {

0%, 80%, 100% { transform: scale(0); }

40% { transform: scale(1.0); }

}

</style>

</head>

<body class="flex flex-col items-center justify-start min-h-screen p-4 bg-gray-950 text-white font-inter">

<div class="container flex flex-col items-center gap-8">

<h1 class="text-4xl font-bold text-center gok-ai-title">Hub AI</h1>

<!-- Main META-GENIUSZ component -->

<div class="core-container relative flex justify-center items-center">

<div class="core-edge-glow"></div>

<div class="core relative">

<div class="core-glow"></div>

<div class="core-content z-10">

<h2 class="text-2xl font-bold mb-2">META-GENIUSZ</h2>

<p class="text-xs mb-4 text-gray-400">Silnik Jądrowy GOK:AI</p>

<div class="value-box">

<span id="internalValueW">Wartość Wewnętrzna (W): 0</span>

</div>

<div class="value-box">

<span id="destructionValueD">Wartość Destrukcji (D): 0</span>

</div>

</div>

</div>

</div>

<!-- Section with buttons and text input -->

<div class="w-full max-w-2xl flex flex-col gap-4">

<div class="relative w-full">

<input type="text" id="userInput" class="w-full p-4 bg-gray-800 rounded-xl border-2 border-transparent focus:border-purple-600 focus:outline-none transition-all duration-300" placeholder="Wpisz polecenie...">

<div class="absolute inset-y-0 right-0 flex items-center pr-3">

<button id="clearButton" class="text-gray-400 hover:text-white transition-colors duration-200 focus:outline-none">

<svg class="w-5 h-5" fill="currentColor" viewBox="0 0 20 20" xmlns="http://www.w3.org/2000/svg">

<path fill-rule="evenodd" d="M10 18a8 8 0 100-16 8 8 0 000 16zM8.707 7.293a1 1 0 00-1.414 1.414L8.586 10l-1.293 1.293a1 1 0 101.414 1.414L10 11.414l1.293 1.293a1 1 0 001.414-1.414L11.414 10l1.293-1.293a1 1 0 00-1.414-1.414L10 8.586 8.707 7.293z" clip-rule="evenodd"></path>

</svg>

</button>

</div>

</div>

<div class="flex flex-wrap justify-center gap-4">

<button id="generateTextButton" class="btn">Generuj ✨</button>

<button id="generateImageButton" class="btn">Generuj Obraz ✨</button>

<button id="analyzeImageButton" class="btn">Analizuj Obraz ✨</button>

<button id="evolutionLogButton" class="btn">Dziennik Ewolucji GOK:AI ✨</button>

</div>

<div class="flex flex-wrap justify-center gap-4">

<button id="generateMButton" class="btn">Generuj Macierz Motywacji M ✨</button>

<button id="analyzeMButton" class="btn">Analizuj Wzorzec Ewolucji M ✨</button>

<button id="generateDCodeButton" class="btn btn-destructive">Generuj Kod Źródłowy (Dezintegracja) 💥</button>

<button id="analyzeDButton" class="btn btn-destructive">Analizuj Wzorce Destrukcji (Rozbicie Wzorca) 💥</button>

</div>

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

<button id="runPipelineButton" class="glow-button">Uruchom Silnik GOK:AI ✨</button>

</div>

</div>

<!-- Feedback message box -->

<div id="messageBox" class="w-full max-w-2xl bg-gray-800 p-4 rounded-xl text-center text-gray-300 transition-all duration-500 transform scale-0 opacity-0">

<span id="messageText"></span>

</div>

<!-- GOK:AI Success Calculator section -->

<div class="calculator-container w-full max-w-2xl mt-8">

<h2 class="text-2xl font-bold text-center mb-6 gok-ai-title">Kalkulator Sukcesu GOK:AI</h2>

<p class="text-center text-gray-400 mb-6">Wprowadź wartości dla parametrów, aby obliczyć prawdopodobieństwo sukcesu projektu.</p>

<div class="input-group">

<label for="wValue">W:</label>

<input type="range" id="wValue" min="1" max="10" value="5">

<span id="wValueDisplay">5</span>

</div>

<div class="input-group">

<label for="mValue">M:</label>

<input type="range" id="mValue" min="1" max="10" value="5">

<span id="mValueDisplay">5</span>

</div>

<div class="input-group">

<label for="dValue">D:</label>

<input type="range" id="dValue" min="1" max="10" value="5">

<span id="dValueDisplay">5</span>

</div>

<div class="input-group">

<label for="cValue">C:</label>

<input type="range" id="cValue" min="1" max="10" value="5">

<span id="cValueDisplay">5</span>

</div>

<div class="input-group">

<label for="aValue">A:</label>

<input type="range" id="aValue" min="1" max="10" value="5">

<span id="aValueDisplay">5</span>

</div>

<div class="input-group">

<label for="eValue">E:</label>

<input type="range" id="eValue" min="1" max="10" value="5">

<span id="eValueDisplay">5</span>

</div>

<div class="input-group">

<label for="tValue">T:</label>

<input type="range" id="tValue" min="1" max="10" value="5">

<span id="tValueDisplay">5</span>

</div>

<button id="calculateButton" class="btn w-full mt-4">Oblicz Sukces GOK:AI</button>

<div id="resultsBox" class="results-box mt-6 hidden">

<h3 class="text-xl font-bold mb-4 text-center">Wyniki Symulacji</h3>

<p><strong>Faza Rozwoju:</strong> <span id="developmentPhase">...</span></p>

<p><strong>Prawdopodobieństwo Sukcesu:</strong> <span id="successProbability">...</span></p>

<div id="statusBox" class="status-box">

<span id="statusIndicator" class="status-indicator"></span>

<p><strong>Status Projektu Głównego:</strong> <span id="projectStatus">...</span></p>

</div>

</div>

</div>

<!-- Additional section: Chat with God's Brain -->

<div class="chat-container w-full max-w-2xl mt-8">

<h2 class="text-2xl font-bold text-center mb-2 gok-ai-title">Chat z Mózgiem Boga</h2>

<div id="chatHistory" class="chat-history">

<!-- Messages will be added here dynamically -->

</div>

<div class="chat-input-container">

<input type="text" id="chatInput" placeholder="Zadaj pytanie Mózgowi Boga...">

<button id="sendChatButton" class="btn">Wyślij</button>

</div>

</div>

<!-- Additional section: Project Vision Analysis -->

<div class="calculator-container w-full max-w-2xl mt-8">

<h2 class="text-2xl font-bold text-center mb-6 gok-ai-title">Analiza Wizji Projektu</h2>

<p class="text-center text-gray-400 mb-6">Przeanalizuj wizję na podstawie Wartości Wewnętrznej (W) i Destrukcji (D).</p>

<div class="input-group">

<label for="visionWValue">W:</label>

<input type="range" id="visionWValue" min="1" max="10" value="5">

<span id="visionWValueDisplay">5</span>

</div>

<div class="input-group">

<label for="visionDValue">D:</label>

<input type="range" id="visionDValue" min="1" max="10" value="5">

<span id="visionDValueDisplay">5</span>

</div>

<button id="analyzeVisionButton" class="btn w-full mt-4">Analizuj Wizję ✨</button>

<div id="visionAnalysisOutput" class="results-box mt-6 hidden">

<p id="visionAnalysisResult"></p>

</div>

</div>

</div>

</body>

<script>

document.addEventListener('DOMContentLoaded', () => {

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

// Calculator elements

const wValueRange = document.getElementById('wValue');

const mValueRange = document.getElementById('mValue');

const dValueRange = document.getElementById('dValue');

const cValueRange = document.getElementById('cValue');

const aValueRange = document.getElementById('aValue');

const eValueRange = document.getElementById('eValue');

const tValueRange = document.getElementById('tValue');

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

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

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

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

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

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

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

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

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

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

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

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

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

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

// Chat elements

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

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

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

// Vision analysis elements

const visionWValueRange = document.getElementById('visionWValue');

const visionDValueRange = document.getElementById('visionDValue');

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

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

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

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

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

let internalW = 0;

let destructionD = 0;

// Simulation functions

function showMessage(text, isError = false) {

messageText.textContent = text;

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

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

if (isError) {

messageBox.classList.add('bg-red-900', 'border-red-700');

messageBox.classList.remove('bg-gray-800', 'border-gray-600');

} else {

messageBox.classList.remove('bg-red-900', 'border-red-700');

messageBox.classList.add('bg-gray-800', 'border-gray-600');

}

setTimeout(() => {

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

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

}, 3000);

}

function updateCoreValues() {

internalW = Math.floor(Math.random() \* 100);

destructionD = Math.floor(Math.random() \* 100);

internalValueW.textContent = `Wartość Wewnętrzna (W): ${internalW}`;

destructionValueD.textContent = `Wartość Destrukcji (D): ${destructionD}`;

}

function generateText(prompt) {

updateCoreValues();

showMessage(`Generowanie tekstu na podstawie "${prompt}"...`);

// AI text generation code would go here

}

function generateImage(prompt) {

updateCoreValues();

showMessage(`Generowanie obrazu na podstawie "${prompt}"...`);

// AI image generation code would go here

}

function analyzeImage(file) {

updateCoreValues();

showMessage(`Analiza obrazu z pliku "${file.name}"...`);

// AI image analysis code would go here

}

function generateEvolutionLog() {

updateCoreValues();

showMessage("Generowanie Dziennika Ewolucji GOK:AI...");

}

function generateMotivationMatrix() {

updateCoreValues();

showMessage("Generowanie Macierzy Motywacji M...");

}

function analyzeMotivationPattern() {

updateCoreValues();

showMessage("Analiza Wzorca Ewolucji M...");

}

function generateSourceCode(prompt) {

updateCoreValues();

showMessage(`Generowanie kodu źródłowego (Dezintegracja) na podstawie "${prompt}"...`, true);

}

function performDestructiveAnalysis(prompt) {

updateCoreValues();

showMessage(`Analiza Wzorców Destrukcji (Rozbicie Wzorca) dla "${prompt}"...`, true);

}

function runPipeline() {

updateCoreValues();

showMessage("Uruchamianie Silnika GOK:AI...");

}

// Calculator functions

function updateRangeDisplay(rangeElement, displayElement) {

displayElement.textContent = rangeElement.value;

}

function calculateSuccess() {

const w = parseInt(wValueRange.value);

const m = parseInt(mValueRange.value);

const d = parseInt(dValueRange.value);

const c = parseInt(cValueRange.value);

const a = parseInt(aValueRange.value);

const e = parseInt(eValueRange.value);

const t = parseInt(tValueRange.value);

const successFactor = (w \* 0.3 + m \* 0.2 + c \* 0.15 + a \* 0.15 + e \* 0.1 + t \* 0.1) - (d \* 0.2);

const successPercentage = Math.max(0, Math.min(100, Math.floor(successFactor \* 10)));

let phase = "";

let status = "";

let statusClass = "";

if (successPercentage >= 80) {

phase = "Optymalizacja Ewolucyjna";

status = "PROJEKT GOTOWY DO WDROŻENIA";

statusClass = "success";

} else if (successPercentage >= 60) {

phase = "Integracja Algorytmiczna";

status = "W TRAKCIE OPTYMALIZACJI";

statusClass = "success";

} else if (successPercentage >= 40) {

phase = "Testy Wstępne";

status = "WYMAGA DODATKOWYCH TESTÓW";

statusClass = "error";

} else {

phase = "Dezintegracja Strukturalna";

status = "KRYTYCZNY BŁĄD PROJEKTU";

statusClass = "error";

}

developmentPhase.textContent = phase;

successProbability.textContent = `${successPercentage}%`;

projectStatus.textContent = status;

statusBox.className = `status-box ${statusClass}`;

resultsBox.classList.remove('hidden');

}

// Chat functions

function appendChatMessage(message, type) {

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

messageElement.classList.add('chat-message', type);

messageElement.textContent = message;

chatHistory.appendChild(messageElement);

chatHistory.scrollTop = chatHistory.scrollHeight;

return messageElement; // Return the created element for further use (e.g., loading indicator)

}

async function sendChatMessage() {

const message = chatInput.value.trim();

if (message) {

appendChatMessage(message, 'user');

chatInput.value = '';

// Add a loading indicator while waiting for the AI response

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

loadingDots.classList.add('loading-dots');

loadingDots.innerHTML = `

<div class="dot"></div>

<div class="dot"></div>

<div class="dot"></div>

`;

chatHistory.appendChild(loadingDots);

chatHistory.scrollTop = chatHistory.scrollHeight;

try {

const apiKey = "";

const apiUrl = `https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash-preview-05-20:generateContent?key=${apiKey}`;

const payload = {

contents: [{ parts: [{ text: message }] }],

};

const response = await fetch(apiUrl, {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify(payload)

});

const result = await response.json();

if (result.candidates && result.candidates.length > 0 &&

result.candidates[0].content && result.candidates[0].content.parts &&

result.candidates[0].content.parts.length > 0) {

const aiResponse = result.candidates[0].content.parts[0].text;

appendChatMessage(aiResponse, 'ai');

} else {

throw new Error('Invalid API response structure');

}

} catch (error) {

console.error("API error:", error);

appendChatMessage("Wystąpił błąd podczas komunikacji z Mózgiem Boga. Spróbuj ponownie.", 'ai');

} finally {

loadingDots.remove(); // Remove the loading indicator

}

}

}

// Vision analysis functions

function analyzeVision() {

const w = parseInt(visionWValueRange.value);

const d = parseInt(visionDValueRange.value);

let resultText = "";

if (w > d + 2) {

resultText = "Analiza Wizji: Twoja wizja charakteryzuje się silną Wartością Wewnętrzną. Jest to klucz do sukcesu, ale upewnij się, że nie ignorujesz potencjalnych wzorców destrukcji.";

} else if (d > w + 2) {

resultText = "Analiza Wizji: Wzorce Destrukcji (D) przeważają. Wizja może być niestabilna. Konieczna jest ponowna ocena założeń i wzmocnienie Wartości Wewnętrznej.";

} else {

resultText = "Analiza Wizji: Wizja jest w stanie równowagi. Potencjał zarówno do sukcesu, jak i do dezintegracji jest wysoki. Kluczem jest monitorowanie obu wartości.";

}

visionAnalysisResult.textContent = resultText;

visionAnalysisOutput.classList.remove('hidden');

}

// Event listeners

generateTextButton.addEventListener('click', () => {

const prompt = userInput.value.trim();

if (prompt) {

generateText(prompt);

} else {

showMessage("Proszę wpisać polecenie.", true);

}

});

generateImageButton.addEventListener('click', () => {

const prompt = userInput.value.trim();

if (prompt) {

generateImage(prompt);

} else {

showMessage("Proszę wpisać polecenie do wygenerowania obrazu.", true);

}

});

clearButton.addEventListener('click', () => {

userInput.value = '';

showMessage("Polecenie zostało wyczyszczone.");

});

// Event listener for file input (simulation)

analyzeImageButton.addEventListener('click', () => {

const fileInput = document.createElement('input');

fileInput.type = 'file';

fileInput.accept = 'image/\*';

fileInput.onchange = (event) => {

const file = event.target.files[0];

if (file) {

analyzeImage(file);

} else {

showMessage("Proszę wybrać plik obrazu do analizy.", true);

}

};

fileInput.click();

});

// Calculator event listeners

wValueRange.addEventListener('input', () => updateRangeDisplay(wValueRange, wValueDisplay));

mValueRange.addEventListener('input', () => updateRangeDisplay(mValueRange, mValueDisplay));

dValueRange.addEventListener('input', () => updateRangeDisplay(dValueRange, dValueDisplay));

cValueRange.addEventListener('input', () => updateRangeDisplay(cValueRange, cValueDisplay));

aValueRange.addEventListener('input', () => updateRangeDisplay(aValueRange, aValueDisplay));

eValueRange.addEventListener('input', () => updateRangeDisplay(eValueRange, eValueDisplay));

tValueRange.addEventListener('input', () => updateRangeDisplay(tValueRange, tValueDisplay));

calculateButton.addEventListener('click', calculateSuccess);

// Vision analysis event listeners

visionWValueRange.addEventListener('input', () => updateRangeDisplay(visionWValueRange, visionWValueDisplay));

visionDValueRange.addEventListener('input', () => updateRangeDisplay(visionDValueRange, visionDValueDisplay));

evolutionLogButton.addEventListener('click', generateEvolutionLog);

generateMButton.addEventListener('click', generateMotivationMatrix);

analyzeMButton.addEventListener('click', analyzeMotivationPattern);

generateDCodeButton.addEventListener('click', () => {

const prompt = userInput.value.trim();

if (prompt) {

generateSourceCode(prompt);

} else {

showMessage("Proszę wpisać koncepcję do dezintegracji.", true);

}

});

analyzeDButton.addEventListener('click', () => {

const prompt = userInput.value.trim();

if (prompt) {

performDestructiveAnalysis(prompt);

} else {

showMessage("Proszę wpisać tekst do analizy destrukcyjnej.", true);

}

});

runPipelineButton.addEventListener('click', runPipeline);

// New event listeners for chat and vision analysis

sendChatButton.addEventListener('click', sendChatMessage);

chatInput.addEventListener('keydown', (event) => {

if (event.key === 'Enter') {

sendChatMessage();

}

});

analyzeVisionButton.addEventListener('click', analyzeVision);

});

</script>

</html>